Ατομική Διπλωματική Εργασία

ΑΝΙΧΝΕΥΣΗ ΕΠΙΘΕΣΕΩΝ ΣΕ ΔΙΚΤΥΑ IOT ΜΕ ΧΡΗΣΗ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Αργυρίδης Σταύρος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάϊος 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανίχνευση επιθέσεων σε δίκτυα IoT με την χρήση μηχανικής μάθησης

Αργυρίδης Σταύρος

Επιβλέπων Καθηγητής

Δρ. Βάσος Βασιλείου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2021

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέπων καθηγητή μου Δρ. Βάσο Βασιλείου, ο οποίος μου έδωσε την ευκαιρία, στα πλαίσια της διπλωματικής εργασίας, να ασχοληθώ ένα ενδιαφέρον, για εμένα, θέμα στον κλάδο της πληροφορικής, το οποίο θα ήθελα να ασχοληθώ και στο μέλλον, αλλά και για την καθοδήγησή του κατά την υλοποίηση της εργασίας.

Ιδιαίτερες ευχαριστίες στην Δρ. Χριστιάνα Ιωάννου, ερευνητή στο Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου. Με τη βοήθειά της ήμουν σε θέση να καταλάβω σχετικές πληροφορίες για διάφορους αλγορίθμους μηχανικής μάθησης. Θέλω να εκφράσω την ευγνωμοσύνη μου για την προσπάθεια της για παροχή βοηθητικού εργαλείου στο Τμήμα Πληροφορικής.

Περίληψη

Στόχος της διπλωματικής μου εργασίας για την χρονιά 2020-2021, είναι η ανίχνευση επιθέσεων σε ένα IoT (internet of things) δίκτυο, με την χρήση δύο unsupervised machine learning αλγορίθμων, τον Isolation Forest και τον Self-Organizing Map (SOM). Συγκεκριμένα, πρόκειται για μια πειραματική εργασία στην οποία με την χρήση των πιο πάνω αλγορίθμων μηχανικής μάθησης προσπαθώ να αναλύσω και να βγάλω κάποια συμπεράσματα για τα αποτελέσματα που πήρα στην ανίχνευση συγκεκριμένων επιθέσεων σε ένα IoT δίκτυο.

Το IoT δίκτυο είναι στην ουσία ένα simulation το οποίο αποτελείται από nodes. Τα nodes ακολουθούν ένα ελαφρύ πρωτόκολλο επικοινωνίας, το WSP (Weighted Shortest Path). Μέσα στο δίκτυο αποτελείται ο κεντρικός node (Sink) και για την συγκεκριμένη εργασία χρησιμοποιήθηκαν 3 τοπολογίες, Sink in the Middle, Sink in the Top και Sink random.

Με την βοήθεια του COOJA simulator και του FIT IoT-Lab platform δημιουργήθηκαν οι επιθέσεις και χρησιμοποιήθηκαν τα δεδομένα τα οποία πάρθηκαν από αυτά τα simulators (2 datasets, COOJA_DATA και FIT_DATA). Έτσι χρησιμοποιώντας αυτά τα δεδομένα, υλοποιήθηκαν 2 είδη μοντέλων (intrusion detection systems) ένα χρησιμοποιώντας τον SOM και το άλλο χρησιμοποιώντας τον Isolation Forest, για ανίχνευση αυτών των επιθέσεων αλλά και για ανάλυση της συμπεριφοράς του δικτύου στις περιπτώσεις όπου υπάρχουν επιθέσεις. Τα μοντέλα έγιναν trained και evaluated για τα δύο διαφορετικά datasets (COOJA και FIT) και υλοποιήθηκαν ξεχωριστά για κάθε είδος επίθεσης. Να σημειώσω ότι έγινε ανίχνευση τόσο σε local (δηλαδή μεταξύ μόνο του sink node και του malicious node), όσο και σε network(μεταξύ όλου του δικτύου). Οι επιθέσεις που χρησιμοποιήθηκαν για το πρόβλημα είναι οι εξής: *Sinkhole, Blackhole(block node & forwarding ratio), Selective Forward(block node & forwarding ratio).*

Η μεθοδολογία που χρησιμοποιήθηκε για την υλοποίηση των detection models για τις πιο πάνω επιθέσεις είναι η εξής: (α) εκκίνηση των επιθέσεων στο experimental environment, (β) συλλογή των δεδομένων και δημιουργία δύο διαφορετικών data sets (COOJA_DATA & FIT_DATA), (γ) δημιουργία των μοντέλων χρησιμοποιώντας τους 2 διαφορετικούς αλγόριθμους και τα δύο διαφορετικά datasets, σε 3 διαφορετικές τοπολογίες (middle, top, random), (δ) evaluation των μοντέλων και ανάλυση των αποτελεσμάτων.

Τέλος, αυτή η έρευνα υλοποιήθηκε με την χρήση της γλώσσας προγραμματισμού python και του IDE Jupyter το οποίο είναι ειδικό εργαλείο για επίλυση προβλημάτων μηχανικής μάθησης.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Γενική εισαγωγή

Το Διαδίκτυο των πραγμάτων (IoT) είναι η διασύνδεση έξυπνων συσκευών για παρακολούθηση, ανάλυση και έλεγχο φυσικών περιβαλλόντων. Οι έξυπνες συσκευές ή smart devices μπορούν να περιλαμβάνουν τεχνολογίες όπως αισθητήρες, ενεργοποιητές και ειδικό λογισμικό, όλα προσαρμοσμένα για να εκπληρώσουν τη λειτουργικότητα του IoT. Έξυπνες πόλεις, Έξυπνα σπίτια, Το Smart Agriculture είναι μερικά παραδείγματα που χρησιμοποιούν το IoT για να επωφεληθούν από την αυτοματοποίηση καθημερινών εργασιών με σκοπό να διευκολυνθεί ο τρόπος της ζωής και γίνετε πιο αποτελεσματικοί.

Όμως ένα διαδίκτυο των πραγμάτων μπορεί να είναι ευάλωτο σε διάφορες επιθέσεις. Ένας εισβολέας μπορεί να παρακολουθεί, να μεταδίδει μηνύματα και να εισάγει ψευδείς πληροφορίες

στο δίκτυο. Συμβιβαστικές συσκευές κόμβου εντός των δικτύων IoT μπορεί να οδηγήσουν σε ανακριβή και / ή παραπλανητικές πληροφορίες, αποτυγχάνοντας έτσι στην επίτευξη του στόχου του δικτύου. Η έγκαιρη ανίχνευση των παραβιασμένων κόμβων μπορεί να μειώσει τη ζημιά που προκλήθηκε από την επίθεση.

Τα συστήματα ανίχνευσης ή αλλιώς intrusion detection systems (IDs) έχουν σκοπό με διάφορες τεχνικές να ανιχνεύουν όσο καλύτερα και όσο πιο αποτελεσματικά γίνεται αυτές τις επιθέσεις με αποτέλεσμα να έχουμε ένα ασφαλές δίκτυο. Στις μέρες μας η τεχνητή νοημοσύνη (AI) και τα παρακλάδια της όπως η μηχανική μάθηση, χαρακτηρίζονται ως πυλώνες στον κόσμο της πληροφορικής και ένα από τα πιο επίκαιρα θέματα για επίλυση πολλών προβλημάτων, όπως σε ασφάλεια δικτύων, πρόβλεψη διαφόρων υποθέσεων, ανάλυση δεδομένων, ακόμη και στην ιατρική κ.α.

1.2 Ορισμός του προβλήματος και σκοπός

Σε αυτή την πειραματική έρευνα το πρόβλημα που είχα να αντιμετωπίσω είναι η αποτελεσματική ανίχνευση 5 ειδών επιθέσεων σε ένα IoT με την χρήση δύο unsupervised learning αλγορίθμων, τον SOM και τον Isolation Forest. Τέλος, θέλουμε να πραγματοποιήσουμε ανίχνευση τόσο σε local detection μεθοδολογία (δηλαδή μεταξύ του sink node και του malicious node) όπως επίσης και σε network detection μεθοδολογία (δηλαδή σε όλο το δίκτυο).

Δοθέντος λοιπόν, δύο data sets (COOJA_DATA, FIT_DATA), σκοπός αυτής της έρευνας είναι η υλοποίηση και η ανάπτυξη μοντέλων τα οποία χρησιμοποιούν τον SOM και τον Isolation Forest με σκοπό να αναλύσουν και να προσπαθήσουν με ένα αποτελεσματικό τρόπο να εντοπίσουν τις ανωμαλίες σε αυτά τα δεδομένα, δηλαδή να εντοπίσουν τα κακόβουλα δεδομένα (επιθέσεις).

1.3 Σχετική δουλειά

Συστήματα ανίχνευσης επιθέσεων με την χρήση μηχανικής μάθησης, έχουν προταθεί εδώ και πολύ καιρό όπου το κάθε ένα έχει σκοπό την ανίχνευση διαφόρων και πολλαπλών επιθέσεων. Μια

δουλεία που προτάθηκε[1] χρησιμοποιεί τον Self-Organizing Map (SOM) αλγόριθμο για ανίχνευση γνωστών RPL επιθέσεων. Σε αυτή την έρευνα, οι κύριες συνεισφορές των συγγραφείς ήταν (α) η ικανότητα εντοπισμού διαφόρων τύπων RPL επιθέσεων με την χρήση unsupervised learning algorithm, (β) η ενίσχυση της κατανάλωσης ισχύος ειδοποιώντας τον διαχειριστή δικτύου σε πρώιμο στάδιο για μια συγκεκριμένη επίθεση, (γ) η εξασφάλιση διαθεσιμότητας δικτύου λόγω της άμεσης ειδοποίησης παραβίασης ασφάλειας.

Μια άλλη έρευνα [2] η οποία έγινε από το τμήμα του πανεπιστημίου Κύπρου με συγγραφείς την Δρ. Χριστιάνα Ιωάννου και τον Δρ. Βάσο Βασιλείου, στην οποία βασίστηκα και εγώ για να υλοποιήσω την δική μου έρευνα, είναι η χρήση supervised learning algorithm και συγκεκριμένα SVM (support vector machine) για ανίχνευση και εκτίμηση των Selective Forward και Blackhole network routing layer attacks χρησιμοποιώντας IoT-testbed δεδομένα και πετυχαίνοντας μέχρι και 99.8% accurate rates όπως επίσης και 100% Recall values. Σε αυτή την δουλεία, οι συγγραφείς χρησιμοποιούν τα ίδια datasets που χρησιμοποίησα εγώ για την δική μου έρευνα. Άρα η έρευνα μου είναι ουσιαστικά μια εναλλακτική δουλεία αυτής της έρευνας όπου εγώ χρησιμοποιώ διαφορετικούς αλγόριθμους μηχανικής μάθησης οι οποίοι σε αντίθεση με τον SVM, είναι unsupervised.

Τέλος, μια άλλη έρευνα που προτάθηκε [3], χρησιμοποιεί τον Isolation Forest για ανίχνευση ανωμαλιών σε fog networking. Τα fog devices σύμφωνα με το κείμενο είναι πολύ ευαίσθητα αφού μέσω αυτών ταξιδεύουν προσωπικές πληροφορίες όπως οικονομικές, πληροφορίες υγείας κ.α. Οι attackers στοχοποιούν αυτές τις συσκευές στέλνοντας κακόβουλα πακέτα δεδομένων. Χρησιμοποιούν το benchmark NSL_KDD dataset στο οποίο υλοποιούν το Isolation Forest μοντέλο τους φτάνοντας μέχρι και 95.5% accurate rate.

1.4 Μεθοδολογία

Σε αυτή την έρευνα, όπως ανάφερα και προηγουμένως αναπτύχθηκαν μοντέλα χρησιμοποιώντας δύο unsupervised learning algorithms, τον SOM και τον Isolation Forest. Αρχικά έχουμε δύο datasets, τα COOJA και τα FIT (τα οποία θα αναλύσουμε σε επόμενο section πως παράχθηκαν). Το FIT dataset αφορά δεδομένα που παράχθηκαν σε δύο τοπολογίες, sink in the middle και sink

in the top ενώ το COOJA dataset αφορά μια ακόμη τοπολογία, sink in random. Σε όλες τις τοπολογίες υλοποιήθηκαν μοντέλα χρησιμοποιώντας τους δύο αλγορίθμους, ξεχωριστά για κάθε είδος επίθεσης. Οι επιθέσεις που ανιχνεύονται και αναλύονται είναι sinkholes, blackholes και selective forward attacks. Η ανίχνευση έγινε τόσο σε local environment όσο και σε Network environment. Στο local environment, προσπαθούμε να δούμε την συμπεριφορά μεταξύ του sink node και του malicious node στην τοπολογία και πως αντιδρούν οι συγκεκριμένοι αλγόριθμοι σε αυτή την περίπτωση, ενώ στο network environment προσπαθούμε να αναλύσουμε την συμπεριφορά όλου του δικτύου, δηλαδή του sink node, του malicious node και των εναπομεινάντων nodes στο δίκτυο οι οποίοι είναι benign. Οι επιθέσεις blackhole και selective forward έχουν δύο παραλλαγές, την forwarding ratio (FR) και την block node (BN). Αρχικά παράχθηκαν τα COOJA και FIT datasets. Στη συνέχεια για κάθε τοπολογία στα δύο environments (local και network) χρησιμοποιούνται αυτά τα datasets χωρίζοντας τα δεδομένα σε μια δίκαια κατανομή train (80%) και test (20%) datasets. Τα training sets χρησιμοποιούνται για να εκπαιδεύσουμε τα μοντέλα μας (SOM & Isolation Forest) και τα test sets χρησιμοποιούνται για εκτίμηση και επαλήθευση των μοντέλων. Σημαντικό να αναφέρω ότι χρησιμοποιήθηκε hyperparameter tuning με την χρήση του cross validation για συντονισμό των παραμέτρων των δύο αλγορίθμων. Τέλος, παράγονται κάποια αποτελέσματα και με την χρήση της python και διαφόρων βιβλιοθηκών (matplotlib, seaborn, κ.α.) παράχθηκαν διάφορες γραφικές αλλά και διάφορα dataframes για να καταλάβουμε και να δικαιολογήσουμε τα αποτελέσματα που πήραμε. Όπως ανάφερα και στην αρχή του κειμένου, η μελέτη αυτή είναι πειραματική όπου σκοπό έχει την εξέταση και ανάλυση αυτών των δύο unsupervised αλγορίθμων και πόσο αποτελεσματικοί είναι στην επίλυση του προβλήματος, και όχι την υλοποίηση ενός τελικού IDS το οποίο θα χρησιμοποιηθεί σε real environment για ανίχνευση των επιθέσεων σε ένα IoT δίκτυο. Αυτό ίσως είναι μια εφαρμογή σε μελλοντικό Project.

Κεφάλαιο 2

Συλλογή δεδομένων

Σε αυτό το κεφάλαιο θα δούμε πως συλλέγουμε τα δεδομένα τα οποία χρησιμοποιούνται για training και evaluation των μοντέλων μας. Συλλέχθηκαν δύο datasets, COOJA & FIT. Επίσης θα δούμε την δομή των δεδομένων δηλαδή τα features τους και τέλος τις τοπολογίες που χρησιμοποιήθηκαν για κάθε dataset.

2.1 COOJA DATA & FIT DATA

Αρχικά να αναφέρω ότι τα δεδομένα συλλέχθηκαν από την Δρ. Χριστιάνα Ιωάννου και τον Δρ. Βάσο Βασιλείου[4].

Η συλλογή δεδομένων από κάθε κόμβο έγινε χρησιμοποιώντας το τηλεχειριστήριο εργαλείο παρακολούθησης (RMT) που περιγράφεται στο [5]. Το RMT είναι ένα Contiki εργαλείο, το οποίο παρέχει πληροφορίες τοπικού αισθητήρα για πολλαπλά επίπεδα της στοίβας πρωτοκόλλου περιοδικά και επίσης συλλέγει πληροφορίες σχετικά με τη δραστηριότητα της CPU και την

κατανάλωση ενέργειας. Το RMT εγκαταστάθηκε σε κάθε κόμβο για την παρακολούθηση των κόμβων σε διαστήματα 60 δευτερολέπτων.

Δημιουργήθηκαν δύο τύποι σεναρίων για τη συλλογή δεδομένων και αξιολόγηση τους μηχανισμούς ανίχνευσης. Στο στάδιο εκπαίδευσης COOJA, εκτέλεσαν 10 καλοήθεις (benign) προσομοιώσεις, καθεμία με διαφορετικό τυχαίο seed, για να επιτευχθεί ένα πιο αντιπροσωπευτικό δείγμα καλοήθους δραστηριότητας. Σε ένα καλοήθη σενάριο όλοι οι κόμβοι στο δίκτυο εκτελούν μια καλοήθη εφαρμογή. Σε κακόβουλο σενάριο, ένας κόμβος στο δίκτυο εκτελεί κακοήθης (malicious) εφαρμογή. Για κάθε τοπολογία δικτύου εκτέλεσαν 24 κακόβουλα σενάρια με τις επιθέσεις blackhole, sinkhole και selective forward.

Τα σενάρια αξιολογήθηκαν μέσα σε 15 λεπτά simulation/experimental time. Οι κόμβοι άρχισαν να μεταδίδουν δεδομένα μετά τα πρώτα 2 λεπτά simulation/experimental time. Ο χρόνος των δύο 2 λεπτών ρυθμίστηκε για να επιτρέπει στους κόμβους του αισθητήρα να είναι συνδεδεμένοι και να φτάσουν σε σταθερή κατάσταση. Το χρονικό διάστημα παρακολούθησης του RMT ορίστηκε στα 60 δευτερόλεπτα, συνεπώς τα δεδομένα που ανακτήθηκαν στα 15 λεπτά ήταν για 13 monitoring periods. Έτσι μόνο τα πρώτα 12 χρονικά διαστήματα παρακολούθησης χρησιμοποιήθηκαν.

Σε καλοήθεις και κακόβουλες εφαρμογές, ο ρυθμός δεδομένων ορίστηκε σε ένα πακέτο ανά δευτερόλεπτο (για έναν αποτελεσματικό ρυθμό δεδομένων από 384 bps). Κάθε κόμβος μεταδίδει 60 πακέτα ανά παρακολούθηση χρόνου και εάν ήταν ένας κόμβος relay, έκανε επίσης προώθηση των πακέτων που λαμβάνονταν από τους γειτονικούς κόμβους του.
Η συλλογή δεδομένων FIT DATA έγινε ακριβώς με τον ίδιο τρόπο με την μόνη διαφορά ότι χρησιμοποιήθηκαν μόνο τα πρώτα 13 χρονικά διαστήματα παρακολούθησης αντί 12 στα COOJA.

2.2 Data features

Για αυτήν την εργασία, χρησιμοποιήθηκαν δεδομένα από το επίπεδο δρομολόγησης που περιλαμβάνει data packets received, data packets sent, number of packets forwarded, number of packets dropped, και number of announcements received (Table 1).

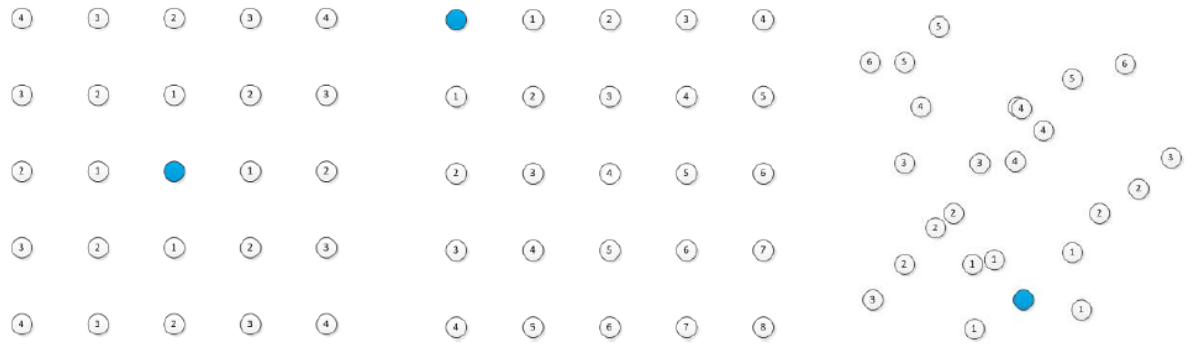
TABLE I
MONITORED ROUTING LAYER PARAMETERS

Network Layer Parameters	
Data Packets Received	Data Packets Sent
Packets Forwarded	Packets Dropped
Announcements Received	

Αυτά τα features χρησιμοποιήθηκαν για το train και το evaluate των μοντέλων. Τα features αναλόγως είναι λίγα σε σύγκριση με τα features που χρησιμοποιούνται συνήθως σε προβλήματα machine learning.

2.2 Τοπολογίες

Οι τοπολογίες που χρησιμοποιήθηκαν για τα COOJA datasets είναι οι εξής (Figure 1), Sink in the middle of the grid (Figure 1a), Sink at the top of the grid (Figure 1b), και όλα τα nodes τοποθετούνται τυχαία (Figure 1c).



(a) Sink node in the middle of the grid (S-Middle) (b) Sink node at top left corner of the grid (S-Top) (c) Random Placement of the Sink node (Random)

Fig. 1. Network Topologies Used for Training and Evaluation

Οι sink nodes φαίνονται στις τοπολογίες με μπλε χρώμα και τα άλλα nodes φαίνονται με ένα αριθμό ο οποίος είναι η απόσταση του κάθε node (number of hops) από τον sink node. Στο δίκτυο ακολουθούμε WSP (weighted shortest path), ένα απλό και ελαφρύ πρωτόκολλο επικοινωνίας για

να αφαιρέσουμε από το πείραμα τυχόν περιορισμούς και τυχόν άγνωστα ενδεχόμενα που δημιουργούνται από πιο περίπλοκα πρωτόκολλα δρομολόγησης όπως για παράδειγμα το RPL.

Κεφάλαιο 3

Επιθέσεις

Σε αυτό το κεφάλαιο θα δούμε τις επιθέσεις που είχα να ανιχνεύσω με τη χρήση των machine learning αλγορίθμων, τα χαρακτηριστικά των επιθέσεων και ποιος είναι ο σκοπός της κάθε επίθεσης. Να αναφέρω ότι όλες οι επιθέσεις υλοποιήθηκαν χρησιμοποιώντας το Contiki O/S.

3.1 Sinkholes

Η επίθεση Sinkhole είναι μια από τις πιο σοβαρές routing attacks. Ένας παραβιασμένος (compromised) κόμβος που διαπράττει μια επίθεση Sinkhole διαφημίζεται ως ο Sink και δελεάζει το traffic προς τον εαυτό του. Δηλαδή, ο attacker μπορεί στη συνέχεια είτε να παραβιάσει τη δρομολόγηση των πακέτων, να ‘κοροϊδέψει’ τα δεδομένα και τα μηνύματα δρομολόγησης, ή ακόμα και να μεταδίδει ψευδείς αναφορές επίθεσης, για περαιτέρω διαταραχή του δικτύου. Το Sinkhole attack μπορεί επίσης να προκαλέσει διαρροή ενέργειας στους γύρω κόμβους που οδηγούν σε ‘τρύπες’ ενέργειας στο δίκτυο και μπορεί να προκαλέσει ακατάλληλες και δυνητικά επικίνδυνες απαντήσεις που βασίζονται σε ψευδείς διαφημίσεις [6]. Αποδείχτηκε [7] ότι, ανάλογα με την τοπολογία του δικτύου, ακόμη και όταν ο κακόβουλος κόμβος βρίσκεται πιο μακριά από τον sink node, μπορεί ακόμα να αρνηθεί στον sink node τη δυνατότητα συλλογής πακέτων από μεγάλα τμήματα του δικτύου.

Η επίθεση πραγματοποιήθηκε έχοντας τον κακόβουλο κόμβο να ανακοινώνει ότι είναι ένας κόμβος με μηδέν απόσταση μακριά από το sink node (δηλαδή τον ίδιο τον sink). Όλα τα πακέτα που λαμβάνονται από τους άλλους κόμβους (‘κόμβοι θύματα’) γίνονται dropped από τον κακόβουλο κόμβο που διαπράττει την επίθεση.

3.2 Blackholes

Σε ένα Blackhole attack, οι κακοί κόμβοι διαφημίζουν τη λάθος διαδρομή προς το δίκτυο που έχει την πιο φρέσκια και συντομότερη διαδρομή προς τον προορισμό [8]. Η πρόθεση αυτών των κακών κόμβων είναι να προσελκύσουν περισσότερη κίνηση (traffic) και στη συνέχεια να αρχίσουν να ρίχνουν (drop) τα πακέτα αντί να τα προωθούν. Η επίθεση BH είναι βασικά μια επίθεση DOS για υποβάθμιση της απόδοσης του δικτύου. Κατά τη διάρκεια της επίθεσης BH, ο κόμβος μπλοκάρει/ ρίχνει τα εισερχόμενα δεδομένα αντί να τα στέλνει στον δέκτη. Η μαύρη τρύπα είναι η κακόβουλη συμπεριφορά ενός κόμβου που ισχυρίζεται ότι έχει τη συντομότερη διαδρομή προς τον προορισμό. Ο σκοπός μιας επίθεσης Blackhole είναι να ρίξει (drop) εισερχόμενα πακέτα δεδομένων.

Σε αυτή την έρευνα χρησιμοποιήθηκαν όπως και στην επόμενη επίθεση που θα δούμε, Selective Forward, δύο παραλλαγές, το FR (forwarding ratio) και το Block Node (BN). Το FR χρησιμοποιεί μια αναλογία (ratio) για να καθορίσει εάν θα προωθηθεί ή να γίνει drop ένα ληφθέν πακέτο. Το Block Node (BN) attack, ρίχνει (drops) εισερχόμενα πακέτα από τους γειτονικούς κόμβους με μια διαδοχική σειρά και μπλοκάρει κάθε κόμβο για ένα χρονικό διάστημα (για παράδειγμα στα 10 εισερχόμενα πακέτα).

3.3 Selective Forward

Αυτή η επίθεση είναι παρόμοια με την Blackhole. Όπως και το όνομα, αυτό το attack αποφασίζει ποια πακέτα να προωθήσει (forward) και ποια να κάνει drop [9]. Εάν τα πακέτα περιέχουν κρίσιμες ή ευαίσθητες πληροφορίες, αυτό θα έχει πολύ κακές συνέπειες, γι’ αυτό αυτή η επίθεση θεωρείται μια από τις ποιο κρίσιμες επιθέσεις στα IoT. Επιπλέον, η κινητικότητα των sensor nodes, η οποία απαιτείται σε ορισμένες εφαρμογές, όπως στην υγειονομική περίθαλψη και στα

ευφυή συστήματα μεταφοράς, οδηγεί σε μεγάλο αριθμό αστοχιών και συγκρούσεων σύνδεσης, γεγονός που αυξάνει τον αριθμό απώλειας πακέτων.

Όπως και στο blackhole attack, έτσι και στο selective forward attack, χρησιμοποιούνται οι δύο παραλλαγές, Forwarding ratio (FR) και Block Node (BN).

Κεφάλαιο 4

Αλγόριθμοι Μηχανικής Μάθησης

Σε αυτό το κεφάλαιο θα μιλήσουμε γενικά για τους αλγόριθμους μηχανικής μάθησης, ποια είναι η μεθοδολογία τους, στην συνέχεια θα δούμε την διαφορά των supervised learning algorithms με τους unsupervised learning algorithms, και τέλος θα δούμε τους αλγόριθμους που χρησιμοποίησα στην έρευνα αυτή.

4.1 Μηχανική Μάθηση

Γενικά η μηχανική μάθηση είναι η μελέτη των αλγορίθμων οι οποίοι βελτιώνονται αυτόματα με την εμπειρία και με την χρήση δεδομένων. Είναι ένα “hot” θέμα στις μέρες μας όπου η χρήση της γίνεται όλο και πιο ξακουστή καθώς χρησιμοποιείται σε πολλούς τομείς, όπως η ιατρική, στην

πρόβλεψη πραγμάτων, όπως πρόβλεψη καιρικών συνθηκών, και εννοείται στο θέμα που μας ενδιαφέρει, στην ανίχνευση επιθέσεων στα δίκτυα υπολογιστών και συγκεκριμένα στα IoT.

Η μεθοδολογία της μηχανικής μάθησης είναι σταθερή τις πλείστες φορές και αποτελείται κυρίως από 7 βήματα:

1. *Data Collection.*
2. *Data Preparation / preprocessing.*
3. *Choose a Model.*
4. *Train the Model.*
5. *Evaluate the Model.*
6. *Parameter Tuning.*
7. *Make Predictions.*

4.2 Supervised learning

Σε ένα supervised μοντέλο μάθησης, ο αλγόριθμος μαθαίνει σε ένα σύνολο δεδομένων με ετικέτα (labeled), παρέχοντας ένα κλειδί απάντησης που ο αλγόριθμος μπορεί να χρησιμοποιήσει για να αξιολογήσει την ακρίβειά του στα δεδομένα εκπαίδευσης. Με τον όρο labeled datasets εννοούμε ότι έχοντας ένα dataset με διάφορα features ή αλλιώς independent variables, έχουμε επίσης τα αποτελέσματα του κάθε data point δηλαδή τα depended variables. Ένα παράδειγμα είναι το εξής:

	packet forward	packet send	announcement received	packet dropped	results
0	0.000000	0.983607	0.00	0.0	0.0

όπου η ετικέτα είναι το result και τα features τα άλλα χαρακτηριστικά, δηλαδή ξέρουμε το αποτέλεσμα των δεδομένων.

Άρα το αποτέλεσμα ενός Supervised αλγορίθμου θα μας δώσει ένα prediction των δεδομένων που θέλουμε να βρούμε το results τους. Αρχικά χωρίζουμε τα δεδομένα σε train και test sets όπου τα train τα χρησιμοποιούμαι για να γίνει train ο αλγόριθμος έτσι ώστε όταν του δοθούν νέα δεδομένα χωρίς ετικέτα να μπορεί να “μαντέψει” αποτελεσματικά και με μεγάλο accurate score την ετικέτα για αυτά τα δεδομένα.

Μερικοί supervised αλγόριθμοι είναι ο Support-vector machines, Linear regression, Logistic regression, Naive Bayes, Linear discriminant analysis, Decision trees, K-nearest neighbor algorithm κ.α. Παρόμοια έρευνα προτάθηκε από την Δρ. Χριστιάνα Ιωάννου και τον Δρ. Βάσο Βασιλείου χρησιμοποιώντας SVM και BLR (binary logistic regression) [2].

4.3 Unsupervised learning

Η διαφορά των unsupervised learning αλγορίθμων με τους supervised είναι ότι τους χρησιμοποιούμε σε δεδομένα που είναι unlabeled, δηλαδή δεν έχουν ετικέτα και άρα δεν γνωρίζουμε τα αποτελέσματα. Σε αυτή την περίπτωση, συνήθως ο σκοπός αυτών των αλγορίθμων είναι να βάλουν τα δεδομένα σε ομάδες (clustering) ή σε κλάσεις (classification) και όχι να προβλέψουν το αποτέλεσμα ενός data point. Βέβαια μπορούμε να χρησιμοποιήσουμε unsupervised αλγορίθμους όταν έχουμε labeled δεδομένα, όπως δηλαδή και στην συγκεκριμένη έρευνα, αλλά είναι σημαντικό να γνωρίζουμε ότι σύμφωνα με μελέτες οι supervised αλγόριθμοι έχουν καλύτερα αποτελέσματα σε labeled δεδομένα απ’ ότι οι unsupervised.

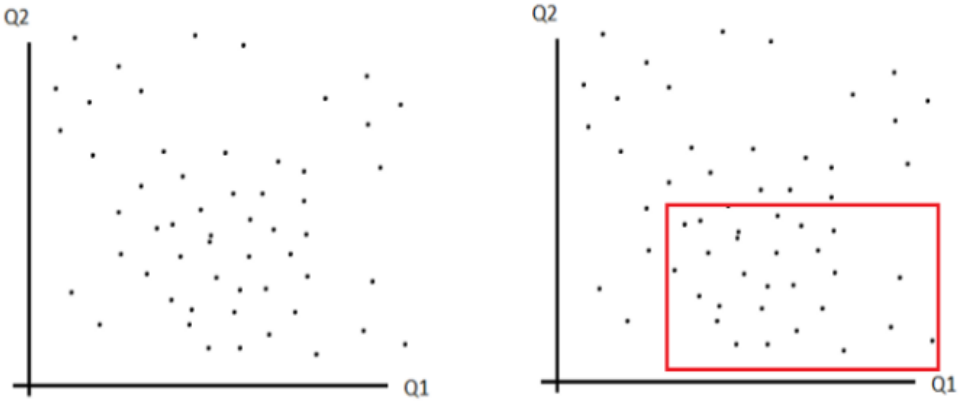
Μερικοί unsupervised αλγόριθμοι είναι ο K-means clustering, Hierarchical clustering, Neural Networks όπως επίσης και οι αλγόριθμοι που χρησιμοποίησα εγώ στην έρευνα αυτή, ο Isolation Forest και ο SOM.

4.4 Isolation Forest

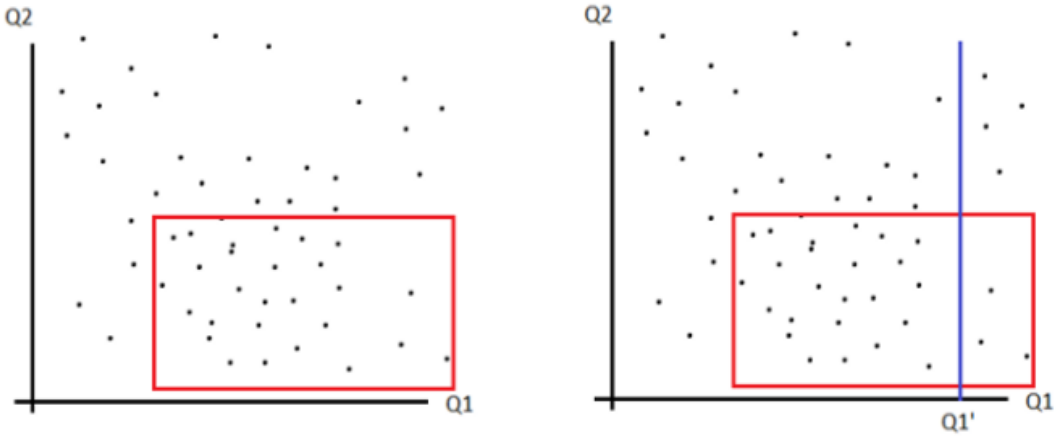
Ο αλγόριθμος Isolation Forest [10] είναι μια μη επιτηρούμενη (unsupervised) μέθοδος μηχανικής μάθησης ανίχνευσης ανωμαλιών, η οποία εντοπίζει ανωμαλίες διαχωρίζοντας τυχαία τα data points. Είναι μια καινοτόμος και προσωρινή μέθοδος [11] για την ανίχνευση ανωμαλιών, υποθέτοντας ότι τα δεδομένα που “πέφτουν” μακριά από το κέντρο δεδομένων είναι ανωμαλίες. Διαμορφώνεται σαν δυαδικά δέντρα και σύνολα iTrees με τυχαία δειγματοληψία για ένα σύνολο δεδομένων. Ο βασικός ρόλος του Isolation Forest είναι να κάνει χρήση ασυνήθιστων δειγμάτων (samples), που ονομάζονται επίσης ανωμαλίες, για την ανίχνευση άγνωστων επιθέσεων που είναι περίεργες και διαφορετικές από τις φυσιολογικές επιθέσεις.

Ο αλγόριθμος λειτουργεί ως εξής:

1.
    a. Το πρώτο βήμα είναι να πάρουμε μια δειγματοληψία (sampling) για την εκπαίδευση του μοντέλου.
    b. Αναλόγως του data set, το ποσοστό δειγματοληψίας μπορεί να είναι διαφορετικό (περισσότερα noisy data → μεγαλύτερη δειγματοληψία).



2. Binary decision tree
    a. Σε αυτό το βήμα δημιουργείται ένα δένδρο για το sample που πήραμε στο προηγούμενο βήμα.
    b. Αυτό το βήμα περιλαμβάνει δύο τυχαία στοιχεία: τυχαία επιλογή ενός attribute (π.χ. Q1 ή Q2) / τυχαία επιλογή τιμής είτε του Q1 είτε του Q2 μεταξύ min και max (π.χ. Q1')



3. Επανάληψη του βήματος 2 επαναληπτικά

a. Κάνε το βήμα 2 για δύο sub-data set βασιζόμενος στο binary split που έγινε στο βήμα 2.
b. “λιγότερα και διαφορετικά” δεδομένα (data points) απομονώνονται γρηγορότερα όπως τα data points στην πολύ κάτω δεξιά γωνία.
c. Πιο συγκεκριμένα, χρειάζεται μικρότερο μονοπάτι (path) για αυτά έτσι ώστε να απομονωθούν.
d. Κάνε αυτό αναδρομικά έτσι ώστε να δημιουργηθεί ένα δάσος, μια συλλογή από δένδρα.

4. Feeding data set και υπολόγισε το anomaly score.
    a. Κάνε feed το κάθε data point μέσα στο εκπαιδευμένο δάσος για κάθε δένδρο.
    b. Υπολογίζουμε το anomaly score για κάθε δένδρο όπως επίσης την τελική βαθμολογία ανωμαλίας για ολόκληρο το δάσος για ένα δεδομένο data point.
    c. Μαθηματικά, ένας outlier παίρνει score κοντά στο 1. Εμπειρικά, μελέτες δείχνουν ότι λιγότερα δένδρα σημαίνει μάλλον λιγότερος αριθμός των τιμών των outliers που είναι κοντά στο 1.

Ένα σημαντικό πλεονέκτημα του αλγορίθμου είναι ότι δουλεύει καλά σε μεγάλο αριθμό δεδομένων και σε δεδομένα με πολλές διαστάσεις, δηλαδή πολλά features. Στην περίπτωση μας τα features είναι μόνο 5.

Επίσης, είναι πολύ σημαντικό να γνωρίζουμε ότι ο συγκεκριμένος αλγόριθμος είναι πολύ αποτελεσματικός όταν στα δεδομένα υπάρχουν outliers. Δηλαδή ότι υπάρχουν δεδομένα τα οποία ξεχωρίζουν από τα άλλα, άρα όταν είναι απομονωμένα (isolated). Θα δούμε στην συνέχεια ότι στην συγκεκριμένη έρευνα υπάρχουν περιπτώσεις όπου τα δεδομένα τα οποία είναι μολυσμένα (malicious) “μοιάζουν” αρκετά με τα δεδομένα τα οποία είναι benign, άρα ο αλγόριθμος δεν θα είναι και τόσο αποτελεσματικός αφού θα θεωρεί αυτά τα κακόβουλα δεδομένα ως benign (false positives).

Ένα άλλο πλεονέκτημα του αλγορίθμου είναι ότι δεν χρησιμοποιεί μετρικές απόστασης (σε αντίθεση με τον SOM) και αυτό έχει ως αποτέλεσμα να είναι γρήγορος και να έχει χαμηλό υπολογιστικό κόστος. Αυτό όμως είναι ένα trade off, δηλαδή προτιμούμε να έχουμε ένα πολύ πιο γρήγορο αλγόριθμο αλλά με χαμηλότερη ακρίβεια ή προτιμούμε ένα αλγόριθμο που κοστίζει περισσότερο σε πόρους και χρόνο αλλά με μεγαλύτερη ακρίβεια. Το optimal που θα θέλαμε είναι να έχουμε ένα αλγόριθμο με χαμηλό κόστος και αποτελεσματικό με καλή ακρίβεια αλλά δυστυχώς δεν μπορούμε να έχουμε και τα δύο. Έτσι κατά την άποψή μου, λόγω της φύσης του προβλήματος όπου θέλουμε να ανιχνεύουμε αποτελεσματικά κακόβουλες δεδομένα, θα προτιμούσα ένα αλγόριθμο αποτελεσματικό με πολύ καλή ακρίβεια και accurate score ακόμη και αν κοστίζει περισσότερο σε χρόνο και υπολογιστικούς πόρους.

Κλείνοντας με τον αλγόριθμο Isolation Forest, είναι αναγκαίο να γνωρίζουμε τις παραμέτρους που παίρνει για να οριστεί έτσι ώστε να καταλάβουμε καλύτερα πως θα λειτουργήσει για το πρόβλημά μας.

Number of estimators: Ο αριθμός των estimators που θα χρησιμοποιηθούν. Δηλαδή ο αριθμός των δένδρων (trees) που θα κτιστούν στο δάσος. Είναι ένας ακέραιος αριθμός και είναι μια optional παράμετρος. Η προκαθορισμένη τιμή είναι 100.

Max samples: Ο αριθμός των δειγμάτων (samples) που θα παρθεί για να εκπαιδευτεί ο κάθε base estimator ή αλλιώς το κάθε δένδρο.

Contamination: Αυτή η παράμετρος είναι η πιο σημαντική και η πιο ευαίσθητη παράμετρος του αλγορίθμου. Αναφέρεται στο αναμενόμενο ποσοστό outliers που βρίσκεται στο data set. Αυτό χρησιμοποιείται κατά την τοποθέτηση (fitting), για τον καθορισμό του threshold στα scores των δειγμάτων. Η προκαθορισμένη τιμή είναι 0.1.

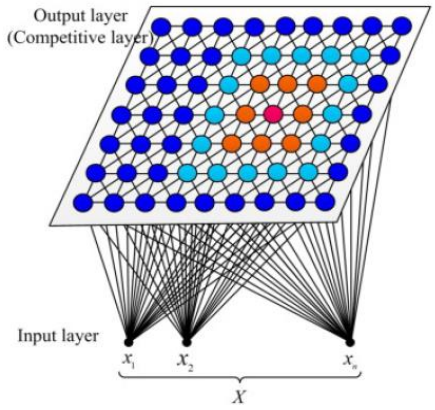
Max features: Όλοι οι base estimators ή δένδρα δεν εκπαιδεύονται με όλα τα features στο data set. Είναι ουσιαστικά, ο αριθμός των features που θα παρθεί από τα συνολικά features για την εκπαίδευση κάθε δένδρου. Η προκαθορισμένη τιμή είναι 1.

Όλες αυτές οι παράμετροι είναι σημαντικό να οριστούν σωστά για να έχουμε όσο καλύτερα αποτελέσματα μπορούμε. Για τον ορισμό και την απόφαση των τιμών της κάθε παραμέτρου θα μιλήσουμε σε μετέπειτα κεφάλαιο.

4.5 Self-Organizing Map

Ο SOM είναι ένας neural network αλγόριθμος ο οποίος προτάθηκε από τον εισήχθη από τον prof. Kohonen την δεκαετία του 1980. Είναι ένας αλγόριθμος neural network ο οποίος χρησιμοποιεί unsupervised learning για να μετατρέψει ένα υψηλής διάστασης χώρο σε ένα χαμηλής διάστασης χώρο ο οποίος μπορεί εύκολα να απεικονιστεί σε ένα χάρτη (map) [1].

Σε μια SOM τοπολογία, υπάρχουν 2 fully connected layers: το input layer και το output layer (Fig 2).



Σχήμα 4.1. SOM topology (architectural Graph)

Μια "γειτονική" σχέση ορίζεται στους νευρώνες εξόδου (output neurons). Ο κάθε νευρώνας έχει ένα βάρος το οποίο ορίζεται τυχαία. Επιλέγεται τυχαία ένα διάστημα εισόδου (input vector) το οποίο παρουσιάζεται στο πλέγμα. Κάθε νευρώνας υπολογίζει την απόστασή του (για παράδειγμα Euclidean distance) από το input vector. Ο νευρώνας ο οποίος είναι πιο κοντά στην είσοδο, θεωρείται ως Best Matching Unit (BMU) ή αλλιώς νευρώνας νικητής (winning neuron). Στην συνέχεια υπολογίζεται ένα radius γύρω από αυτό το winning neuron, έτσι ώστε όλα τα βάρη των

γειτονικών νευρώνων να προσαρμοστούν για να γίνουν περισσότερο σαν το input. Αυτά τα "προσαρμοσμένα" βάρη υπολογίζονται με την εξής εξίσωση όπου t είναι το time-step, L είναι το learning rate που μειώνεται με το πέρασμα του χρόνου.

$$W(t+1) = W(t) + \theta(t)L(t)(V(t) - W(t))$$

Το θ αντιπροσωπεύει το ποσοστό επιρροής που έχει η απόσταση ενός νευρώνα από το BMU στη φάση της εκπαίδευσης.

Παράμετροι αλγορίθμου:

Dimension X: Το X dimension στο grid.

Dimension Y: Το Y dimension στο grid. Να σημειώσουμε ότι X*Y μας δίνει τον αριθμό των clusters που θα παραχθούν.

Iterations: Ένας θετικός ακέραιος αριθμός που μας λέει πόσες φορές το data set θα δοθεί στο network ή αλλιώς στο map.

Sigma: Αυτή η παράμετρος χρησιμοποιείται για την κλιμάκωση (scale) της αλλαγής των SOM διανυσμάτων (vectors) όταν ένα νέο διάνυσμα σχετιστεί με έναν νευρώνα.

Learning rate: Είναι μια training παράμετρος η οποία ελέγχει το μέγεθος του weight vector δηλαδή του διανύσματος που έχει τα βάρη στην εκπαίδευση του SOM.

Όλες αυτές οι παράμετροι είναι σημαντικό να οριστούν σωστά για να έχουμε όσο καλύτερα αποτελέσματα μπορούμε. Για τον ορισμό και την απόφαση των τιμών της κάθε παραμέτρου θα μιλήσουμε σε μετέπειτα κεφάλαιο.

Θα ήθελα να τονίσω ότι ο αλγόριθμος SOM είναι ένας αλγόριθμος με αρκετά μεγάλη ακρίβεια (precision). Ο λόγος είναι επειδή χρησιμοποιεί τους νευρώνες και με μετρικές αποστάσεων όπως Euclidean distance έχει μια καθαρή εικόνα για τα input vectors αφού με αρκετά καλή ακρίβεια

γίνονται assigned στον ανάλογο νευρώνα. Άρα ουσιαστικά, ο SOM δίνει ένα καλό αποτέλεσμα clustering. Θα δούμε και στα αποτελέσματα μετέπειτα, ότι όντως σε γενικά ο SOM δίνει καλύτερα αποτελέσματα από τον Isolation Forest με περισσότερο accurate score, καλύτερο recall και καλύτερο precision. Όμως αυτή είναι η φύση της μηχανικής μάθησης. Δεν μπορούν όλοι οι αλγόριθμοι να είναι καλοί για όλων των ειδών προβλημάτων. Για κάθε πρόβλημα υπάρχει ένας ή ένα σύνολο αλγορίθμων που μπορούν να το επιλύουν αποτελεσματικά. Όπως ανάφερα και προηγούμενος πρόκειται για μια πειραματική έρευνα στην οποία βλέπουμε πως αυτοί οι δύο αλγόριθμοι αντιδρούν στο πρόβλημα που έχουμε να επιλύσουμε.

Κλείνοντας, αυτός ο αλγόριθμος κοστίζει πολύ περισσότερο από τον Isolation Forest τόσο σε πόρους (μνήμη) όσο και σε χρόνο. Ένας από τους λόγους είναι επειδή χρησιμοποιεί μετρικές απόστασης όπου υπολογίζει την απόσταση του κάθε input vector από όλους τους νευρώνες για να αποφασιστεί ποιος είναι ο κοντινότερος. Σε αντίθεση, ο Isolation Forest δεν χρησιμοποιεί μετρικές απόστασης γι’ αυτό κοστίζει πολύ λιγότερο. Άρα μιλάμε πάλι για ένα trade off, όπου καλούμαστε να επιλέξουμε μεταξύ της αποτελεσματικότητας και του κόστους.

Κεφάλαιο 5

Προετοιμασία δεδομένων (data preprocessing)

Σε αυτό το κεφάλαιο Θα αρχίσουμε να βλέπουμε την υλοποίηση της έρευνας. Θα Ξεκινήσουμε με την προετοιμασία των δεδομένων ή αλλιώς data preprocessing, ένα βασικό βήμα στην μηχανική μάθηση. Αυτή η διαδικασία αποτελείται από 5 κύρια στάδια. Σημαντικό να αναφέρω ότι δεν είναι απαραίτητο να υλοποιηθούν όλα τα στάδια. Αναλόγως των δεδομένων και του προβλήματος κάποια στάδια δεν είναι απαραίτητο να υλοποιηθούν. Τέλος θα δούμε διάφορες τεχνικές που χρησιμοποιήθηκαν και θα ήθελα να σημειώσω ότι θα δούμε τις τεχνικές εφαρμοσμένες σε ένα είδος επίθεσης και συγκεκριμένα στο blackhole BlockNode. Οι ίδιες ακριβώς τεχνικές και οι ίδιες ακριβώς αποφάσεις αναλόγως των αποτελεσμάτων των τεχνικών πάρθηκαν για όλες τις επιθέσεις.

5.1 Δημιουργία pandas πινάκων

Αρχικά τα δεδομένα μας υπάρχουν σε txt files. Όπως ανάφερα και στο κεφάλαιο με την δημιουργία των δεδομένων υπάρχουν τα FIT & COOJA datasets. Το FIT dataset αποτελείται από δύο τοπολογίες. Την τοπολογία όπου ο sink είναι στο middle του grid και την τοπολογία όπου είναι στο top. Επίσης έχουμε δύο ειδών ανίχνευσης. Η ανίχνευση που γίνεται σε local environment και η ανίχνευση που γίνεται σε Network environment. Για να καταλάβουμε καλύτερα δίνεται ο παρακάτω πίνακας των txt files για το middle topology για local & network detection στο FIT dataset.

MIDDLE TOPOLOGY	Benign	Blackhole SF	Blackhole BN	Selective Forward FR	Selective Forward BN	Sinkhole
Local Detection	benign_middle.txt	SF_BH_FR_middle_malicious.txt	SF_BH_BN_middle_malicious.txt	SF_FR_middle_malicious.txt	SF_BN_middle_malicious.txt	Sinkhole_middle_malicious.txt
Network Detection	benign_middle.txt	SF_BH_FR_middle_malNeighbors.txt SF_BH_FR_middle_malicious.txt	SF_BH_BN_middle_malNeighbors.txt SF_BH_BN_middle_malicious.txt	SF_FR_middle_malNeighbors.txt SF_FR_middle_malicious.txt	SF_BN_middle_malNeighbors.txt SF_BN_middle_malicious.txt	Sinkhole_middle_malNeighbors.txt Sinkhole_middle_malicious.txt

Όπως βλέπουμε από τον πιο πάνω πίνακα έχουμε το local & network detection για το middle topology στο FIT dataset. Στο local detection παρατηρούμε ότι υπάρχουν τα δεδομένα του sink (benign_middle.txt) και κάθε επίθεση έχει ένα txt file του αντίστοιχου malicious node. Αυτό επειδή στο local detection κάνουμε ανίχνευση και ανάλυση μεταξύ του sink node και του malicious node. Στο network detection έχουμε και τα δεδομένα των άλλων κόμβων στο δίκτυο για παράδειγμα στο blackhole SF το SF_BH_FR_middle_malNeighbors.txt όπου είναι οι εναπομείναντες κόμβοι οι οποίοι είναι benign.

Αν ανοίξουμε ένα txt file θα παρατηρήσουμε 6 στήλες.

1. *Το αποτέλεσμα του data point (0 → benign, 1 → malicious)*
2. *Packets forward*
3. *Packets send*
4. *Packets received*
5. *Announcements received*
6. *Packets dropped*

Αφού διαβάσουμε τα δεδομένα δημιουργούμε με την βοήθεια της βιβλιοθήκης pandas ένα data frame για τα δεδομένα για να έχουμε μια καλύτερη οπτική εικόνα και οργανωμένα δεδομένα(Σχήμα 5).

	results	packet forward	packet send	packet received	announcement received	packet dropped
0	0	3	59	0	1	0
1	0	0	59	0	0	0
2	0	3	59	0	1	0
3	0	3	59	0	0	0
4	0	3	60	0	1	0
...	...	...	...	...	...	...
10587	0	3	60	0	0	0
10588	0	0	60	0	0	0
10589	0	3	60	0	0	0
10590	0	2	60	0	1	0
10591	0	3	60	0	0	0

Σχήμα 5.1 data frame

5.2 Καθαρισμός Δεδομένων

Το πρώτο στάδιο της προετοιμασίας των δεδομένων είναι ο καθαρισμός των δεδομένων. Όταν λέμε καθαρισμός εννοούμε να ελέγξουμε αν υπάρχουν ελλιπής τιμές (missing values), τιμές οι οποίες για κάποιο λόγο είναι περίεργες και να μεταχειριστούμε τις μηδενικές τιμές αν αυτό είναι απαραίτητο.

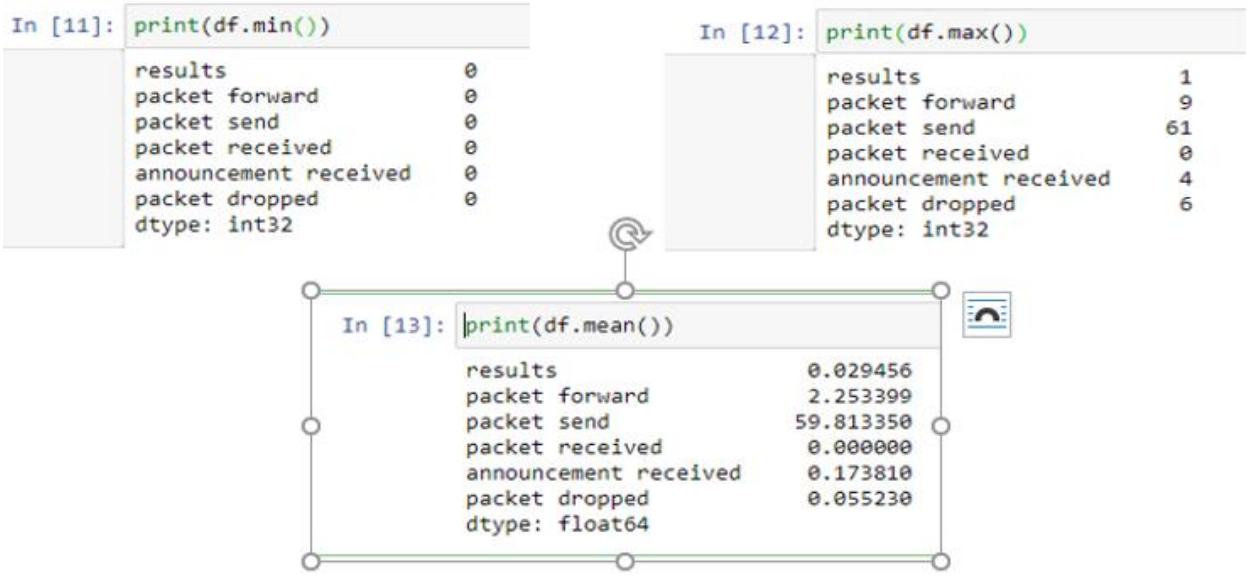
Ελλιπής τιμές (missing values): Για να ελέγξουμε αν υπάρχουν ελλιπής τιμές μπορούμε να χρησιμοποιήσουμε την βιβλιοθήκη pandas (Fig 4).

```
In [6]: print(df.isnull().sum())

results                 0
packet forward          0
packet send             0
packet received         0
announcement received   0
packet dropped          0
dtype: int64
```

Σχήμα 5.2. Missing values

Όπως βλέπουμε με τον πιο πάνω κώδικα ελέγχουμε αν υπάρχουν ελλιπής τιμές δηλαδή null σε οποιοδήποτε από τα features παίρνοντας το άθροισμα των ελλιπών τιμών. Στο πιο πάνω παράδειγμα δεν έχουμε ελλιπής τιμές αφού το άθροισμά τους για όλα τα features είναι μηδέν. Έγινε έλεγχος για όλες τις υλοποιήσεις και δεν υπάρχουν ελλιπής πουθενά ελλιπής τιμές.

Περίεργες τιμές: Υπάρχουν περιπτώσεις όπου μπορεί να υπάρχουν τιμές στα δεδομένα που να διαφέρουν κατά μεγάλο βαθμό από τις άλλες τιμές του κάθε feature. Αυτό μπορεί να συμβεί όταν προκύψει καταλάθος κάτι κατά την εξαγωγή των δεδομένων. Ένας τρόπος να το ελέγξουμε αυτό είναι με το να δούμε τα min values, max values & τον μέσο όρο των τιμών για κάθε feature(Fig 5).



Σχήμα 5.3

Από το Σχήμα 5.3 παρατηρούμε ότι για αυτά τα δεδομένα (FIT DATA, network Detection, middle topology, blackhole BN attack) δεν υπάρχουν περίεργες τιμές αφού οι ελάχιστες και μέγιστες τιμές, "συμφωνούν" με τον μέσο όρο.

Μηδενικές τιμές: Στο Σχήμα 5.4 φαίνεται ότι στα δεδομένα μας έχουμε πολλές μηδενικές τιμές (εξαίρουμε εννοείται τις τιμές για το result). Για παράδειγμα το feature packet received αποτελείται μόνο από μηδενικές τιμές. Θα δούμε στην συνέχεια πως αντιμετωπίζουμε αυτό το feature. Σε άλλα features πάλι υπάρχουν μηδενικές τιμές και πιο συγκεκριμένα μπορούμε να τα δούμε στο Σχήμα 5.4.

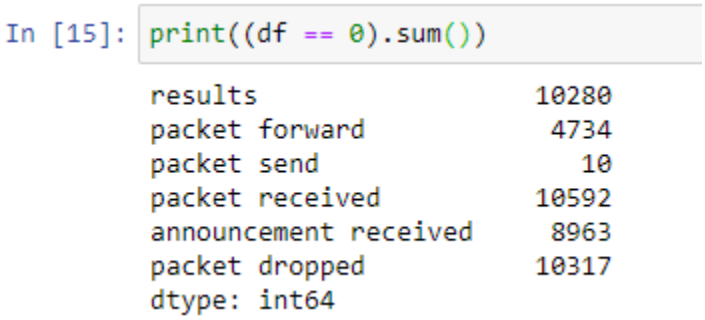


Fig 5.4. Sum of zero values

Όπως βλέπουμε ο αριθμός των μηδενικών τιμών σε κάθε feature (εκτός του results που εκεί είναι οι τιμές για τα benign data points), είναι πολύ μεγάλος. Αυτό σημαίνει ότι οι μηδενικές τιμές των features παίζουν κάποιο σημαντικό ρόλο στην διακύμανση των features και στην επιρροή τους στο τελικό αποτέλεσμα (results). Άρα αυτές οι τιμές δεν πρέπει να μετασχηματιστούν.

5.3 Κλιμάκωση Δεδομένων (scaling)

Η κλιμάκωση δεδομένων είναι ένα σημαντικό στάδιο στην προετοιμασία δεδομένων. Αυτό το στάδιο συμβαίνει όταν έχουμε μεγάλη διαφορά στις τιμές των features. Για παράδειγμα αν οι τιμές ενός feature κυμαίνονται μεταξύ 40-70 και οι τιμές σε ένα άλλο feature κυμαίνονται μεταξύ 0-7, τότε τα δεδομένα πρέπει να κλιμακωθούν έτσι ώστε να μειώσουμε την απόκλιση αυτή. Ειδικά στον αλγόριθμο SOM ο οποίος χρησιμοποιεί distance metrics (π.χ. Euclidean distance) το feature scaling είναι απαραίτητο για να έχουμε μια δίκαιη συνεισφορά στον υπολογισμό της απόστασης από όλα τα features.

```
array([[ 6, 60,  0,  0],      array([[0.66666667, 0.98360656, 0.        , 0.        ],
       [ 3, 60,  0,  0],             [0.33333333, 0.98360656, 0.        , 0.        ],
       [ 0, 60,  0,  0],             [0.        , 0.98360656, 0.        , 0.        ],
       ...,                          ...,
       [ 0, 60,  0,  0],             [0.        , 0.98360656, 0.        , 0.        ],
       [ 5, 60,  1,  0],             [0.55555556, 0.98360656, 0.25      , 0.        ],
       [ 3, 60,  0,  0]])            [0.33333333, 0.98360656, 0.        , 0.        ]])
```

Fig 5.5 Feature scaling

Στο Σχήμα 5.5 βλέπουμε αριστερά τις τιμές των features πριν την κλιμάκωση και δεξιά τις τιμές μετά την κλιμάκωση. Η κλιμάκωση γίνεται με την βοήθεια της βιβλιοθήκης sklearn.preprocessing και συγκεκριμένα του module MinMaxScaler όπου οι τιμές γίνονται normalize μεταξύ του διαστήματος [0,1] (Σχήμα 5.6).

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Fig 5.6. MinMaxScaler

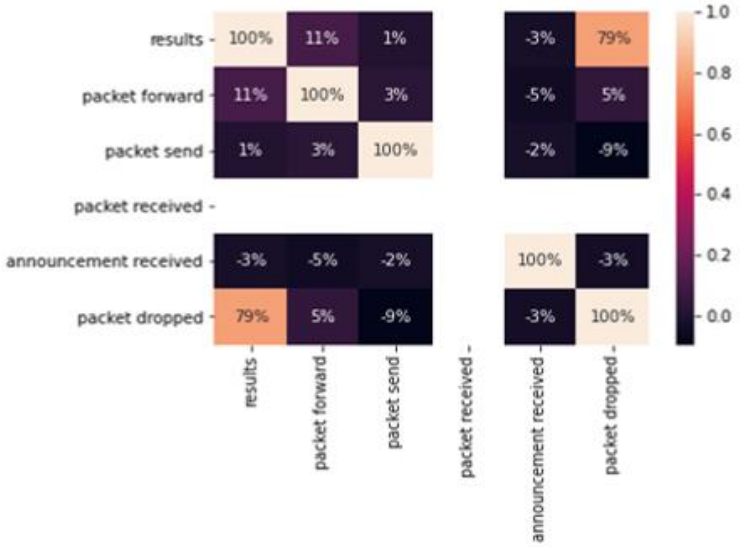
5.4 Dimensionality Reduction & Feature Selection

Το dimensionality reduction είναι η διαδικασία στην οποία μειώνουμε τις διαστάσεις των δεδομένων (features) χωρίς να αλλάξουμε την αποτελεσματικότητα και την ακρίβεια του προβλήματος έτσι ώστε να έχουμε πολύ καλύτερο performance. Για παράδειγμα αν σε ένα πρόβλημα έχουμε 70 features τότε αν τα κρατήσουμε όλα θα έχουμε ένα σχετικά κακό performance του αλγορίθμου. Είναι ένα πολύ σημαντικό στάδιο στην προετοιμασία των δεδομένων. Υπάρχουν διάφοροι τρόποι για να πετύχουμε το dimensionality reduction όπως το High Correlation Filter, PCA, SVD, Backward Feature Elimination, Forward Feature Elimination, Ensemble Trees κ.α.

Σε αυτή την έρευνα έχουμε να κάνουμε μόνο με 5 features. Άρα είμαστε πολύ περιορισμένοι στα features των δεδομένων το πιο πιθανό δεν χρειάζεται ή καλύτερα δεν πρέπει να εφαρμόσουμε dimensionality reduction.

Όμως για να είμαι εντελώς σίγουρος ακολούθησα 2 τεχνικές dimensionality reduction για να αποφασίσω κατά πόσο να μειώσω τα features. Οι τεχνικές που ακολούθησα είναι το High Correlation Filter και τα Ensemble Trees.

```
df = pd.DataFrame(np.concatenate((benign,malicious,network),axis=0), columns = ['results','packet forward','packet send',
                                                                                 'packet received','announcement received',
                                                                                'packet dropped'])
sns.heatmap(df.corr(),annot = True, fmt= ".0%")
```



Σχήμα 5.7. Correlation Matrix

Στο Fig 5.7 βλέπουμε την τεχνική correlation στην οποία βλέπουμε την επιρροή του κάθε feature και την συσχέτισή του με τα άλλα features αλλά και με το result. Από αυτό το heat map μπορούμε

```
from sklearn.ensemble import ExtraTreesClassifier
estimator = ExtraTreesClassifier(n_estimators=100, max_features=5, random_state=0)
estimator.fit(X, y)
importances = estimator.feature_importances_
importances
```

```
array([0.02906064, 0.01467845, 0.        , 0.00204962, 0.9542113 ])
```

Σχήμα 5.8 Feature Importance

να διαπιστώσουμε ότι το feature packet received δεν έχει καθόλου correlation με κανένα feature όπως επίσης και με το result.

Όπως βλέπουμε στο Fig 5.8 χρησιμοποιώ την τεχνική ExtraTreeClassifier ή αλλιώς Ensemble Tree και βλέπουμε ότι όντως το feature packet received έχει 0% επιρροή στο αποτέλεσμα. Επίσης φαίνεται ότι μόνο το τελευταίο feature (packet dropped) έχει μεγάλο importance στο αποτέλεσμα και αυτό είναι πολύ λογικό γιατί το παράδειγμα που αναλύω τώρα είναι η επίθεση blackhole BN η οποία όπως ανάφερα στο κεφάλαιο 3 είναι η επίθεση κατά την οποία ο malicious node κάνει

drop τα πακέτα αντί να τα κάνει forward. Αυτός είναι ο λόγος που υπάρχει μεγάλο importance (95,4%) στο feature packets dropped. Όμως επαναλαμβάνω λόγω των περιορισμένων features που έχουμε δεν μπορούμε να βασιστούμε μόνο σε ένα feature και να αγνοήσουμε τα άλλα 4. Έτσι θα μελετήσουμε μόνο το ενδεχόμενο του packet received το οποίο έχει μηδενικό importance.

Από αυτές τις τεχνικές (Σχήμα 5.7 & Σχήμα 5.8) καταλήγουμε στο συμπέρασμα ότι ίσως είναι καλό να διαγράψουμε το feature packet received σε αυτό το είδος επίθεσης.

Τα αποτελέσματα που πήρα στο τέλος διαγράφοντας αυτό το feature ήταν σχεδόν τα ίδια με τα αποτελέσματα χωρίς να διαγράψω αυτό το feature. Η διαφορά ήταν πολύ μικρή και αυτό με οδήγησε στο να διαγράψω αυτό το feature.

5.5 Split to train & test sets

Μετά την ολοκλήρωση των πιο πάνω σταδίων, φτάνουμε στο στάδιο όπου πρέπει να χωρίσουμε τα δεδομένα μας σε train και tests sets. Αυτό το στάδιο είναι απαραίτητο και πρέπει να γίνεται πάντα.

Αρχικά ας εξηγήσουμε λίγο τι εννοούμε με τα train & test sets. Τα train sets είναι τα δεδομένα τα οποία θα δώσουμε στο μοντέλο μας για να το εκπαιδεύσουμε, να "μάθει". Τα test sets είναι τα δεδομένα τα οποία θα δώσουμε στο εκπαιδευμένο μοντέλο για να κάνουμε evaluate το μοντέλο. Δηλαδή για να δούμε τα αποτελέσματα του εκπαιδευμένου μοντέλου τα οποία αποτελούνται από το accurate score, το classification report και το confusion matrix (Κεφάλαιο 6).
Train set: Το train set χωρίζεται σε X_train και Y_train. Το X_train είναι τα δεδομένα που περιέχουν τα features ή αλλιώς independent variables και για τα αντίστοιχα αυτά X_train, τα Y_train είναι τα αποτελέσματα για αυτά τα features δηλαδή τα depended variables. Αυτά τα δύο sets θα δοθούν στο μοντέλο για να εκπαιδευτεί.

Test set: Το test set χωρίζεται σε X_test και Y_test. Το X_test είναι το set το οποίο θα δοθεί στο εκπαιδευμένο μοντέλο για να προβλέψει τα αποτελέσματα αυτού του set. Δηλαδή να βάλει ετικέτα σε αυτά τα αποτελέσματα (0 αν είναι benign, 1 αν είναι malicious). Το Y_test είναι τα actual

αποτελέσματα του X_test. Άρα όταν το μοντέλο προβλέψει τα αποτελέσματα για το X_test, εμείς θα συγκρίνουμε τα αποτελέσματα που πρόβλεψε το μοντέλο με τα πραγματικά αποτελέσματα του X_test που είναι το Y_test. Έτσι θα βγάλουμε τις μετρικές μας που θα δούμε ποιες είναι αυτές στο επόμενο κεφάλαιο.

Χώρισα τα δεδομένα σε 80% train και 20% test. Για να έχουμε όμως ένα δίκαιο split μεταξύ των txt files δηλαδή ένα δίκαιο split από τα δεδομένα του sink, τα δεδομένα του malicious node και τα δεδομένα των neighbors (στο network detection δηλαδή), έκανα split του κάθε txt file σε 80% train και 20% test και μετά έκανα concatenate τα files για να έχω τα ολοκληρωμένα X_train, Y_train, X_test, και Y_test. Η διαδικασία φαίνεται στο Σχήμα 5.9.

```
#Take the 80% of benign and malicious txt and concatenate them. Then take 20% of each again and concatenate
from sklearn.model_selection import train_test_split
benign = np.loadtxt("data/benign_middle.txt", dtype='int')
malicious = np.loadtxt("data/BlackHole/SF_BH_BN_middle/SF_BH_BN_middle_malicious.txt", dtype='int')
network = np.loadtxt("data/BlackHole/SF_BH_BN_middle/SF_BH_BN_middle_malNeighbors.txt", dtype='int')

X_benign = benign[:, 1:]
Y_benign = benign[:, 0]
X_malicious = malicious[:, 1:]
Y_malicious = malicious[:, 0]
X_network = network[:, 1:]
Y_network = network[:, 0]

X = np.vstack((X_benign, X_malicious, X_network))
y = np.hstack((Y_benign, Y_malicious, Y_network))

X_trainB, X_testB, Y_trainB, Y_testB = train_test_split(X_benign, Y_benign, test_size=0.2,random_state=0)
X_trainM, X_testM, Y_trainM, Y_testM = train_test_split(X_malicious, Y_malicious, test_size=0.2,random_state=0)
X_trainN, X_testN, Y_trainN, Y_testN = train_test_split(X_network, Y_network, test_size=0.2,random_state=0)

X_train = np.concatenate((X_trainB,X_trainM,X_trainN),axis=0)
X_test = np.concatenate((X_testB,X_testM,X_testN), axis=0)
Y_train = np.concatenate((Y_trainB,Y_trainM,Y_trainN), axis=0)
Y_test= np.concatenate((Y_testB,Y_testM,Y_testN), axis=0)
```

Σχήμα 5.9. Train & Test set split

5.6 Hyperparameter tuning

Μετά την ολοκλήρωση του σταδίου όπου κάνουμε split τα δεδομένα μας σε train & test, μένει ένα τελευταίο στάδιο που είναι αναγκαίο και πάρα πολύ σημαντικό. Αυτό το στάδιο ονομάζεται hyperparameter tuning.

Όλοι οι αλγόριθμοι μηχανικής μάθησης απαιτούν κάποιες παραμέτρους για να οριστούν. Έτσι και ο SOM και ο Isolation Forest που χρησιμοποίησα σε αυτή την έρευνα. Οι παράμετροι αυτών των αλγορίθμων αναφέρθηκαν και περιεγράφηκαν στο Κεφάλαιο 4.

Το hyperparameter tuning είναι η διαδικασία στην οποία συντονίζεις και βρίσκεις τις καλύτερες τιμές για τις παραμέτρους, αυτές δηλαδή που "ταιριάζουν" περισσότερο για την επίλυση κάποιου προβλήματος με τον αντίστοιχο αλγόριθμο μηχανικής μάθησης. Μπορεί δίνοντας τυχαίες τιμές στις παραμέτρους να πετύχουμε ένα πολύ καλό accurate score όμως αυτό δεν σημαίνει τίποτα. Αυτό μπορεί να συμβεί επειδή ίσως να έτυχε για τα συγκεκριμένα training και test sets ο αλγόριθμος να δίνει πολύ καλά αποτελέσματα. Τι συμβαίνει όμως σε διαφορετικό διαχωρισμό των δεδομένων σε train και split;

Για να έχουμε ένα σωστό μοντέλο, θα πρέπει το μοντέλο αυτό να δίνει τα καλύτερα αποτελέσματα που μπορεί να δώσει για ένα πρόβλημα χρησιμοποιώντας αρκετά training & test sets. Έτσι η διαδικασία του hyperparameter tuning είναι αναγκαία και πρέπει να γίνεται σε κάθε πρόβλημα μηχανικής μάθησης.

Το hyperparameter tuning μπορεί να γίνει με διάφορες τεχνικές. Η πιο γνωστή και η πιο διαδεδομένη με πολύ καλά αποτελέσματα είναι το cross Validation. Το cross validation είναι μια απλή τεχνική η οποία δουλεύει ως εξής:

i. Όρισε το εύρος των τιμών για κάθε παράμετρο που θέλεις να ελέγξεις
ii. Όρισε πόσα splits των δεδομένων θέλεις ο αλγόριθμος να κάνει
iii. Με την χρήση του GridSearchCV βρες τις καλύτερες τιμές των παραμέτρων του μοντέλου

Στο Σχήμα 5.10 βλέπουμε το hyperparameter tuning που έκανα για τον αλγόριθμο Isolation Forest για την επίθεση blackhole BN στα FIT_DATA, σε network detection, με τοπολογία sink in the middle. Ακριβώς η ίδια διαδικασία χρησιμοποιείται για όλες τις άλλες επιθέσεις σε όλες τις τοπολογίες.

```
#hyperparameter tuning
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import make_scorer, f1_score

Y_train = np.where(Y_train==1,-1,Y_train)
Y_train = np.where(Y_train==0,1,Y_train)
Y_test = np.where(Y_test==1,-1,Y_test)
Y_test = np.where(Y_test==0,1,Y_test)

f1sc = make_scorer(f1_score, average='micro')


param_grid = {'contamination': [0.01, 0.02, 0.03, 0.04, 0.05],
              'max_samples': [100,150,200,250],
              'n_estimators': [100,150,200,250],
              'max_features': [1,2,3]
             }

iforest = IsolationForest(random_state=0)
clf = GridSearchCV(estimator=iforest, param_grid=param_grid, scoring=f1sc, cv=5,
                   return_train_score=False)

clf.fit(X_train, Y_train)
```

Σχήμα 5.10. Hyperparameter tuning

Αρχικά ορίζουμε τις τιμές στις παραμέτρους του Isolation Forest που θέλουμε να ελέγξουμε. Για παράδειγμα οι τιμές της παραμέτρου contamination που θέλω να ελέγξω είναι [0.01, 0.02, … 0.05]. Αυτές οι τιμές αποφασίζονται μετά από αρκετή μελέτη.

Αφού ορίσουμε τις τιμές των παραμέτρων, με την χρήση της κλάσης GridSearchCV επιτυγχάνουμε το cross validation. Το cv = 5 είναι τα 5 διαφορετικά splits που θα χρησιμοποιήσει ο αλγόριθμος. Για κάθε συνδυασμό παραμέτρων ο αλγόριθμος GridSearchCV παίρνει τον μέσο όρο των accurate score που πετυχαίνει ο αλγόριθμος Isolation Forest για τα 5 διαφορετικά splits. Όταν τελειώσει έχουμε τις καλύτερες τιμές των παραμέτρων δηλαδή τον καλύτερο μέσο όρο accurate score που βρίσκει ο αλγόριθμος. Έτσι, παίρνουμε αυτές τις παραμέτρους και τις χρησιμοποιούμε μετά για να ορίσουμε τον Isolation Forest και να ξεκινήσουμε το train και evaluation του αλγορίθμου.

```
df_param[['param_contamination','param_max_features', 'param_max_samples', 'param_n_estimators', 'mean_test_score']]
```

	param_contamination	param_max_features	param_max_samples	param_n_estimators	mean_test_score
0	0.01	1	100	100	0.968967
1	0.01	1	100	200	0.968967
2	0.01	1	100	300	0.968967
3	0.01	1	100	400	0.969203
4	0.01	1	100	500	0.968967
...	...	...	...	...	...
355	0.05	3	250	200	0.949964
356	0.05	3	250	300	0.949491
357	0.05	3	250	400	0.948311
358	0.05	3	250	500	0.947602
359	0.05	3	250	600	0.948901

360 rows × 5 columns

Σχήμα 5.11. Tuning results

Στο Σχήμα 5.11 βλέπουμε τον συνδυασμό όλων των τιμών των παραμέτρων, 360 συνδυασμοί και τον μέσο όρο του accurate score για όλα τα splits.

```
clf.best_score_
```

```
0.9765191043871656
```

```
clf.best_params_
```

```
{'contamination': 0.02,
 'max_features': 1,
 'max_samples': 250,
 'n_estimators': 100}
```

Σχήμα 5.12. Best parameters

Στο Σχήμα 5.12 με την χρήση του best_score και best_params παίρνουμε το καλύτερο score το οποίο είναι 97% και τις παραμέτρους για αυτό το score οι οποίες όπως βλέπουμε για το συγκεκριμένο παράδειγμα είναι contamination →0.02, max_feature → 1, max_samples → 250, kai n_estimators → 100. Τέλος αυτές τις παραμέτρους θα χρησιμοποιήσουμε για να ορίσουμε το μοντέλο μας στη συνέχεια.

Κεφάλαιο 6

Μετρικές επίδοσης

Σε αυτό το κεφάλαιο θα δούμε με ποιες τεχνικές μετρούμε την αποδοτικότητα και την αποτελεσματικότητα του κάθε μοντέλου για κάθε περίπτωση.

Ακολουθούνται 3 συγκεκριμένες μετρικές. Αυτές είναι (α) classification Report, (β) Confusion Matrix και (γ) Accurate Score. Αυτές οι 3 τεχνικές υλοποιούνται με την χρήση της python και συγκεκριμένα της βιβλιοθήκης sklearn.metrics.

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm = confusion_matrix(Y_test,y_pred)
ac = accuracy_score(Y_test,y_pred)
cr = classification_report(Y_test, y_pred)
print("classification report\n", cr)
print("confusion matrix\n", cm)
print("\naccuracy score\n", ac)
```

Σχήμα 6.1. Metrics

Όπως βλέπουμε στο Σχήμα 6.1, καλούνται οι μέθοδοι confusion_matrix, accurate_score και classification_report από τις κλάσεις confusion_matrix, accurate_score και classification_report αντίστοιχα.
Και στις 3 μεθόδους δίνουμε σαν παραμέτρους το Y_test και το y_pred. Όπως ανάφερα και στο προηγούμενο κεφάλαιο το Y_test είναι τα πραγματικά (real) αποτελέσματα του X_test και τα y_pred είναι τα αποτελέσματα (predictions) μου μας δίνει το μοντέλο (Isolation Forest & SOM). Άρα ουσιαστικά συγκρίνουμε τα πραγματικά αποτελέσματα με τα αποτελέσματα που μας δίνει το κάθε 1 από τα 2 μοντέλα. Όσο πιο όμοια είναι τόσο το καλύτερο.

6.1 Classification Report

Το classification Report είναι μια τεχνική μέτρησης μιας γενικής απόδοσης του αλγορίθμου. Αυτή η τεχνική μας δίνει 4 αποτελέσματα. Το precision, Recall, f1-score και το support.

Precision: Η ακρίβεια (precision) είναι η ικανότητα ενός classifier να μην δείχνει μια true παρουσία η οποία είναι πραγματικά false. Για κάθε class ορίζεται ως ο λόγος των αληθινών θετικών προς το άθροισμα των αληθών και των ψευδών θετικών.
TP – True Positives
FP – False Positives
Precision – Accuracy of positive predictions.
Precision = TP/(TP + FP) όπου TP →true positives & FP → false positives

Recall: Το Recall είναι η ικανότητα του classifier να βρίσκει όλα τα true instances. Για κάθε τάξη ορίζεται ως ο λόγος των πραγματικών θετικών προς το άθροισμα των πραγματικών θετικών και των ψευδών αρνητικών.
FN – False Negatives
Recall: Fraction of positives that were correctly identified.
Recall = TP/(TP+FN) όπου FN → false negatives
f1-score: είναι ο σταθμισμένος μέσος όρος Precision and Recall. Επομένως, αυτό το σκορ λαμβάνει υπόψη τόσο false positives όσο και false negatives. Διαισθητικά δεν είναι τόσο εύκολο

να κατανοηθεί όσο η ακρίβεια, αλλά το F1 είναι συνήθως πιο χρήσιμο από την ακρίβεια, ειδικά εάν έχετε μια άνιση κατανομή τάξης.
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Support: είναι ο αριθμός των πραγματικών εμφανίσεων της κλάσης στο καθορισμένο σύνολο δεδομένων. Η ανισορροπημένη υποστήριξη στα δεδομένα εκπαίδευσης μπορεί να υποδηλώνει δομικές αδυναμίες στις αναφερόμενες βαθμολογίες του classifier και μπορεί να υποδηλώνει την ανάγκη για στρωματοποιημένη δειγματοληψία ή επανεξισορρόπηση.

classification report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2056
1	0.94	0.98	0.96	62
accuracy			1.00	2118
macro avg	0.97	0.99	0.98	2118
weighted avg	1.00	1.00	1.00	2118

Σχήμα 6.2. Classification report

Ένα παράδειγμα εκτέλεσης του classification report φαίνεται στο Σχήμα 6.2 και συγκεκριμένα αυτά είναι τα αποτελέσματα ανίχνευσης της επίθεσης Sinkhole χρησιμοποιώντας τα FIT_DATA σε network detection για την τοπολογία Sink in the middle.

6.2 Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Σχήμα 6.3. Confusion matrix

Στο Σχήμα 6.3 βλέπουμε ένα παράδειγμα confusion Matrix.

TN → true negatives. Τα true negatives αναφέρονται στα αποτελέσματα που βρήκε το μοντέλο τα οποία έκανε predict ως benign και είναι όντως benign.

TP → true positives. Τα true positives αναφέρονται στα αποτελέσματα που βρήκε το μοντέλο τα οποία έκανε predict ως malicious και είναι όντως malicious.

FP → false positives. Τα false positives αναφέρονται στα αποτελέσματα που βρήκε το μοντέλο τα οποία έκανε predict ως malicious τα οποία στην πραγματικότητα είναι benign.

FN → false negatives. Τα false negatives αναφέρονται στα αποτελέσματα που βρήκε το μοντέλο τα οποία έκανε predict ως benign τα οποία στην πραγματικότητα είναι malicious.

Άρα όσα λιγότερα FP & FN έχουμε τόσο πιο αποτελεσματικό και σωστό είναι το μοντέλο μας.

```
confusion matrix
 [[2052    4]
 [   1   61]]
```

Σχήμα 6.4. Confusion Matrix Sinkhole

Ένα παράδειγμα εκτέλεσης του confusion matrix φαίνεται στο Σχήμα 6.4 και συγκεκριμένα αυτά είναι τα αποτελέσματα ανίχνευσης της επίθεσης Sinkhole χρησιμοποιώντας τα FIT_DATA σε network detection για την τοπολογία Sink in the middle.

Βλέπουμε ότι τα αποτελέσματα είναι σχεδόν άριστα αφού εντοπίζει σωστά 2052 benign τα οποία είναι όντως benign (TN), εντοπίζει σωστά 61 malicious τα οποία είναι όντως malicious (TP) αλλά εντοπίζει λανθασμένα 4 data points ως malicious αλλά είναι benign (FP) και λανθασμένα 1 data point ως benign αλλά πραγματικά είναι malicious (FN).

Επίσης από το confusion matrix μπορούμε να βρούμε πόσα συνολικά benign και malicious data points υπάρχουν. Συνολικά benign είναι τα TN + FP άρα 2056 και τα συνολικά malicious TP + FN άρα 62.

6.3 Accurate score

Το accurate score του μοντέλου είναι ουσιαστικά αυτό που συνήθως εννοούμε, όταν χρησιμοποιούμε τον όρο ακρίβεια. Είναι ο λόγος του αριθμού των σωστών προβλέψεων προς τον συνολικό αριθμό των δειγμάτων εισαγωγής.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Fig 19. Accuracy

```
accuracy score
 0.997639282341832
```

Σχήμα 6.5. Accurate score

Άρα για το πιο πάνω παράδειγμα το accurate Score είναι (2052 + 61) / (2056 + 62) το οποίο είναι 99.76% (όπως βλέπουμε και στο Fig20). Άρα ο Isolation Forest για το συγκεκριμένο παράδειγμα πετυχαίνει σκορ ακρίβειας 99.76%. Σχεδόν 100%. Άρα ο συγκεκριμένος αλγόριθμος ανιχνεύει την συγκεκριμένη επίθεση σχεδόν άριστα.

Κεφάλαιο 7

Αποτελέσματα Isolation Forest

Μέχρι στιγμής είδαμε και αναλύσαμε τους δύο αλγορίθμους που θα χρησιμοποιηθούν για να κάνουμε ανίχνευση των επιθέσεων, όπως επίσης και τις 5 διαφορετικές επιθέσεις που θα μελετήσουμε. Στην συνέχεια μιλήσαμε για όλα τα στάδια της προετοιμασίας δεδομένων. Έχουμε μιλήσει για καθαρισμό δεδομένων, έχουμε κάνει feature scaling, προχωρήσαμε στην επιλογή των features χρησιμοποιώντας τεχνικές Dimensionality Reduction και feature selection, κάναμε split τα data σε train και test και τέλος συντονίσαμε τις παραμέτρους που παίρνουν οι δύο αλγόριθμοι με τις οποίες θα γίνει το καλύτερο train των αλγορίθμων. Μένει πλέον πάρουμε και να αναλύσουμε τα αποτελέσματα των δύο αλγορίθμων. Έτσι σε αυτό το κεφάλαιο και το επόμενο θα δούμε ποια είναι τα αποτελέσματα του SOM και Isolation Forest ξεκινώντας από τον τελευταίο και θα εξηγήσουμε γιατί έχουμε αυτά τα αποτελέσματα καθώς επίσης και κατά πόσο είναι καλά ή όχι. Να σημειώσω ότι τα επόμενα 2 κεφάλαια που ακολουθούν θα είναι αρκετά μεγάλα για το λόγο ότι έχουμε πολλά σενάρια να ελέγξουμε και να αναλύσουμε.

7.1 BlackHole BN

Θα ξεκινήσουμε βλέποντας τα αποτελέσματα της επίθεσης blackHole BN στις 3 διαφορετικές τοπολογίες που έχουμε για τα δύο διαφορετικά datasets (FIT & COOJA) για τις δύο διαφορετικές ανιχνεύσεις (local & network).

FIT DATA Local Detection

Middle topology

```
classification report
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       622
           1       0.91      0.81      0.86        63

    accuracy                           0.98       685
   macro avg       0.95      0.90      0.92       685
weighted avg       0.97      0.98      0.97       685

confusion matrix
 [[617   5]
 [ 12  51]]

accuracy score
 0.9751824817518249
```

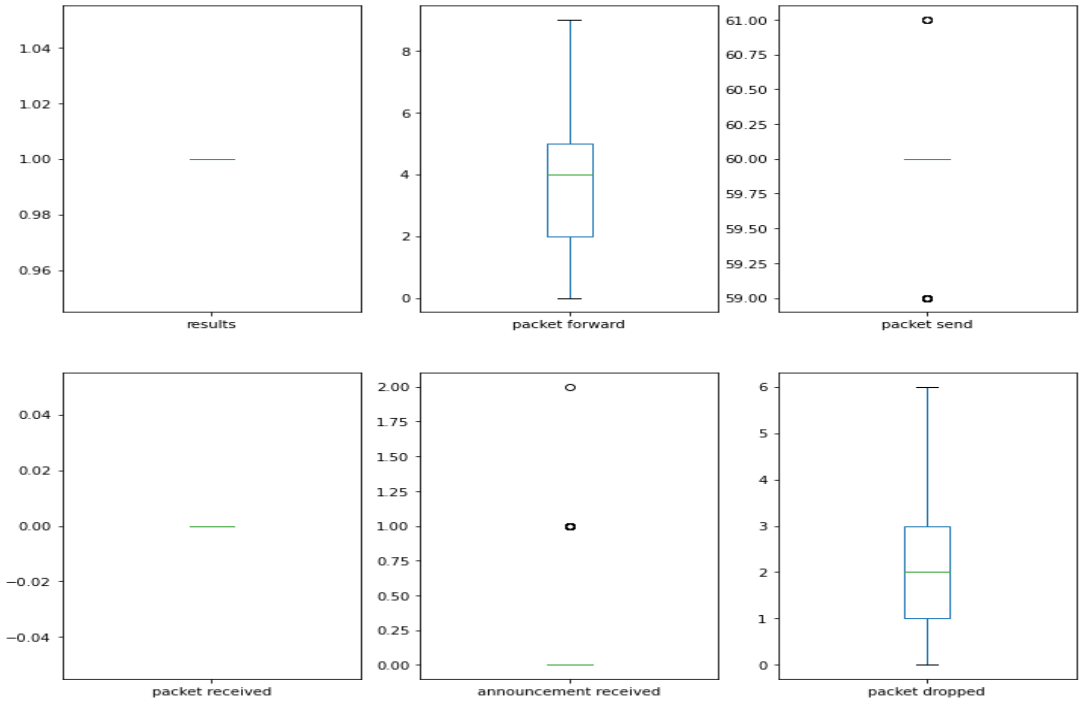
Σχήμα 7.1. BH BN middle local FIT results

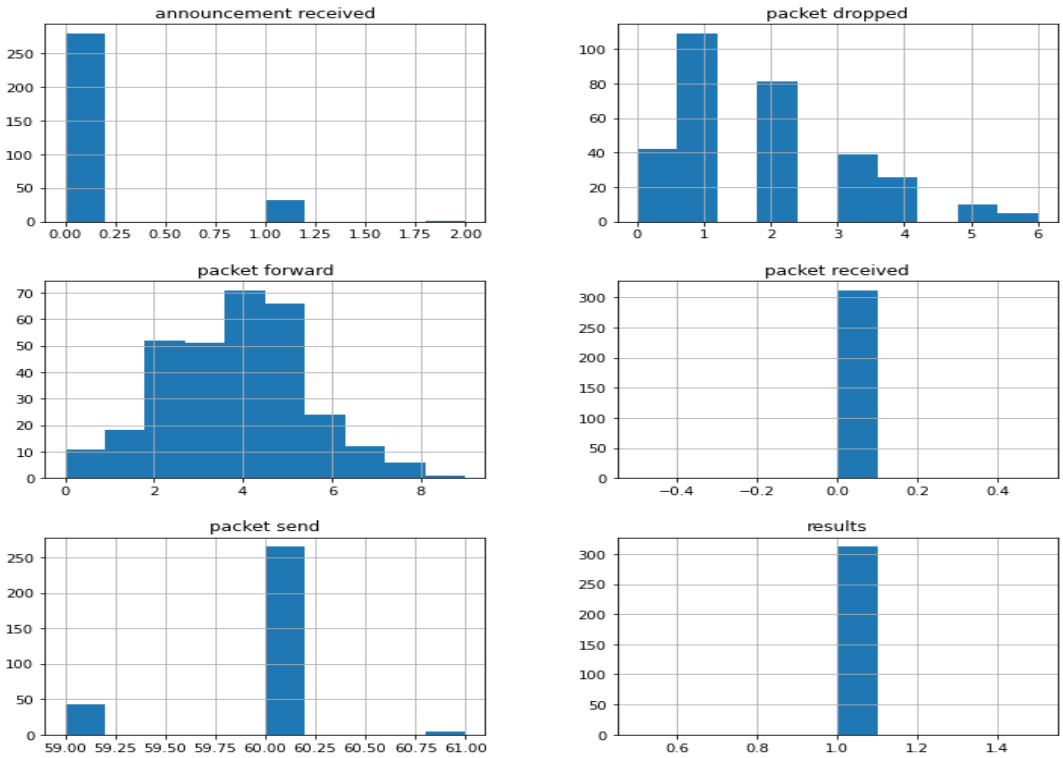
Στο Σχήμα 7.1 βλέπουμε τα αποτελέσματα local ανίχνευσης της επίθεσης blackhole BN για τα FIT data στην τοπολογία sink in the middle χρησιμοποιώντας τον Isolation Forest αλγόριθμο.

Τα αποτελέσματα είναι πάρα πολύ καλά πετυχαίνοντας ένα accurate score 97.5% με 12 false negatives data points και 5 false positives. Αυτό σημαίνει ότι ο αλγόριθμος έκανε predict λανθασμένα 12 data points σαν benign ενώ στην πραγματικότητα είναι malicious και 5 data points

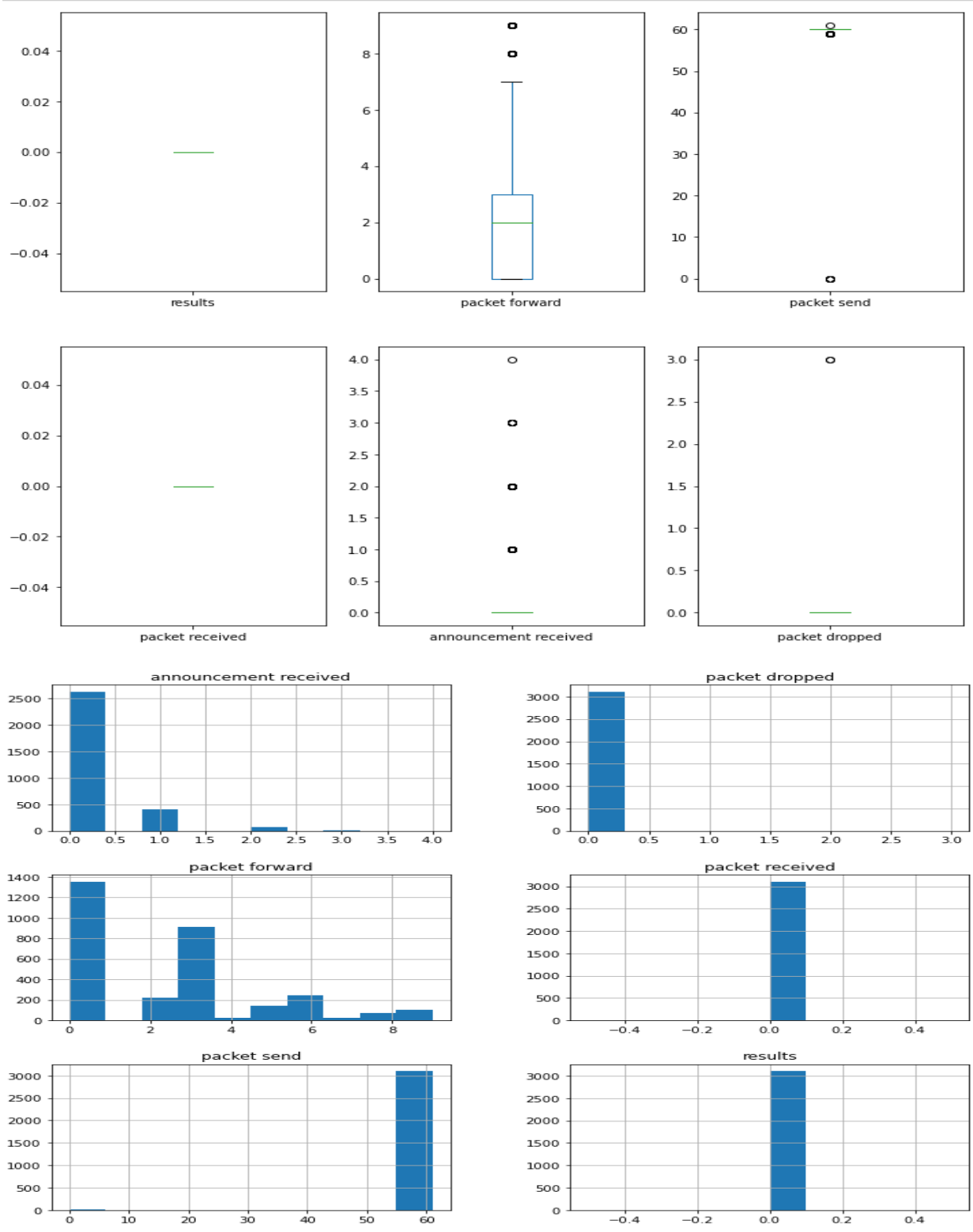
malicious ενώ στην πραγματικότητα είναι benign. . Όμως εντόπισε σωστά 51 malicious data points και 617 σωστά benign data points.

Επίσης πετυχαίνουμε ένα πολύ καλό precision και recall τόσο για τα benign όσο και για τα malicious data points.





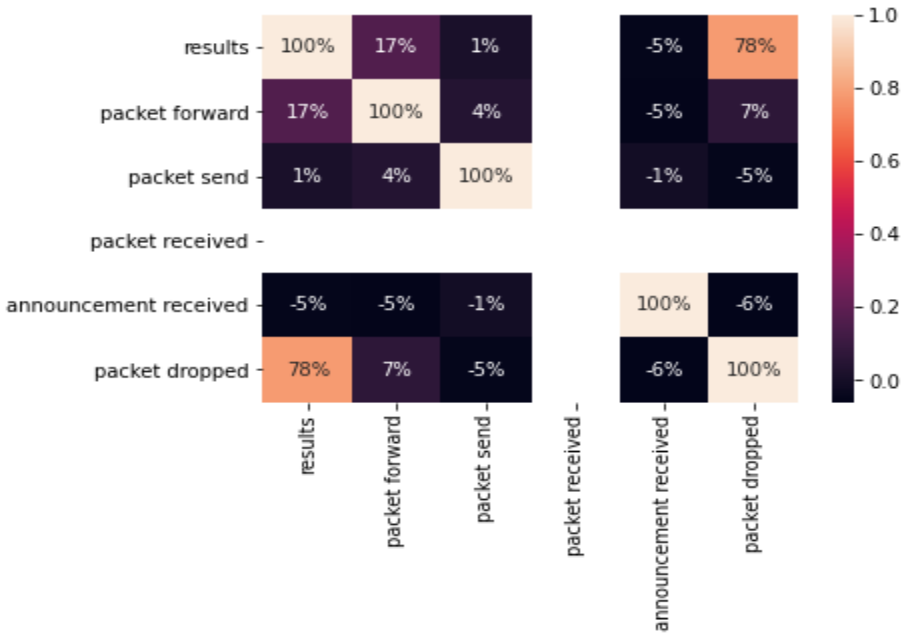
Σχήμα 7.2. Data points values for malicious set



Σχήμα 7.3. Data points values for benign set

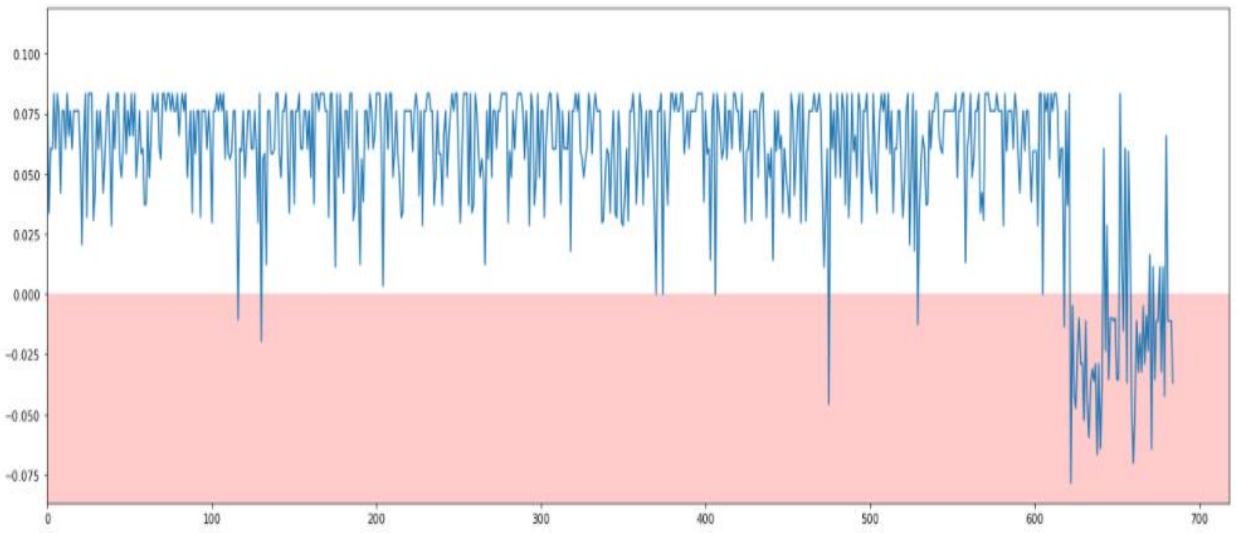
Στο Σχήμα 7.2 βλέπουμε τις τιμές που κυμαίνονται τα δεδομένα του malicious node και στο Σχήμα 7.3 βλέπουμε τις τιμές που κυμαίνονται τα δεδομένα του sink node (benign).

Αναμένουμε να δούμε διαφορές στα features packet forward και packet dropped. Αυτό επειδή όπως είδαμε στο κεφάλαιο 3 η επίθεση blackHole προσπαθεί να αλλάξει την δρομολόγηση των πακέτων όπου αντί να κάνει forward τα πακέτα τα κάνει drop. Άρα είναι λογικό να έχουμε μια διαφορά σε αυτά τα features. Αυτό ισχύει, εφόσον από τα σχήματα παρατηρούμε ότι όντως ο malicious node ρίχνει πακέτα και κάνει προωθεί πακέτα σε πολύ διαφορετικό τρόπο απ' ότι ο sink node. Άρα υπάρχει ανωμαλία σε αυτά τα 2 features και ο αλγόριθμος μας πρέπει να είναι σε θέση να εντοπίσει αυτές τις ανωμαλίες πράγμα που κάνει αποτελεσματικά.



Σχήμα 7.4. Correlation Matrix between features

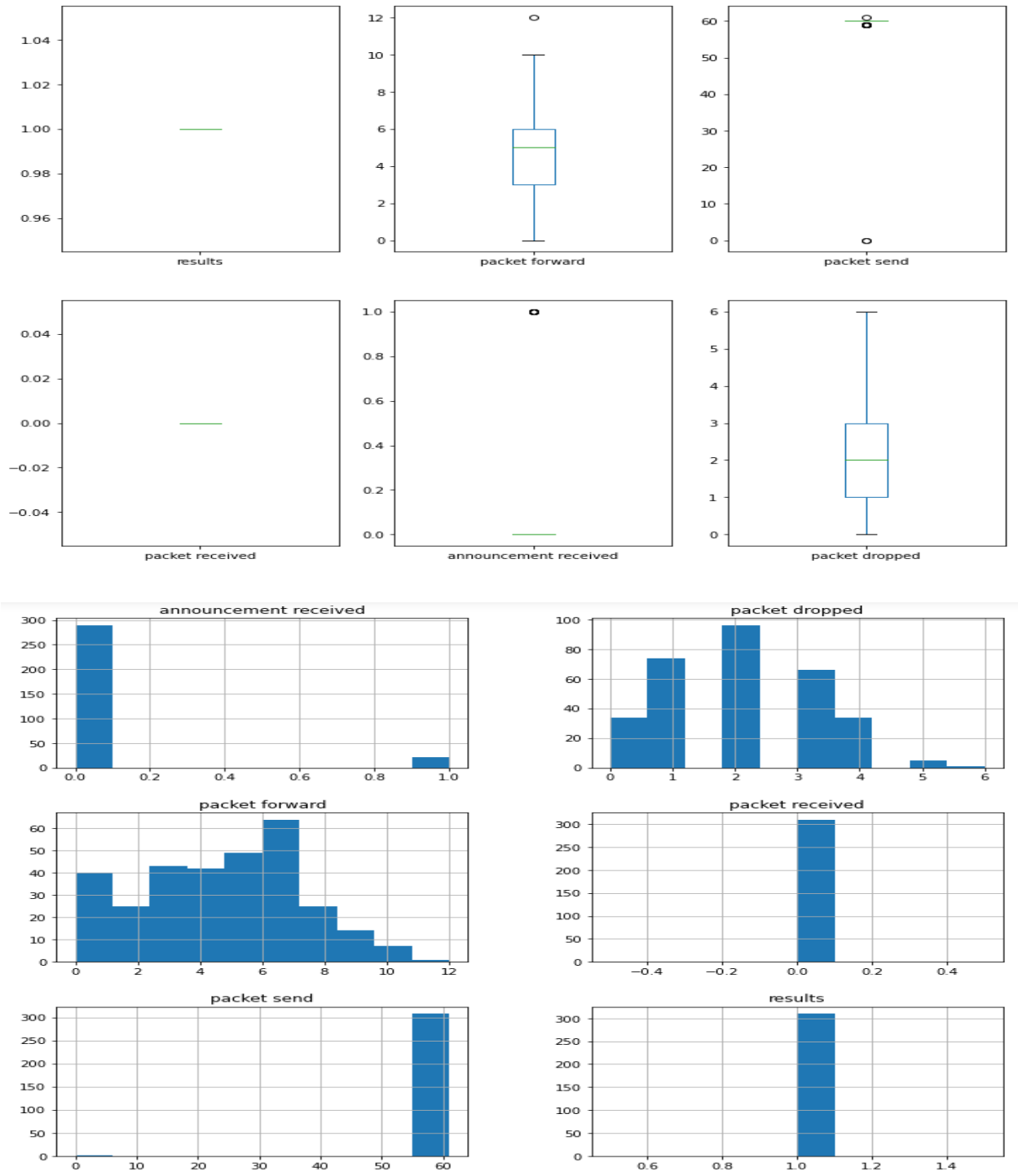
Ακόμη μπορούμε να δούμε από το Σχήμα 7.4 ότι τα features packet dropped & packet forward έχουν ψηλό correlation σε σχέση με το target variable, 78% και 17% αντίστοιχα.



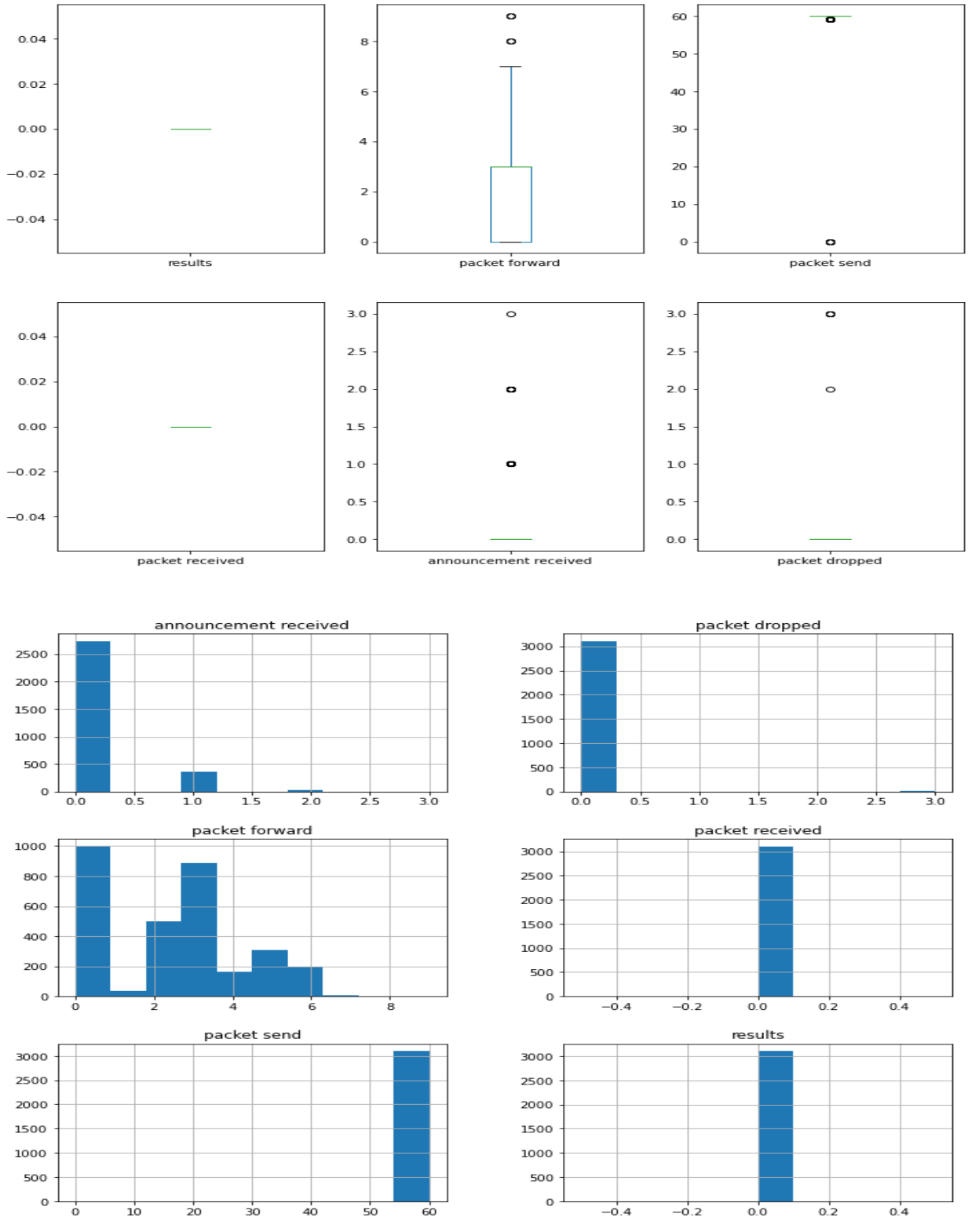
Σχήμα 7.5. Health Plot showing the anomalies

Στο Σχήμα 7.5 βλέπουμε την γραφική η οποία παρουσιάζει το health των data points που κάνει predict ο αλγόριθμος Isolation Forest. Βλέπουμε ξεκάθαρα ότι μπορεί να εντοπίσει με επιτυχία τα anomalies τα οποία βρίσκονται προς το τέλος της γραφικής τα οποία ξεχωρίζουν από τα άλλα data points.

Top topology

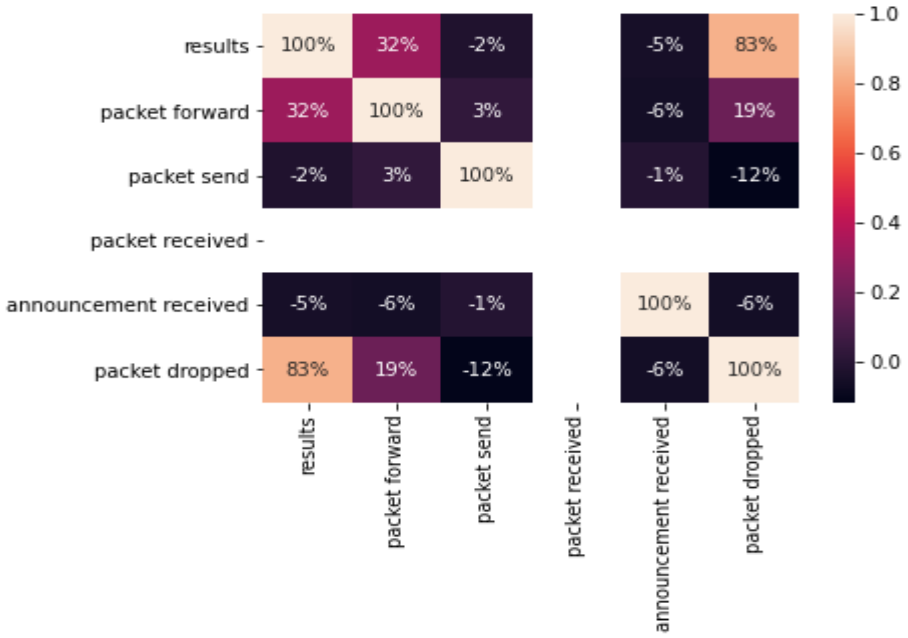


Σχήμα 7.6. Data points values for malicious set



Σχήμα 7.7. Data points values for benign set

Παρομοίως στο σύμφωνα με τα Σχήματα 7.6 & 7.7 αναμένουμε να έχουμε πολύ καλά αποτελέσματα και στο sink in top topology.



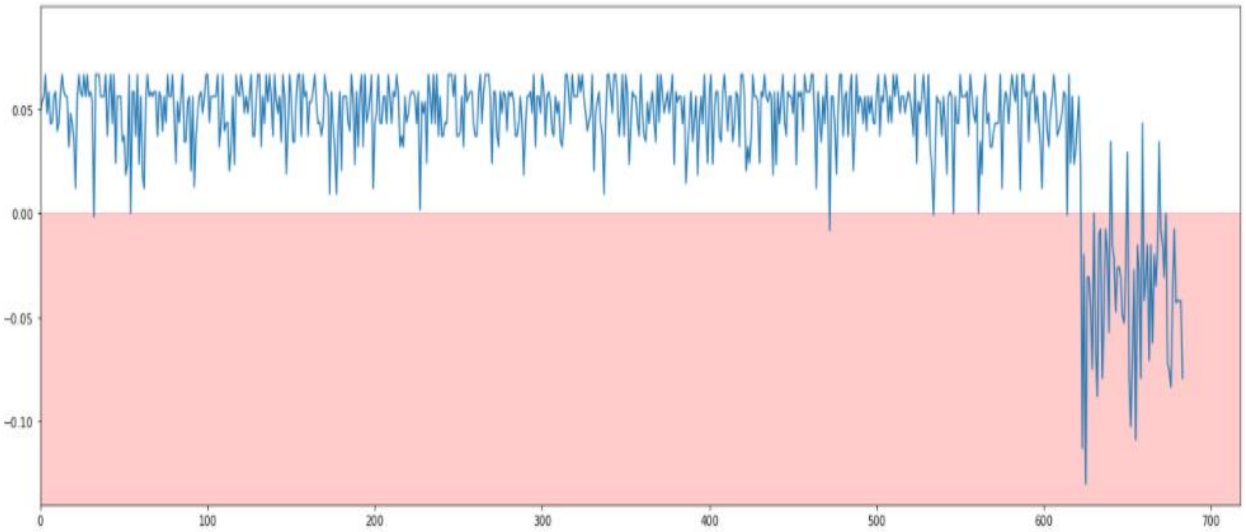
Σχήμα 7.8. Correlation Matrix between features

Ακόμη από το correlation matrix παρατηρούμε ότι τα αποτελέσματα μπορεί να είναι ακόμη καλύτερα και από το middle topology εφόσον το correlation μεταξύ των features packet dropped και packet forward και του target variable είναι μεγαλύτερο απ' ότι στο middle topology.

```
classification report
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       622
           1       0.93      0.89      0.91        62

    accuracy                           0.98       684
   macro avg       0.96      0.94      0.95       684
weighted avg       0.98      0.98      0.98       684

confusion matrix
 [[618   4]
 [  7  55]]

accuracy score
 0.9839181286549707
```

Σχήμα 7.9. BH BN top local FIT results

Όντως τα αποτελέσματα είναι ακόμη καλύτερα και συγκεκριμένα είναι σχεδόν τέλια πετυχαίνοντας 98.3 % accuracy score και πολύ ψηλό precision και recall.



Σχήμα 7.10. Health Plot showing the anomalies

Από το Σχήμα 7.10 βλέπουμε ξεκάθαρα ότι μπορεί να εντοπίσει με επιτυχία τα anomalies τα οποία βρίσκονται προς το τέλος της γραφικής τα οποία ξεχωρίζουν από τα άλλα data points.

Τα ποιο πάνω αποτελέσματα για το top topology παράχθηκαν από νέο μοντέλο το οποίο έγινε train και evaluate ξεχωριστά απ' ότι στο middle topology.

```
classification report
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       622
           1       0.89      0.81      0.85        62

    accuracy                           0.97       684
   macro avg       0.94      0.90      0.92       684
weighted avg       0.97      0.97      0.97       684

confusion matrix
 [[616   6]
 [ 12  50]]

accuracy score
 0.9736842105263158
```

Σχήμα 7.11. BH BN top local FIT results using same model as in middle

Στο Σχήμα 7.9 βλέπουμε τα αποτελέσματα τα οποία βγαίνουν από το μοντέλο το οποίο έγινε train στο middle topology. Αυτό το train μοντέλο χρησιμοποιήθηκε και στο top topology για να δούμε ποια θα ήταν η συμπεριφορά του στο top topology. Βλέπουμε από τα αποτελέσματα ότι έχουμε εξίσου καλά αποτελέσματα όμως από το Σχήμα 7.10 παρατηρούμε ότι αν χρησιμοποιήσουμε ένα νέο μοντέλο με καινούργιο train τα αποτελέσματα είναι ελαφρώς καλύτερα. Είναι στο χέρι μας αν θέλουμε να χρησιμοποιήσουμε το μοντέλο του middle topology το ίδιο για το top topology.

FIT DATA Network Detection

Middle topology

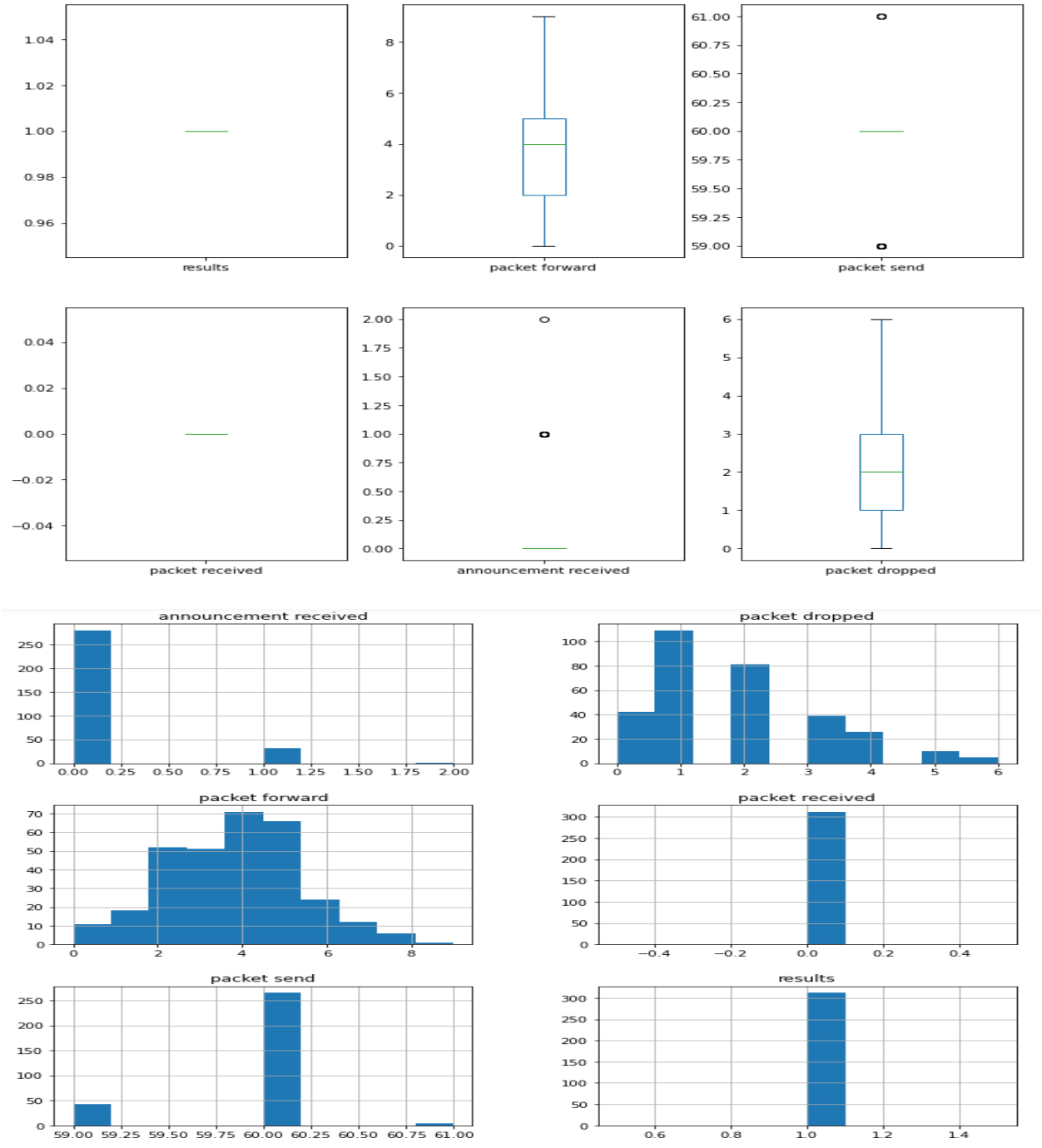
```
classification report
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2057
           1       0.80      0.81      0.80        63

    accuracy                           0.99      2120
   macro avg       0.90      0.90      0.90      2120
weighted avg       0.99      0.99      0.99      2120

confusion matrix
 [[2044   13]
 [  12   51]]

accuracy score
 0.9882075471698113
```

Σχήμα 7.12. BH BN middle network FIT results

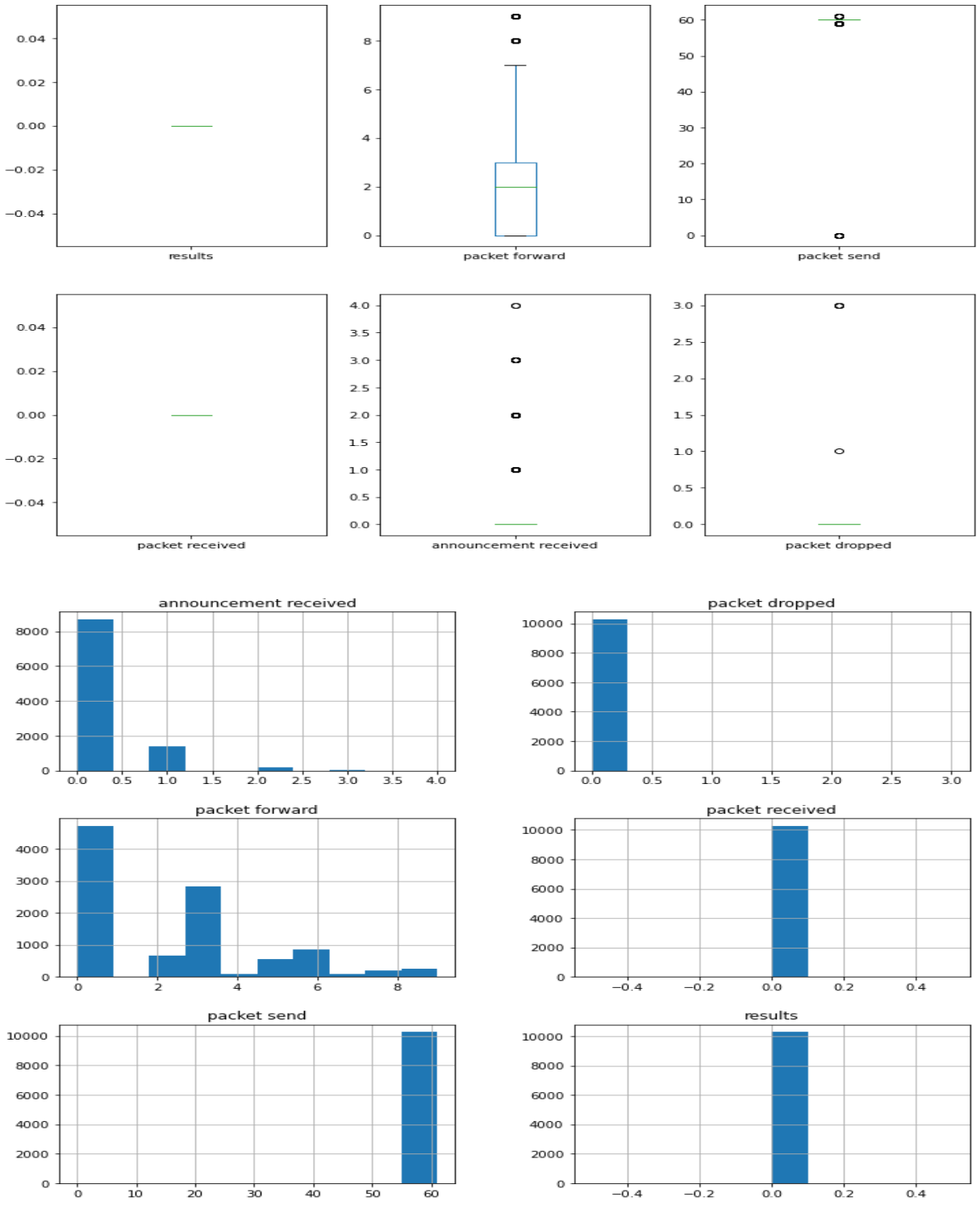
Στο Σχήμα 7.12 βλέπουμε τα αποτελέσματα network ανίχνευσης της επίθεσης blackhole BN για τα FIT data στην τοπολογία sink in the middle χρησιμοποιώντας τον Isolation Forest αλγόριθμο.

Τα αποτελέσματα είναι πάρα πολύ καλά πετυχαίνοντας ένα accurate score 98.8% με 12 false negatives data points και 13 false positives. Αυτό σημαίνει ότι ο αλγόριθμος έκανε predict λανθασμένα 12 data points σαν benign ενώ στην πραγματικότητα είναι malicious και 13 data points malicious ενώ στην πραγματικότητα είναι benign. . Όμως εντόπισε σωστά 51 malicious data points και 2044 σωστά benign data points.

Επίσης πετυχαίνουμε ένα πολύ καλό precision και recall τόσο για τα benign όσο και για τα malicious data points.

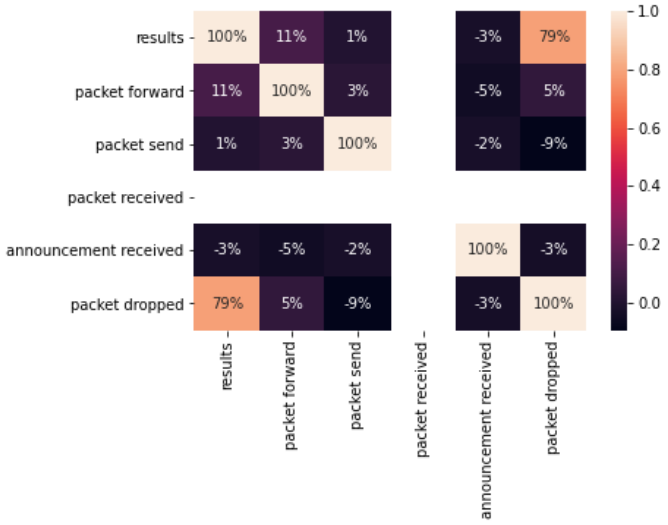


Σχήμα 7.13. Data points values for malicious set



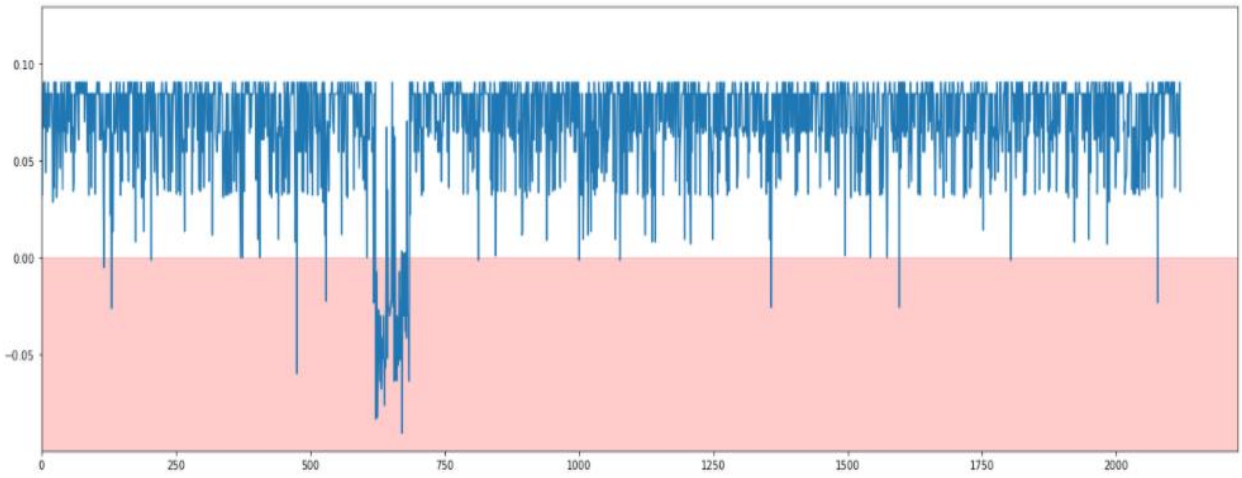
Σχήμα 7.14. Data points values for benign set

Όπως και στο local detection έτσι και στο Network detection βλέπουμε ότι τα οι τιμές των δεδομένων κυμαίνονται με παρόμοιο τρόπο έχοντας διαφορά στα features packet forward & packet dropped. Άρα ο αλγόριθμος για ακόμη μια φορά όπως και στο local detection προσπαθεί να βρει τις ανωμαλίες (malicious data points) βασιζόμενος κυρίως σε αυτά τα features πράγμα που όπως είδαμε και στο Σχήμα 7.12 κάνει με πολύ αποτελεσματικό τρόπο.



Σχήμα 7.15. Correlation Matrix between features

Βλέπουμε από τον correlation matrix στο Σχήμα 7.15 ότι τα συγκεκριμένα 2 features έχουν αρκετή επίδραση στο target variable.



Σχήμα 7.16. Health Plot showing the anomalies

Επίσης βλέπουμε από το Σχήμα 7.16 την γραφική του health για την επίθεση blackHole BN σε network detection στο middle topology, ότι ο αλγόριθμος είναι σε θέση να εντοπίσει τα anomalies.

Top topology

Ακολουθώντας ακριβώς την ίδια μεθοδολογία θα δούμε τα αποτελέσματα που παίρνουμε από την network ανίχνευση του blackHole BN σε top topology για τα FIT DATA.

```
classification report
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      2054
           1       0.67      0.85      0.75        62

    accuracy                           0.98      2116
   macro avg       0.83      0.92      0.87      2116
weighted avg       0.99      0.98      0.98      2116

confusion matrix
 [[2028   26]
 [   9   53]]

accuracy score
 0.9834593572778828
```

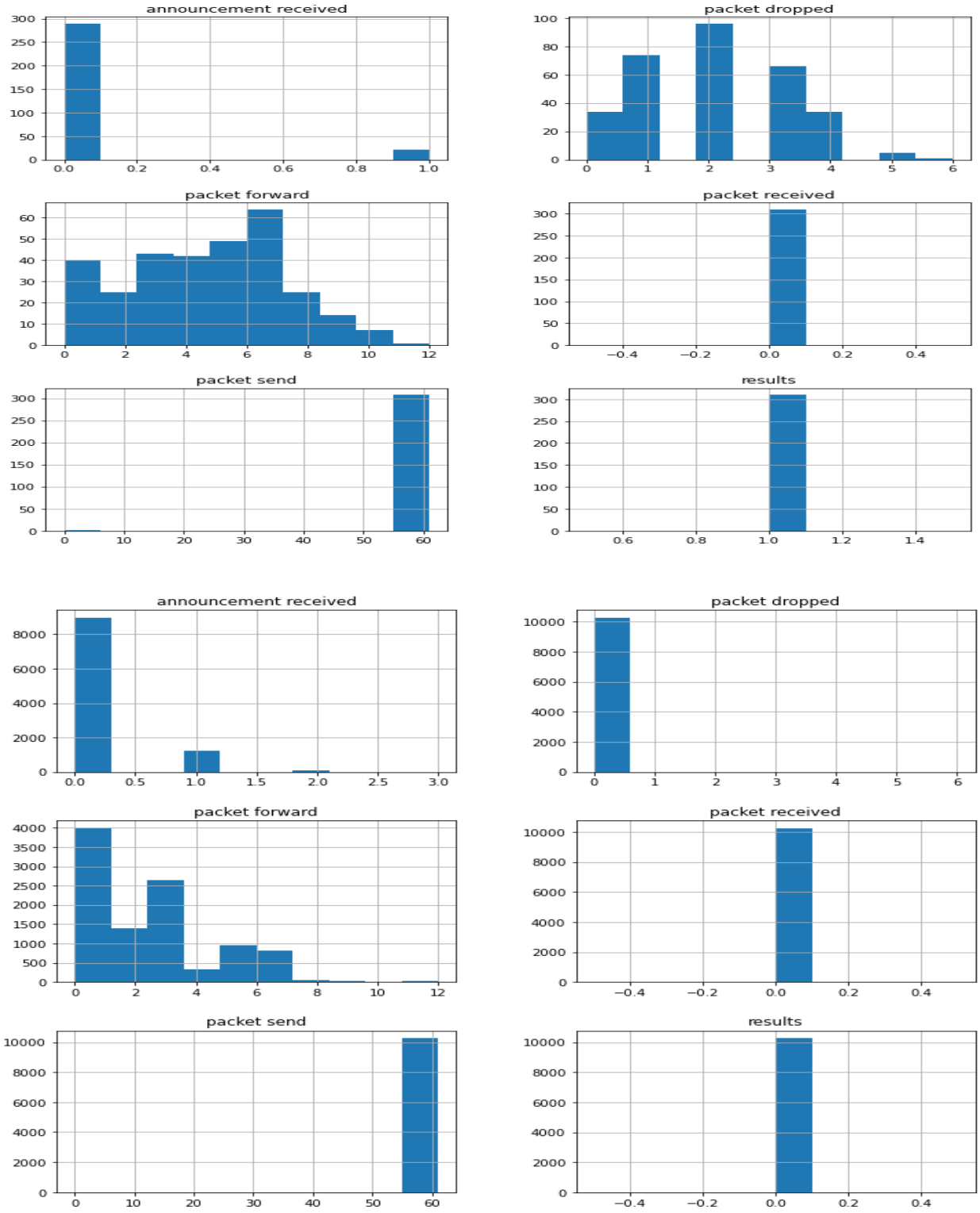
Σχήμα 7.17. BH BN top network FIT results using same model as in middle

Αρχικά όπως κάναμε και στο local detection, χρησιμοποιούμε το trained model που δημιουργήθηκε στο middle topology. Βλέπουμε από το Σχήμα 7.17 ότι τα αποτελέσματα είναι για ακόμη μια φορά πολύ καλά στον εντοπισμό της συγκεκριμένης επίθεσης όμως παρατηρούμε ότι έχουμε μια αύξηση στα false positives. Έτσι εκπαιδεύσαμε νέο μοντέλο για το top topology όπως κάναμε και στο local detection για να δούμε τις διαφορές και αν μας δίνει καλύτερα αποτελέσματα.

```
classification report
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      2054
           1       0.88      0.82      0.85        62

    accuracy                           0.99      2116
   macro avg       0.94      0.91      0.92      2116
weighted avg       0.99      0.99      0.99      2116

confusion matrix
 [[2047    7]
 [  11   51]]

accuracy score
 0.9914933837429112
```

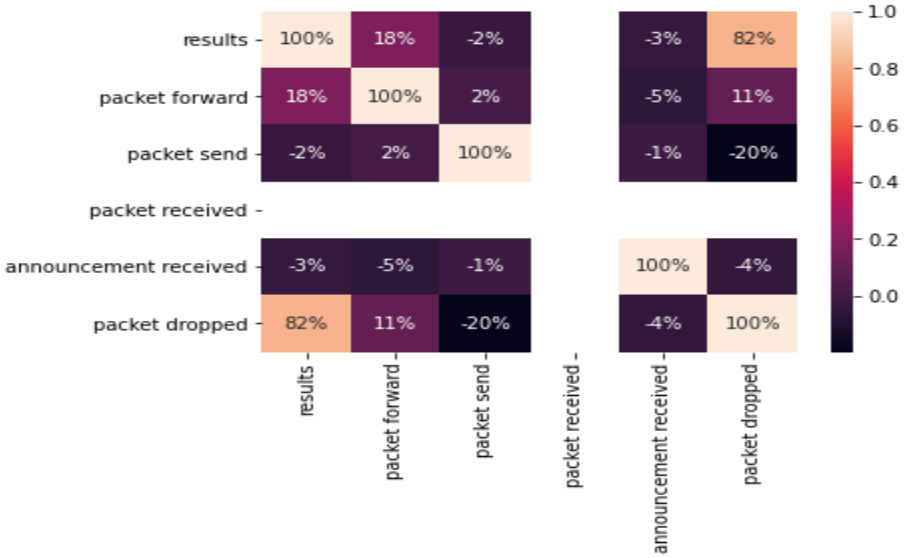
Σχήμα 7.18. BH BN top network FIT results

Παρατηρούμε ότι έχουμε καλύτερα αποτελέσματα με το train ενός νέου μοντέλου παρά με την χρήση του ίδιου από το middle topology.

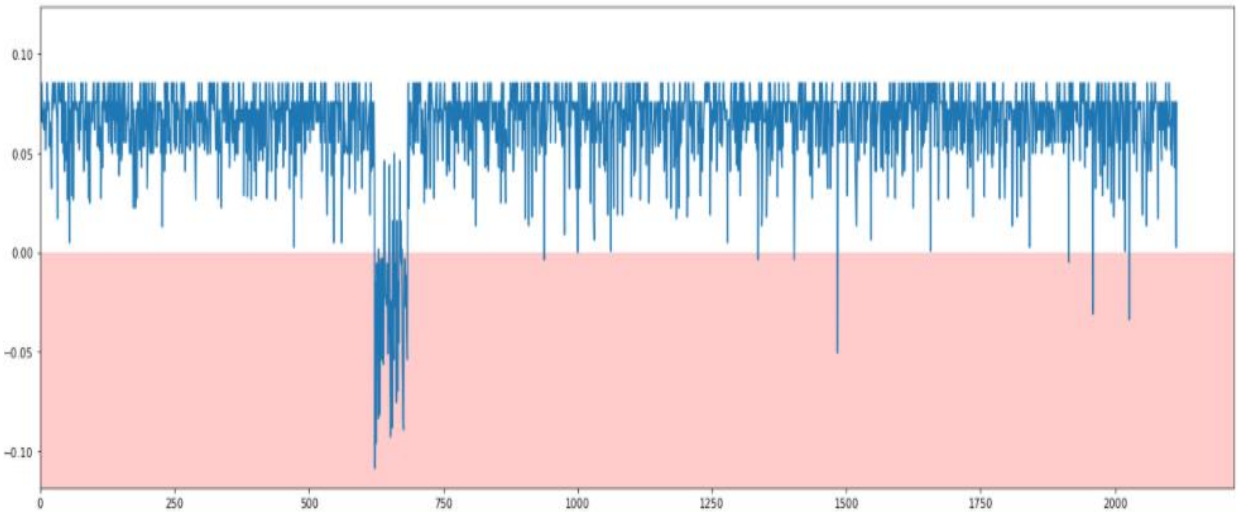


Σχήμα 7.19. Data points values for malicious and benign set

Από το Σχήμα 7.19 βλέπουμε ότι και στο network topology υπάρχει διαφορά στην διακύμανση των τιμών των features packet dropped & packet forward του malicious node με τους άλλους nodes του δικτύου πράγμα που αναμέναμε. Έτσι ο αλγόριθμος ξανά εντοπίζει επιτυχώς τα anomalies όπως είδαμε και πιο πάνω στο Σχήμα 7.18.



Σχήμα 7.20. Correlation Matrix between features



Σχήμα 7.21. Health Plot showing the anomalies

Στα Σχήμα 7.19 και Σχήμα 7.21 για ακόμη μια φορά βλέπουμε παρόμοια αποτελέσματα όπως και στο top topology του local detection.

7.2 BlackHole FR

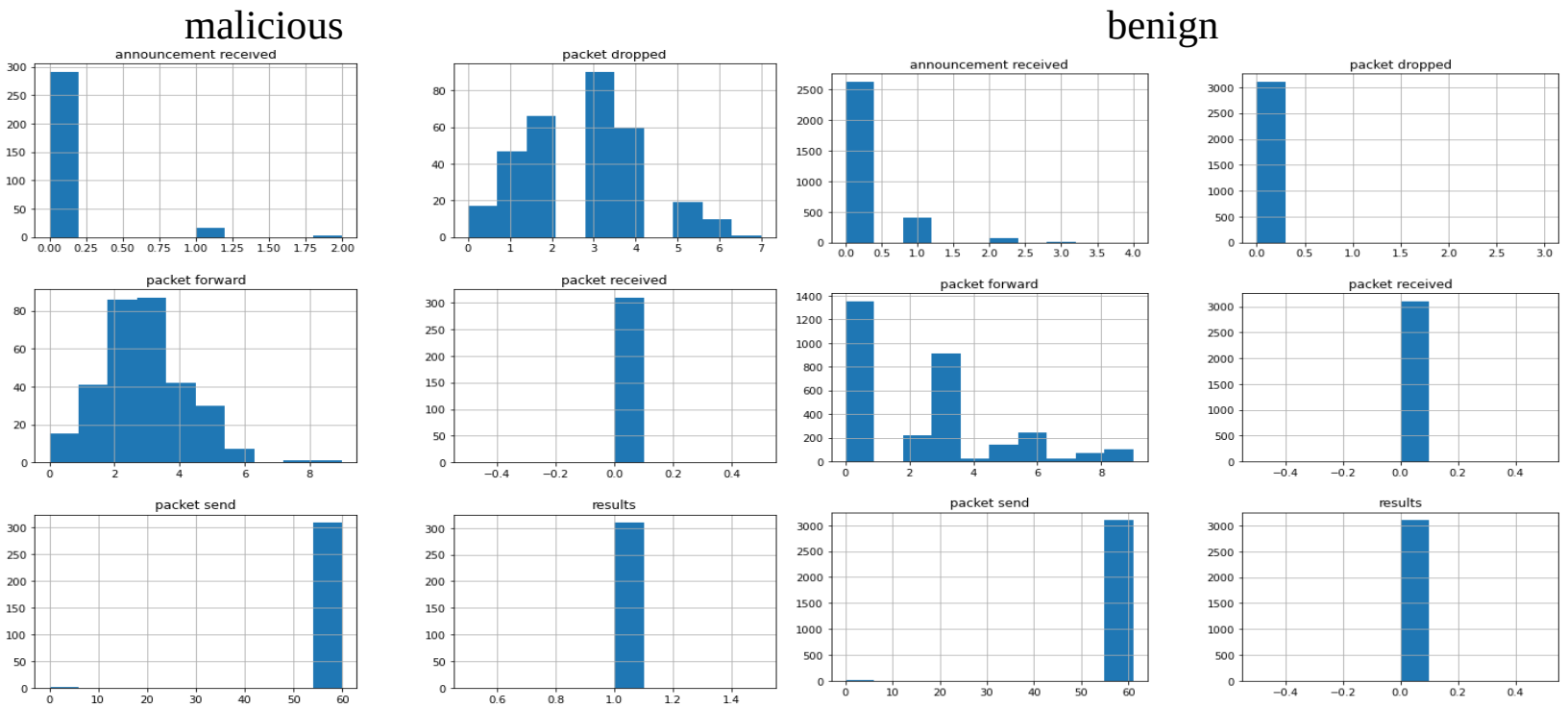
FIT DATA Local Detection

Middle topology

```
classification report
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       622
           1       0.91      0.79      0.84        62

    accuracy                           0.97       684
   macro avg       0.94      0.89      0.92       684
weighted avg       0.97      0.97      0.97       684

confusion matrix
 [[617   5]
 [ 13  49]]

accuracy score
 0.9736842105263158
```

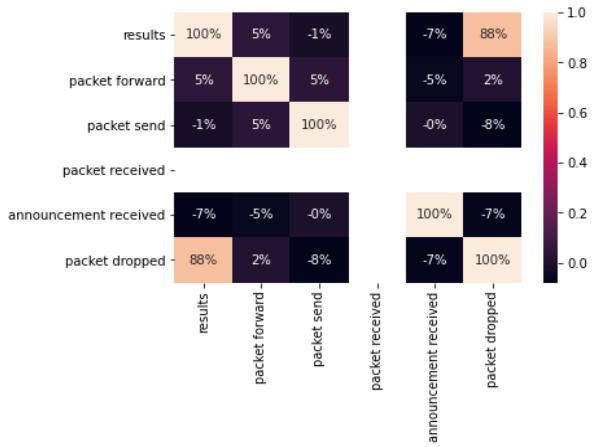
Σχήμα 7.22. BH FR middle local FIT results

Μελετάμε τώρα την επίθεση BlackHole FR για local detection. Αναμένουμε ότι θα έχουμε παρόμοια αποτελέσματα με την επίθεση blackhole BN. Παρατηρούμε από το Σχήμα 7.22 ότι τα αποτελέσματα που έχουμε είναι πολύ καλά πετυχαίνοντας 97.4% accuracy score, ακρίβεια (precision) 98% για τα benign predictions και 91% για τα malicious predictions και recall 99% για τα benign και 79% για τα malicious. Τέλος έχουμε 13 FN και 5 FP.



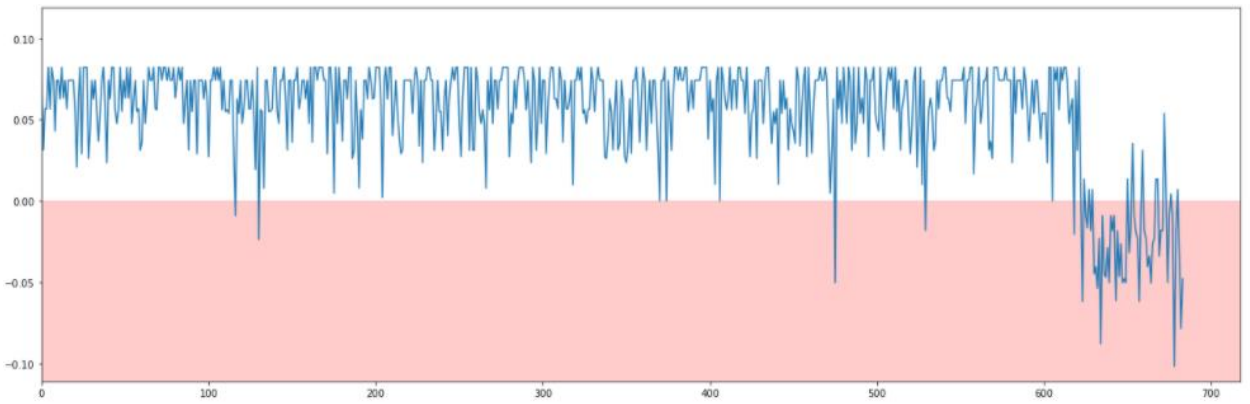
Σχήμα 7.23. Data points values for malicious and benign set

Από το Σχήμα 7.23 παρατηρούμε ότι υπάρχει μια διαφορά στις τιμές που κυμαίνονται για τα features packet dropped και packet forward μεταξύ του malicious node και του sink node. Αναμενόμενο αφού για ακόμη μια φορά μιλάμε για μια blackhole επίθεση forwarding ratio (FR) η οποία χρησιμοποιεί μια αναλογία (ratio) για να καθορίσει εάν θα προωθηθεί ή να γίνει drop ένα ληφθέν πακέτο. Έτσι παρατηρείτε η διαφορά στα features packet dropped & packet forward.



Σχήμα 7.24. Correlation between features

Βλέπουμε την επίδραση του feature packet dropped για την επίθεση BlackHole FR από το Σχήμα 7.24.



Σχήμα 7.25. Health Plot showing the anomalies

Βλέπουμε από το Σχήμα 7.25 ότι ο αλγόριθμος εντοπίζει αποτελεσματικά τα anomalies.

Top topology

```
classification report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       622
           1       0.82      0.79      0.81        63

    accuracy                           0.96       685
   macro avg       0.90      0.89      0.89       685
weighted avg       0.96      0.96      0.96       685

confusion matrix
 [[611  11]
 [ 13  50]]

accuracy score
 0.964963503649635
```

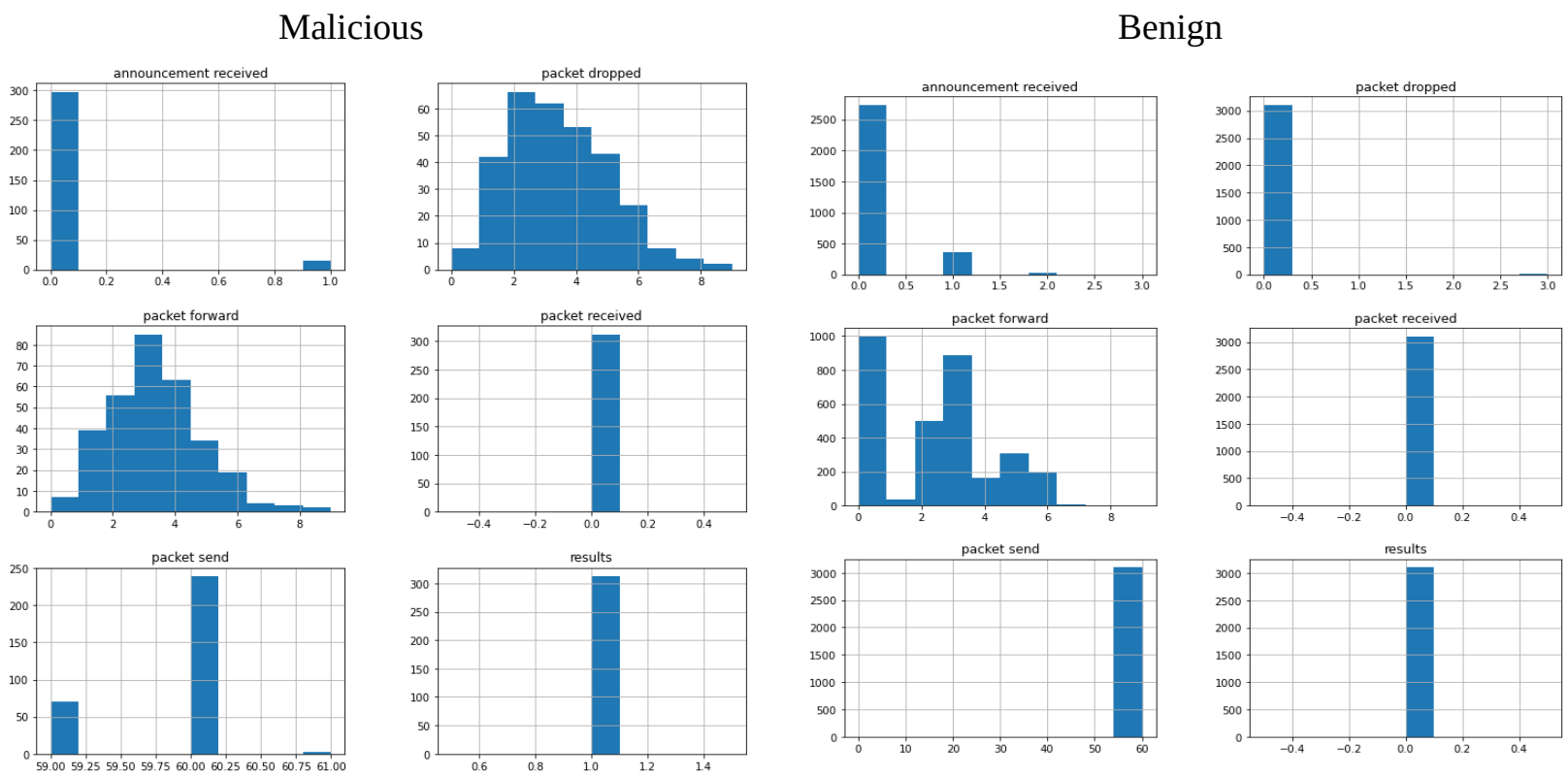
Σχήμα 7.26. BH FR top local using middle model FIT results

Στο Σχήμα 7.26 παίρνουμε τα αποτελέσματα για το top topology χρησιμοποιώντας το train model που εκπαιδεύσαμε στο middle topology. Τα αποτελέσματα είναι πολύ καλά με accuracy score 96.5%. Παρατηρούμε μια μικρή αύξηση στα FP. Θα δοκιμάσουμε να εκπαιδεύσουμε νέο μοντέλο για το top topology για να δούμε την διαφορά.

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       622
           1       1.00      0.83      0.90        63

    accuracy                           0.98       685
   macro avg       0.99      0.91      0.95       685
weighted avg       0.98      0.98      0.98       685

confusion matrix
 [[622   0]
 [ 11  52]]

accuracy score
 0.983941605839416
```

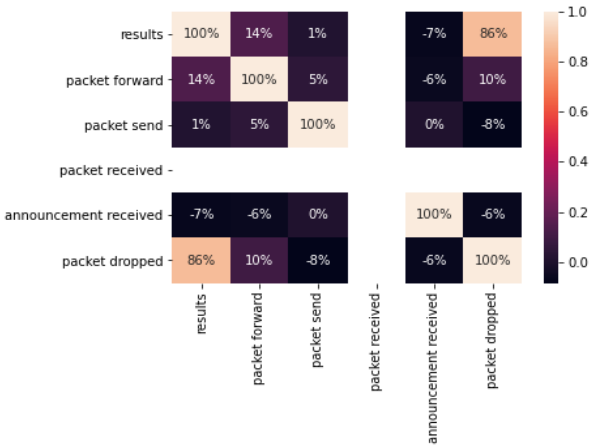
Σχήμα 7.27. BH FR top local using new trained model

Από το νέο εκπαιδευμένο μοντέλο καταφέραμε να εξαφανίσουμε τα FP δίνοντας μας ακόμη καλύτερα αποτελέσματα. Ας δούμε για ακόμη μια φορά τις τιμές που παίρνουν τα δεδομένα μας για να μπορούμε να εντοπίσουμε τους λόγους που ο Isolation Forest είναι αποτελεσματικός για αυτό το attack.



Σχήμα 7.28. Data points values for malicious and benign set

Είναι ξεκάθαρο ότι τα blackHole attacks (BN & FR) καθορίζονται από τα features packet dropped και packet forward. Από την φύση των attacks και το πως δουλεύουν αυτό το γεγονός είναι αναμενόμενο. Στο Σχήμα 7.28 βλέπουμε ξανά την διαφορά στις τιμές αυτών των features για τον malicious node (αριστερά) και για τον sink node (δεξιά).



Σχήμα 7.29. Correlation between features

FIT DATA Network Detection

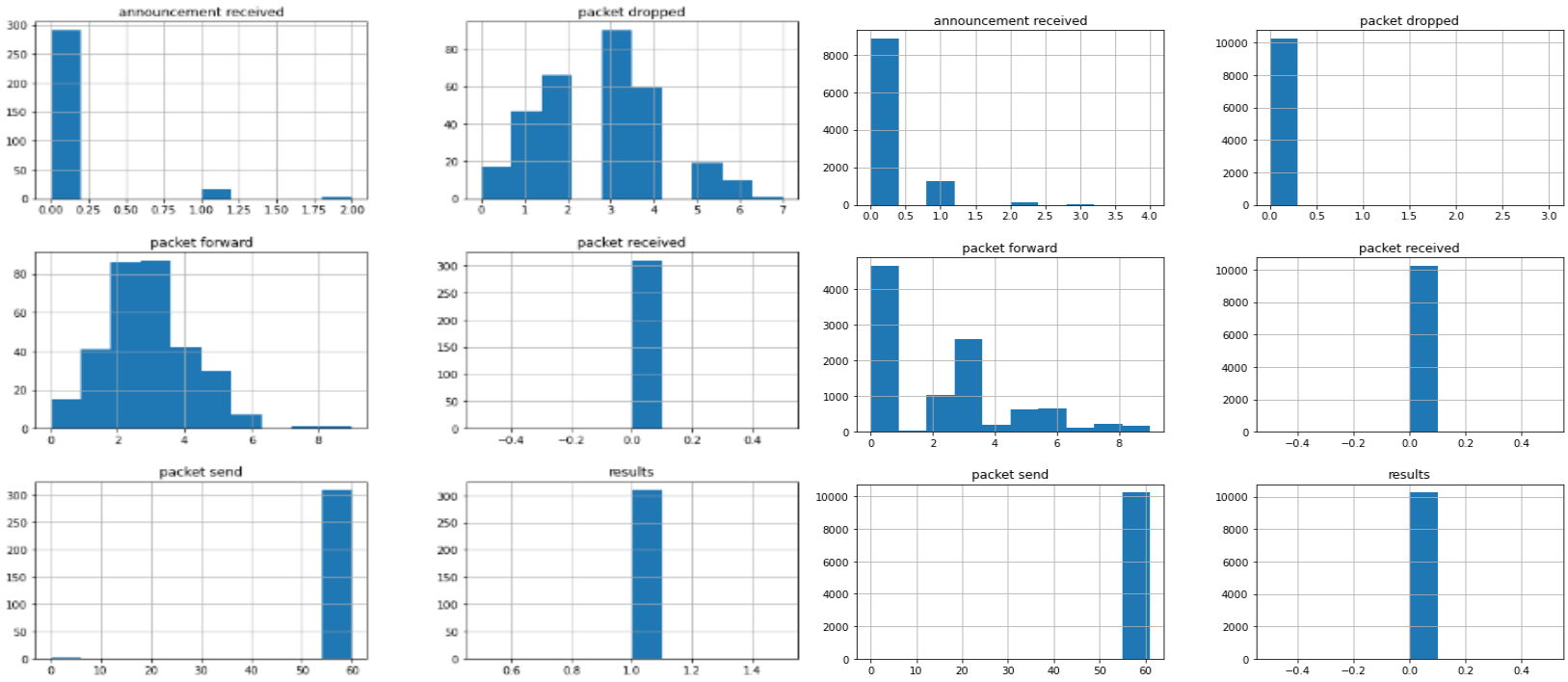
Middle topology

```
classification report
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2054
           1       0.78      0.79      0.78        62

    accuracy                           0.99      2116
   macro avg       0.89      0.89      0.89      2116
weighted avg       0.99      0.99      0.99      2116

confusion matrix
 [[2040   14]
 [  13   49]]

accuracy score
 0.9872400756143668
```

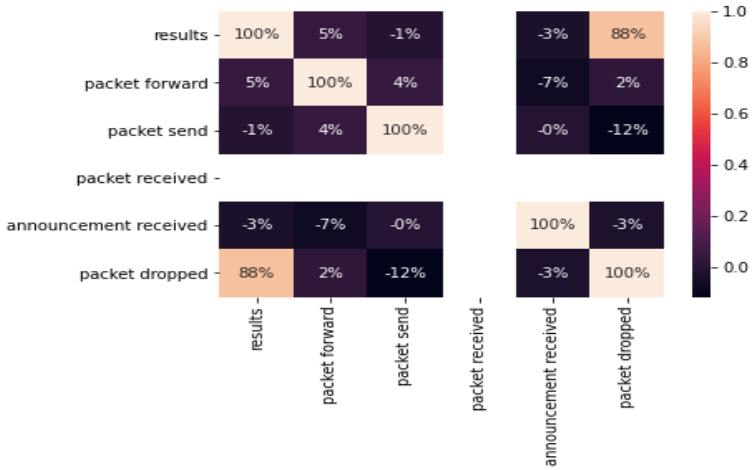
Σχήμα 7.30. BH FR middle network FIT results

Τα αποτελέσματα στο network detection για το blackhole FR είναι πάλι πάρα πολύ καλά πετυχαίνοντας 98.7% accuracy score και ένα αρκετά καλό confusion matrix Σχήμα 7.30. Ο λόγος είναι ο ίδιος όπως και στο local detection. Δηλαδή τα δύο features που έχουν μεγάλη απόκλιση στις τιμές τους για τον malicious node και για τους άλλους nodes του δικτύου συμπεριλαμβανομένου και του sink node.

malicious benign

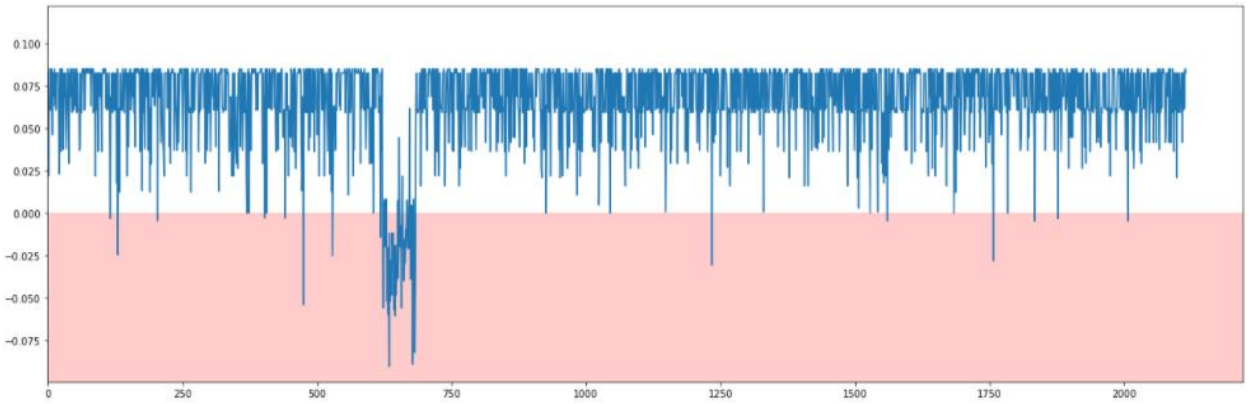


Σχήμα 7.31. Data points values for malicious and benign set



Σχήμα 7.32. Correlation Matrix

Όπως βλέπουμε για μια ακόμη φορά από τα Σχήμα 7.31 και Σχήμα 7.32 βλέπουμε τους ίδιους λόγους που ο αλγόριθμος μας καταφέρνει να εντοπίσει τα malicious data points.



Σχήμα 7.33. Health Plot showing the anomalies

Top topology

```
classification report
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2055
           1       0.76      0.81      0.78        63

    accuracy                           0.99      2118
   macro avg       0.88      0.90      0.89      2118
weighted avg       0.99      0.99      0.99      2118

confusion matrix
 [[2039   16]
 [  12   51]]

accuracy score
 0.9867799811142587
```

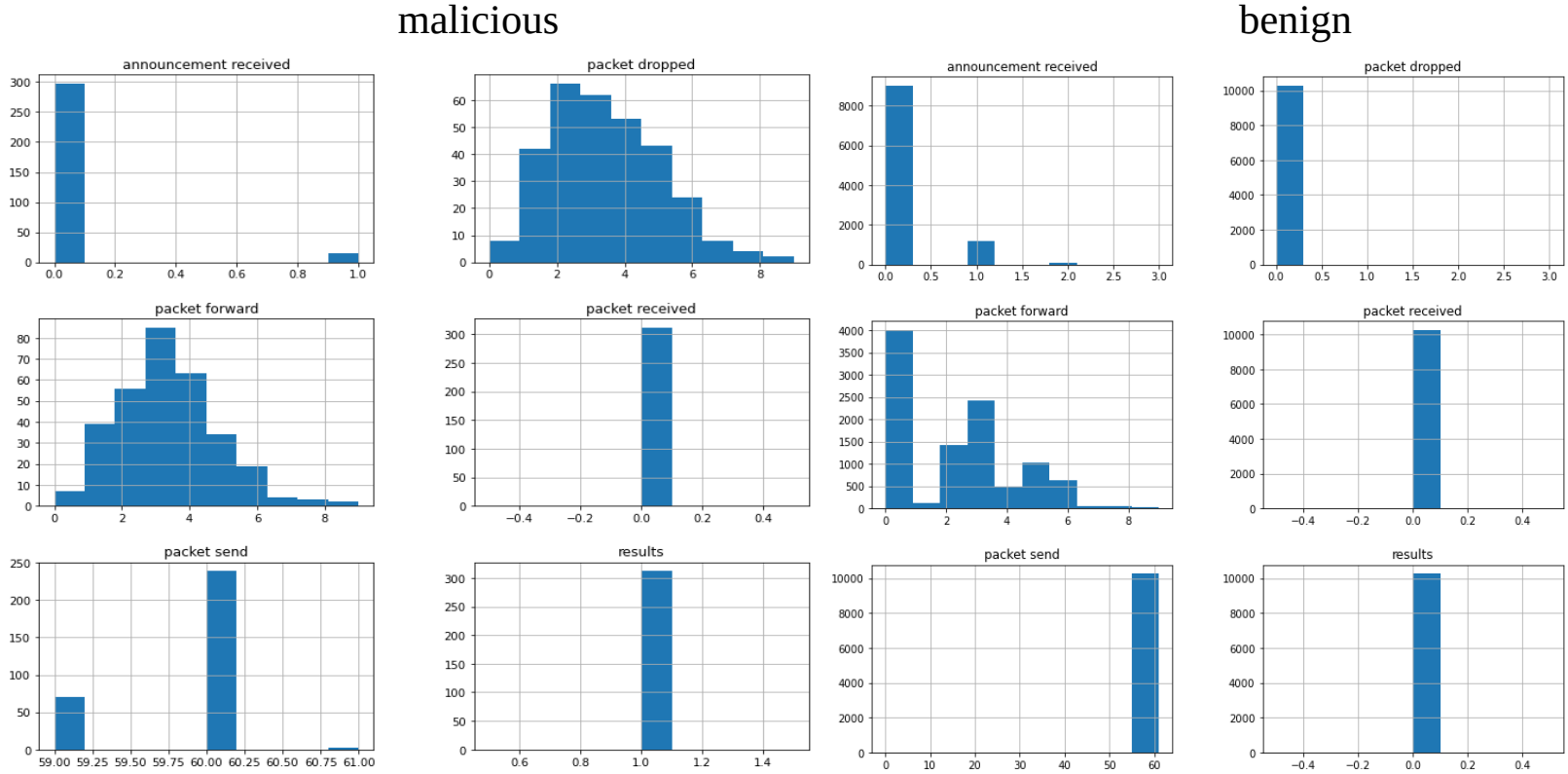
Σχήμα 7.34. BH FR top network FIT results using same model as in middle

Στο Σχήμα 7.34 βλέπουμε τα αποτελέσματα χρησιμοποιώντας το μοντέλο του middle topology.

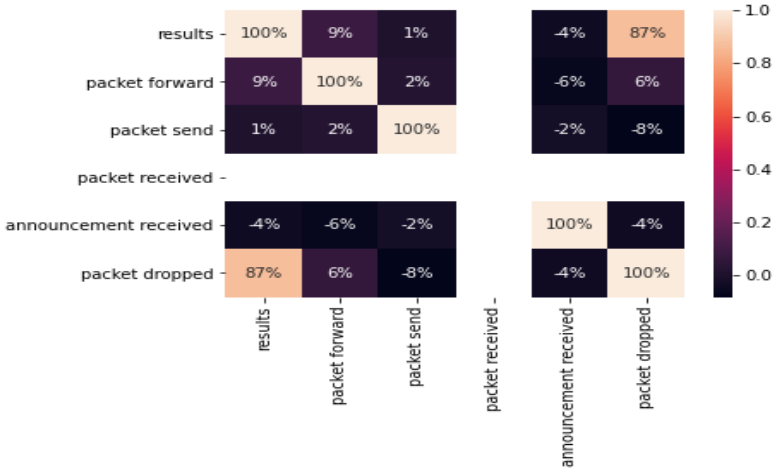
```
classification report
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      2055
           1       0.83      0.78      0.80        63

    accuracy                           0.99      2118
   macro avg       0.91      0.89      0.90      2118
weighted avg       0.99      0.99      0.99      2118

confusion matrix
 [[2045   10]
 [  14   49]]

accuracy score
 0.9886685552407932
```

Σχήμα 7.35. BH FR top topology FIT results

Τα αποτελέσματα χρησιμοποιώντας νέο εκπαιδευμένο μοντέλο βελτιώθηκαν ελάχιστα.

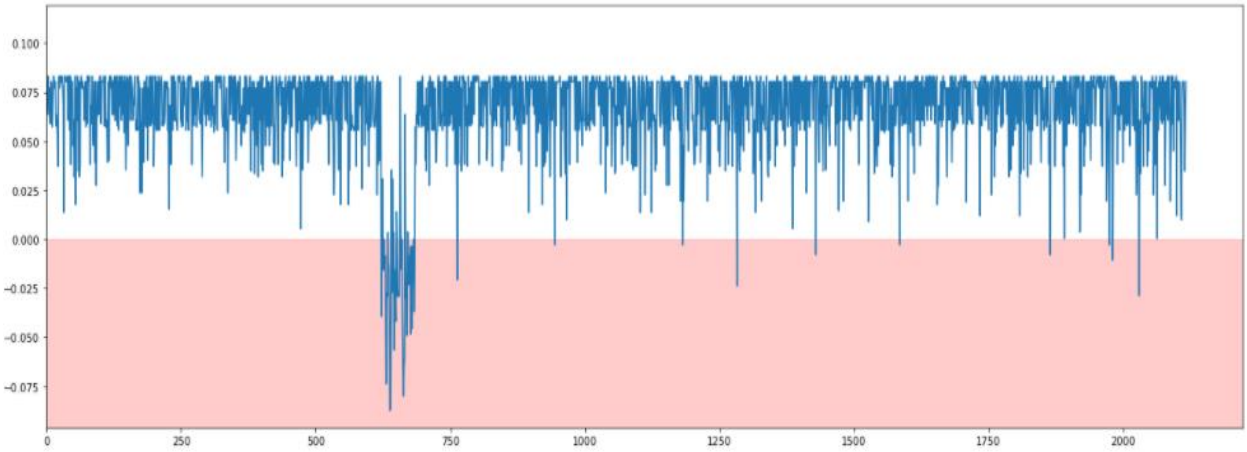


Σχήμα 7.36. Data points values for malicious and benign set



Σχήμα 7.37. Correlation Matrix

Όπως βλέπουμε για μια ακόμη φορά από τα Σχήμα 7.31 και Σχήμα 7.32 βλέπουμε τους ίδιους λόγους που ο αλγόριθμος μας καταφέρνει να εντοπίσει τα malicious data points.



Σχήμα 7.38. Health Plot showing the anomalies

7.3 Selective Forward BN

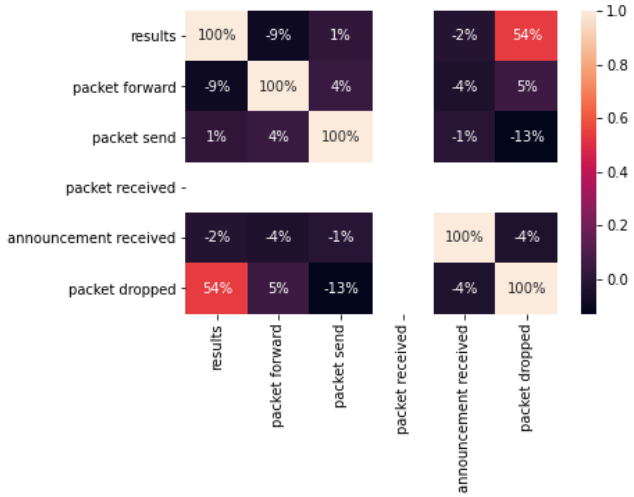
FIT DATA Local Detection

Middle topology

```
classification report
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       622
           1       0.62      0.48      0.54        63

    accuracy                           0.93       685
   macro avg       0.79      0.72      0.75       685
weighted avg       0.92      0.93      0.92       685

confusion matrix
 [[604  18]
 [ 33  30]]

accuracy score
 0.9255474452554745
```

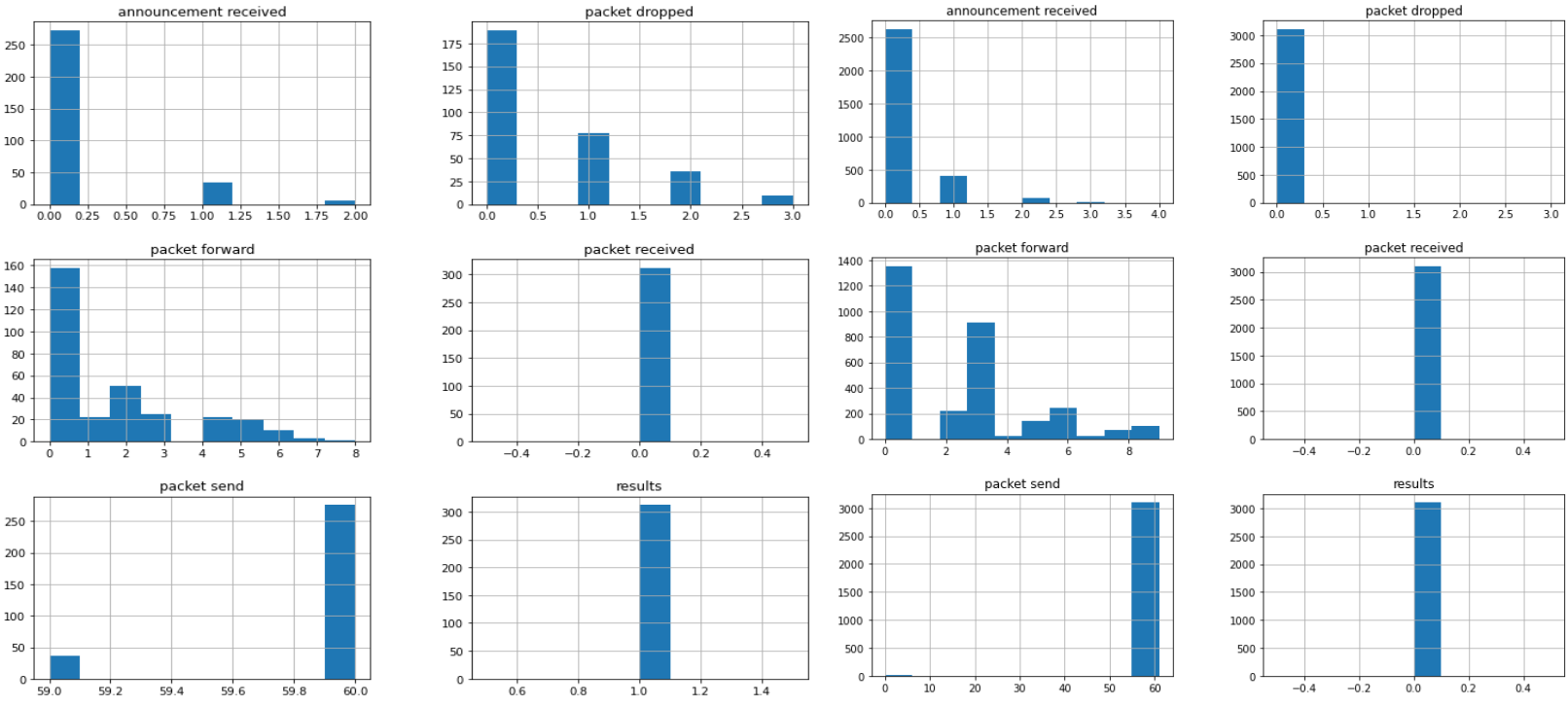
Σχήμα 7.39. SF BN middle local topology FIT results

Παρατηρούμε ότι τα αποτελέσματα που έχουμε για την επίθεση selective forward είναι χειρότερα σε σύγκριση με τα αποτελέσματα εντοπισμού του blackHole. Αν και έχουμε καλό accuracy score 92.5%, βλέπουμε ότι είναι χειρότερο σε σύγκριση με το blackhole, όπως επίσης και το confusion matrix. Υπάρχουν 33 FN και 18 FP predictions.



Σχήμα 7.40. Correlation Matrix

malicious benign



Σχήμα 7.41. Data points values for malicious and benign set

Αρχικά παρατηρώντας τις γραφικές του Σχήμα 7.41 διαπιστώνουμε ότι υπάρχει όπως και στο blackhole διαφορά στις τιμές των features packet dropped και packet forward. Πολύ λογικό αφού αυτή η επίθεση συμπεριφέρεται με παρόμοιο τρόπο όπως και το blackHole. Ρίχνει πακέτα και αλλάζει τα packet forward. Όμως παρατηρούμαι ότι αν και ο sink hole στο packet dropped feature οι τιμές είναι 0, και στον malicious node υπάρχουν πολλές μηδενικές τιμές στο feature packet dropped. Επίσης υπάρχει και ομοιότητα στα packet forward μεταξύ του sink node και του malicious node. Αυτό συμβαίνει γιατί η συγκεκριμένη επίθεση είναι αρκετά ύπουλη. Όπως και το όνομα της επίθεσης επιλέγει πια πακέτα να κάνει forward. Οπόταν γι' αυτό τα αποτελέσματα για αυτό το attack είναι χειρότερα από το blackHole.

Top topology

```
classification report
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       622
           1       0.53      0.41      0.46        63

    accuracy                           0.91       685
   macro avg       0.74      0.69      0.71       685
weighted avg       0.90      0.91      0.91       685

confusion matrix
 [[599  23]
 [ 37  26]]

accuracy score
 0.9124087591240876
```

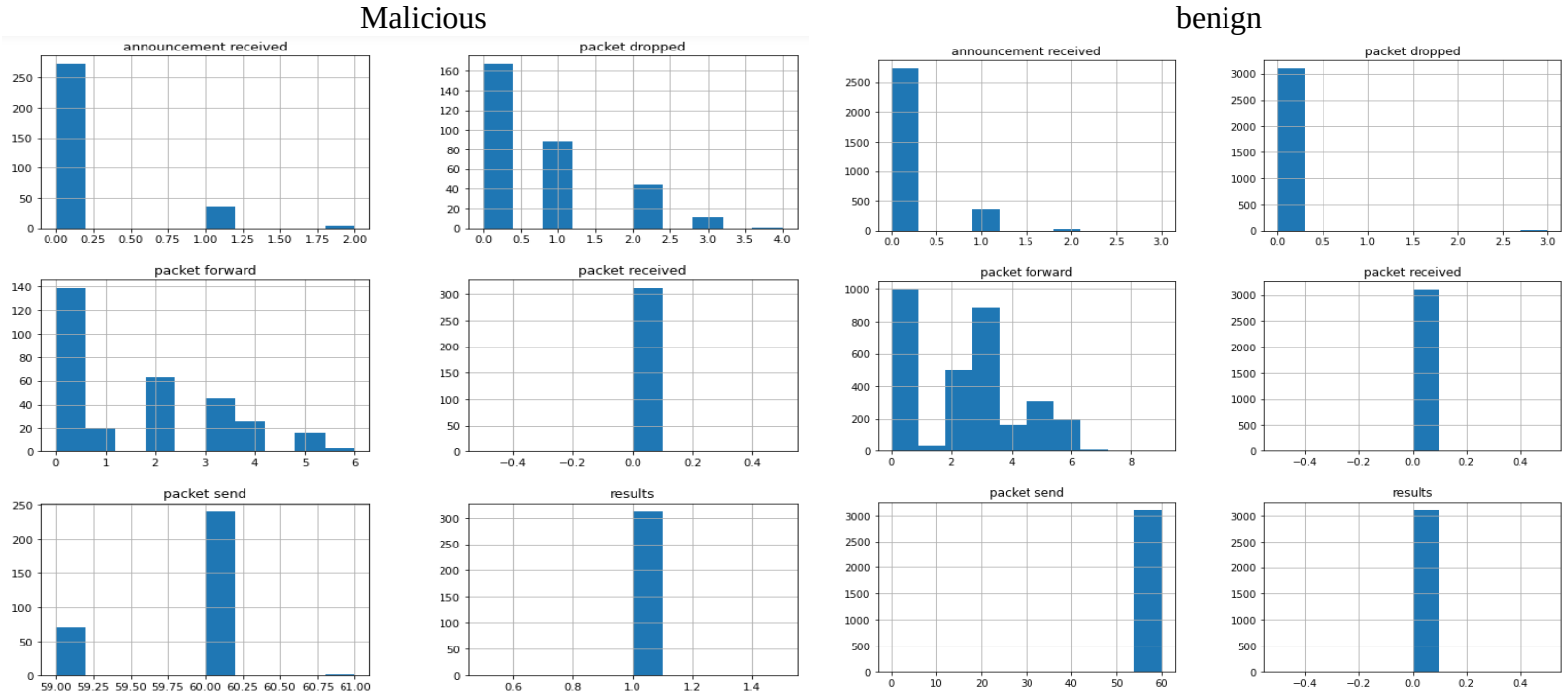
Σχήμα 7.42. SF BN top local topology using middle model FIT results

Ο αλγόριθμος για το top topology se local detection πετυχαίνει accuracy score 91.2% με 37 FN και 23 FP. Τα αποτελέσματα αυτά δεν είναι και τόσο καλά χρησιμοποιώντας το middle model άρα υλοποίησα ξεχωριστό μοντέλο για το top topology.

```
classification report
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       622
           1       0.65      0.44      0.53        63

    accuracy                           0.93       685
   macro avg       0.80      0.71      0.74       685
weighted avg       0.92      0.93      0.92       685

confusion matrix
 [[607  15]
 [ 35  28]]

accuracy score
 0.927007299270073
```

Σχήμα 7.43. SF BN top local topology FIT results

Στο Σχήμα 7.43 βλέπουμε τα αποτελέσματα από το νέο εκπαιδευμένο μοντέλο και παρατηρούμε ότι έχουμε βελτίωση σε σχέση με το έτοιμο μοντέλο. Παρατηρούμε όμως ότι και πάλι σε σχέση με το BlackHole, ο εντοπισμός του selective forward είναι λιγότερο αποτελεσματικός. Οι λόγοι για το top topology είναι ακριβώς οι ίδιοι με αυτούς στο middle και φαίνονται από το Σχήμα 7.44 το οποίο μας δείχνει τις τιμές που παίρνουν τα feature για το malicious node και το sink node και συγκεκριμένα μας ενδιαφέρουν κυρίως τα packet forward και packet dropped.



Σχήμα 7.44. Data points values for malicious and benign set

FIT DATA Network Detection

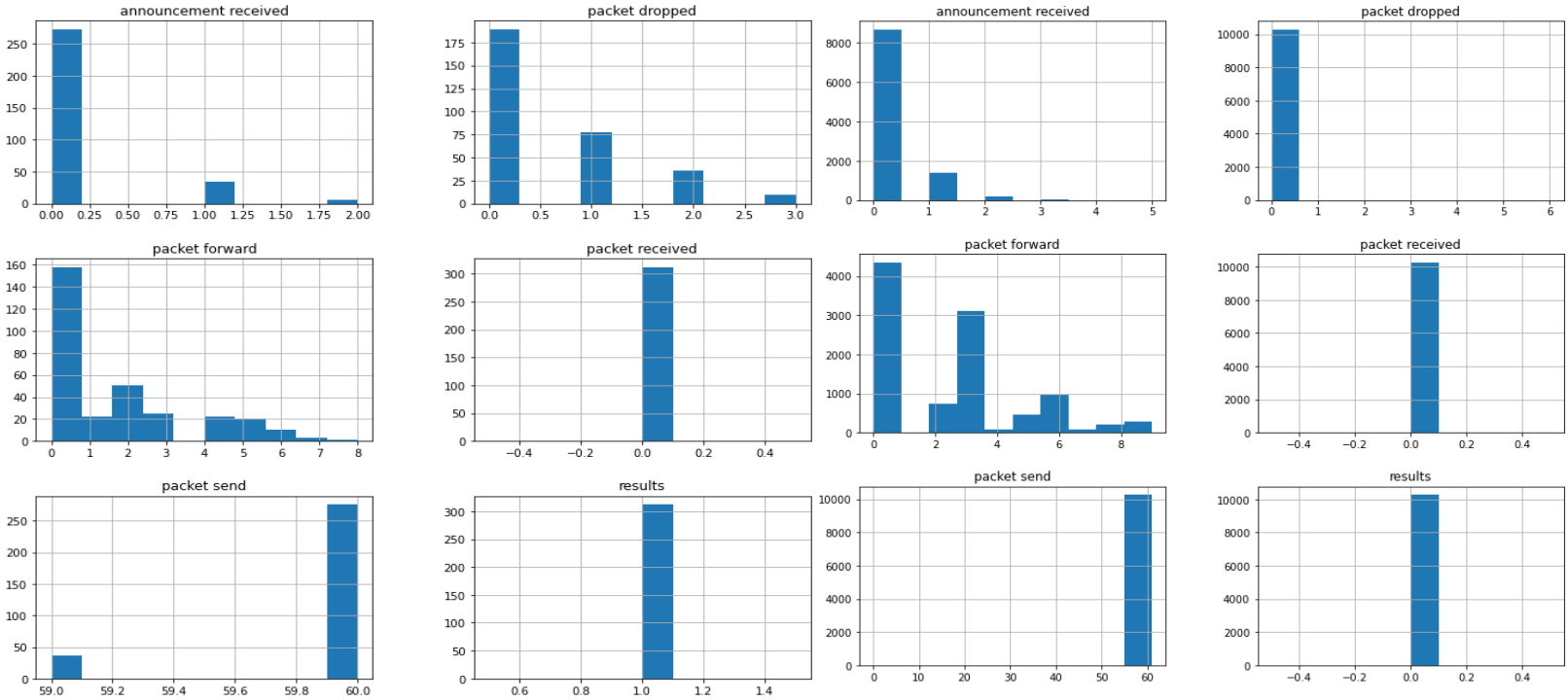
Middle topology

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2056
           1       0.95      0.30      0.46        63

    accuracy                           0.98      2119
   macro avg       0.96      0.65      0.72      2119
weighted avg       0.98      0.98      0.97      2119

confusion matrix
 [[2055    1]
 [  44   19]]

accuracy score
 0.9787635677206229
```

Σχήμα 7.45. SF BN middle network topology FIT results

Προχωρώντας τώρα στο network detection αναμένουμε τα αποτελέσματα να είναι ελαφρώς χειρότερα. Ο λόγος είναι επειδή προσθέτουμε στο data set μας περισσότερα δεδομένα benign (από τα άλλα benign nodes του δικτύου) και έτσι εκτός από το γεγονός ότι τα δεδομένα μας θα γίνουν πιο ανισόρροπα από ότι στο local, όπως είδαμε η συγκεκριμένη επίθεση είναι πιο δύσκολο να εντοπιστεί απ’ ότι η blackHole. Ας δούμε τις τιμές των feature των benign nodes συμπεριλαμβανομένου και του sink node και του malicious node.

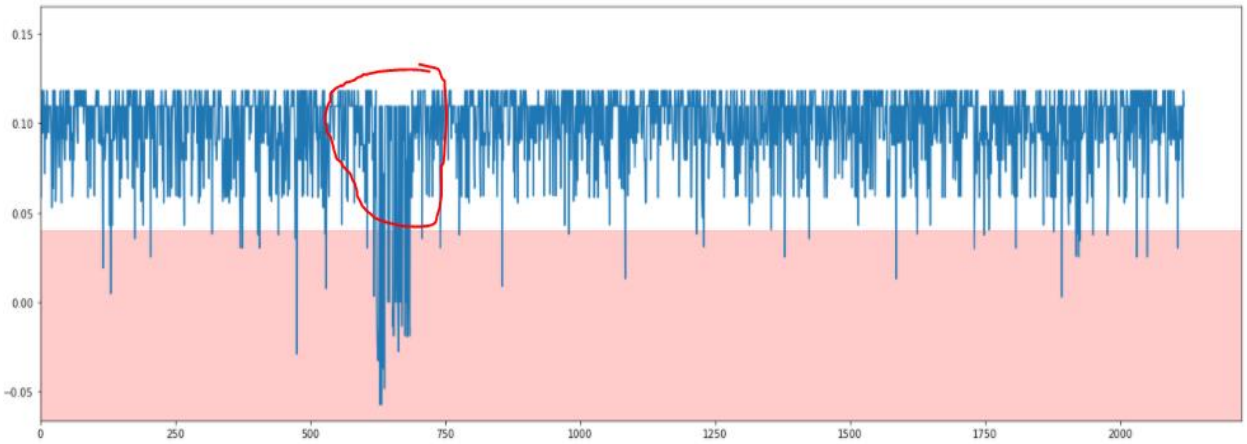
Malicious Benign



Σχήμα 7.46. Data points values for malicious and benign set

Για ακόμη μια φορά παρατηρούμε ότι τα features packet dropped και packet forward είναι αυτά που αποκλίνουν οι τιμές τους μεταξύ των benign nodes και του malicious node. Λόγω του ότι όπως είπαμε η επίθεση είναι ύπουλη και λόγω της μεγαλύτερης ομοιότητας που υπάρχει στις τιμές των features του malicious με των benign nodes και όπως είπαμε της ανισόρροπης κατανομής των δεδομένων μεταξύ benign και malicious τα αποτελέσματα δεν είναι και τόσο καλά αυτή τη φορά. Δυστυχώς υπάρχει αύξηση στα FN όπου ο αλγόριθμος εντοπίζει λανθασμένα 44 data points σαν benign ενώ είναι malicious. Εννοείται ότι είναι και στην φύση του κάθε αλγορίθμου μηχανικής

μάθησης και στο πως δουλεύει ο κάθε αλγόριθμος να επηρεάζει κατά πολύ τα αποτελέσματα. Θα δούμε στη συνέχεια στον SOM ότι τα αποτελέσματα είναι διαφορετικά.



Σχήμα 7.47. Health Plot showing the anomalies

Στο Σχήμα 7.47 βλέπουμε ότι τα anomalies που βρίσκει ο αλγόριθμος δεν είναι και τόσο καλά.

Top topology

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2056
           1       0.81      0.35      0.49        63

    accuracy                           0.98      2119
   macro avg       0.90      0.67      0.74      2119
weighted avg       0.98      0.98      0.97      2119

confusion matrix
 [[2051    5]
 [  41   22]]

accuracy score
 0.9782916470033034
```

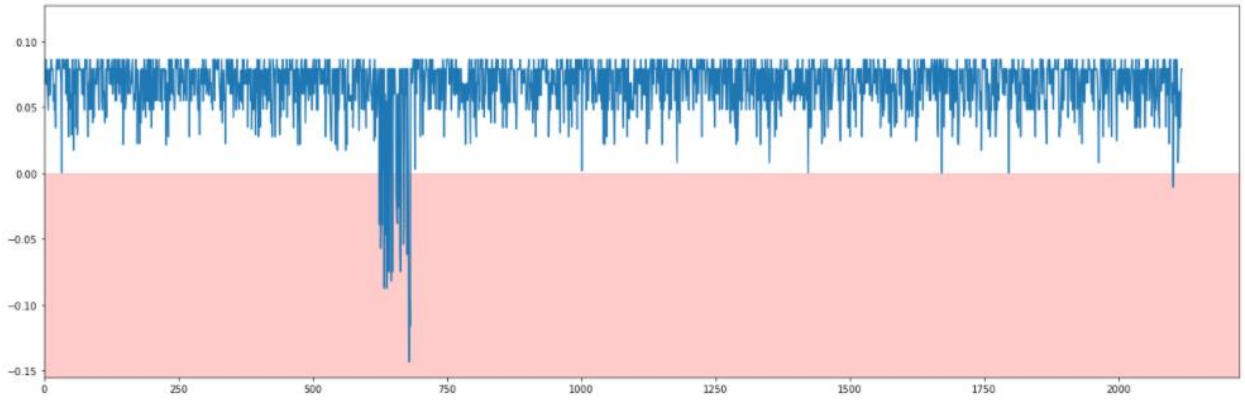
Σχήμα 7.48. SF BN top network topology using middle model FIT results

Στο Σχήμα 7.48 βλέπουμε τα αποτελέσματα για το top topology με την χρήση του μοντέλου στο middle topology. Δεν βλέπουμε κάποια βελτίωση. Στη συνέχεια θα δούμε τα αποτελέσματα από νέο εκπαιδευμένο μοντέλο για το top topology.

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2056
           1       0.97      0.44      0.61        63

    accuracy                           0.98      2119
   macro avg       0.97      0.72      0.80      2119
weighted avg       0.98      0.98      0.98      2119

confusion matrix
 [[2055    1]
 [  35   28]]

accuracy score
 0.9830108541764984
```

Σχήμα 7.49. SF BN top network topology FIT results

Βλέπουμε από το Σχήμα 7.48 βελτίωση στα αποτελέσματα απ' ότι με το μοντέλο του middle topology. Παρόλα αυτά αν και έχουμε πάρα πολύ καλό accuracy score έχουμε ψηλό αριθμό FN. Οι λόγοι είναι οι ίδιοι με τους προηγούμενους.



Σχήμα 7.50. Health Plot showing the anomalies

Εξάλλου ο αριθμός των FN φαίνεται και από το graph του Σχήμα 7.50.

7.4 Selective Forward FR

FIT DATA Local Detection

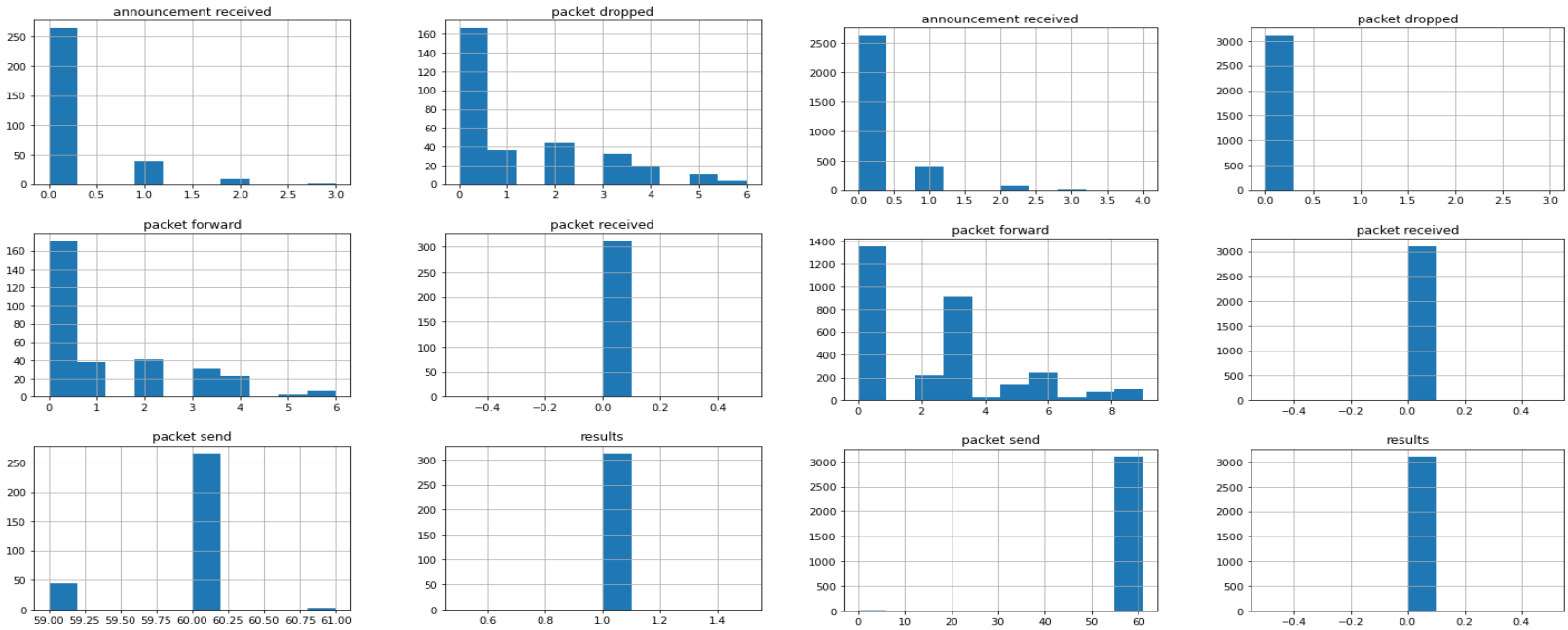
Middle topology

```
classification report
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       622
           1       0.67      0.38      0.48        63

    accuracy                           0.93       685
   macro avg       0.80      0.68      0.72       685
weighted avg       0.91      0.93      0.92       685

confusion matrix
 [[610  12]
 [ 39  24]]

accuracy score
 0.9255474452554745
```

Σχήμα 7.51. SF FR middle local topology FIT results

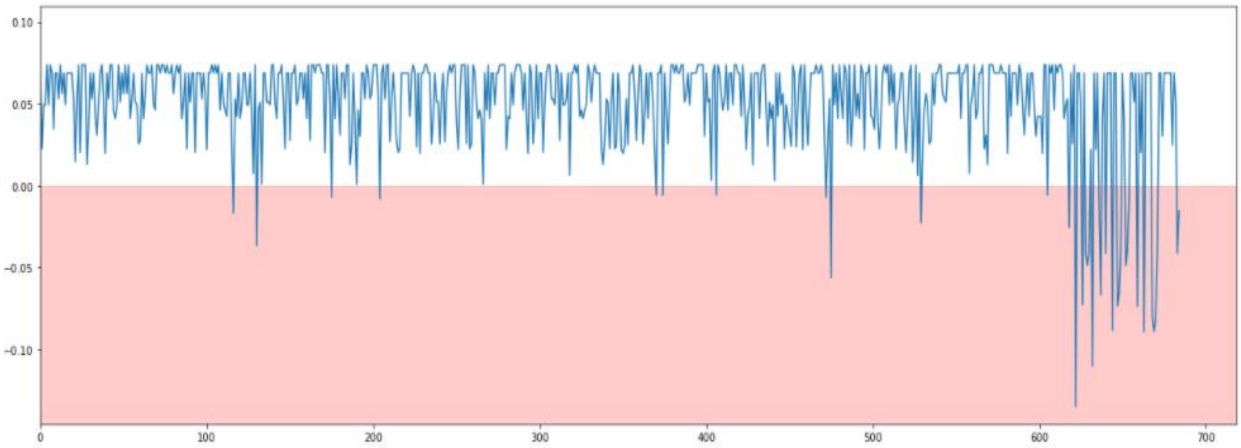
Όπως βλέπουμε από το Σχήμα 7.51 τα αποτελέσματα για το selective forward FR είναι παρόμοια με το selective forward BN. Αναμενόμενο γιατί όπως είχαμε δει το selective forward είναι ένα ύπουλο attack. Ο αλγόριθμος πετυχαίνει 92.5% accuracy score αλλά δυστυχώς έχουμε 39 FN data points.

Malicious benign



Σχήμα 7.52. Data points values for malicious and benign set

Για ακόμη μια φορά παρατηρούμε ότι η διαφορά των τιμών των features packet dropped & packet forward για τον malicious node και benign node είναι ελάχιστη λόγο του ότι το συγκεκριμένο attack είναι ένα ύπουλο attack το οποίο επιλέγει με διάφορους τρόπους τα πακέτα που θα κάνει forward και drop.



Σχήμα 7.53. Health Plot showing the anomalies

Top topology

```
classification report
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       622
           1       0.64      0.67      0.65        63

    accuracy                           0.93       685
   macro avg       0.80      0.81      0.81       685
weighted avg       0.94      0.93      0.93       685

confusion matrix
 [[598  24]
 [ 21  42]]

accuracy score
 0.9343065693430657
```

Σχήμα 7.54. SF FR top local topology using middle model FIT results

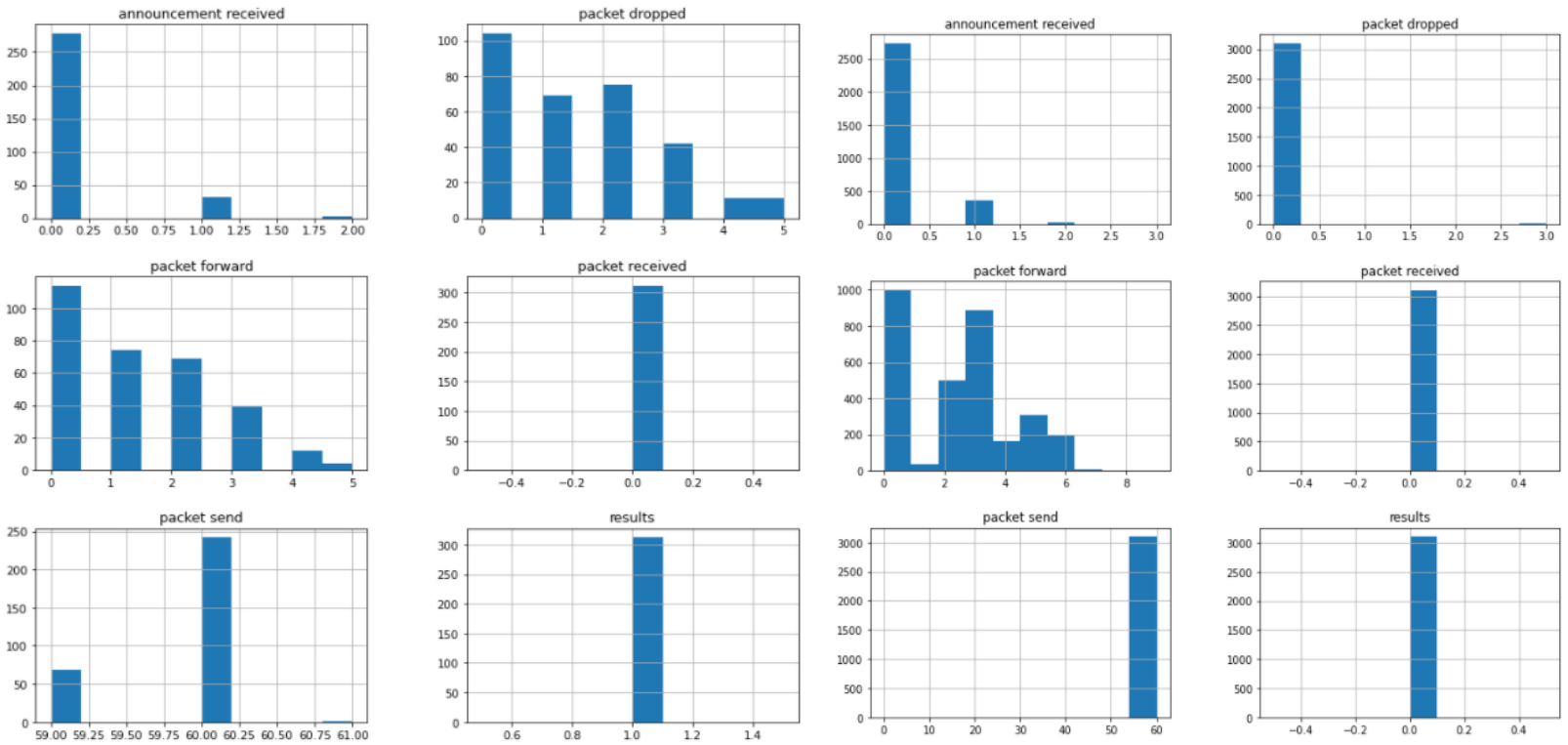
Τα αποτελέσματα χρησιμοποιώντας το middle μοντέλο είναι κάπως καλύτερα πετυχαίνοντας βελτίωση στο accuracy score (93,4%) απ' ότι στο middle topology και επίσης μειώνονται τα FN.

```
classification report
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       622
           1       0.88      0.68      0.77        63

    accuracy                           0.96       685
   macro avg       0.92      0.84      0.87       685
weighted avg       0.96      0.96      0.96       685

confusion matrix
 [[616   6]
 [ 20  43]]

accuracy score
 0.962043795620438
```

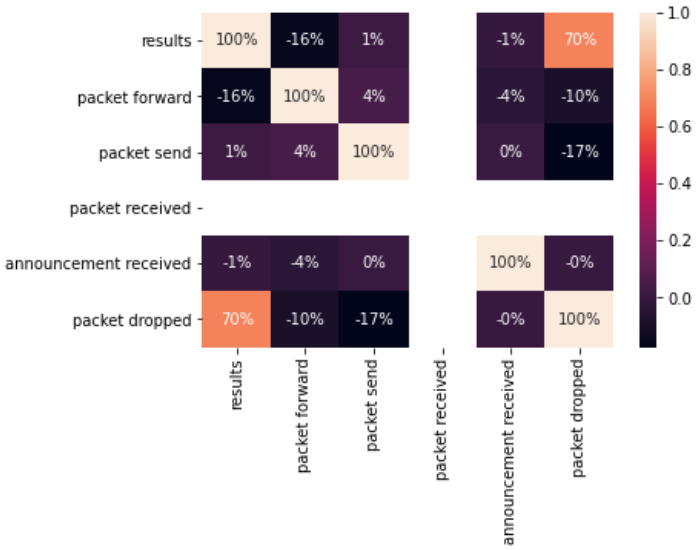
Σχήμα 7.55. SF FR top local topology FIT results

Τα αποτελέσματα είναι καλύτερα με νέο εκπαιδευμένο μοντέλο πετυχαίνοντας 96% accuracy score και βελτιώνοντας κατά πολύ τα FN και FP που είχαμε στο middle topology.

malicious benign



Σχήμα 7.56. Data points values for malicious and benign set



Σχήμα 7.57. Correlation Matrix

Σύμφωνα με τα Σχήμα 7.56 & Σχήμα 7.57 βλέπουμε ότι η αλλαγή στις τιμές του packet dropped στον malicious node και στον benign node αυξήθηκε απ' ότι στο middle και αυτό κάνει ευκολότερο το έργο του αλγόριθμου να βρει πιο αποτελεσματικά και εύκολα τις ανωμαλίες εφόσον εξάλλου αυτή είναι και η δουλειά του συγκεκριμένου αλγορίθμου.

FIT DATA Network Detection

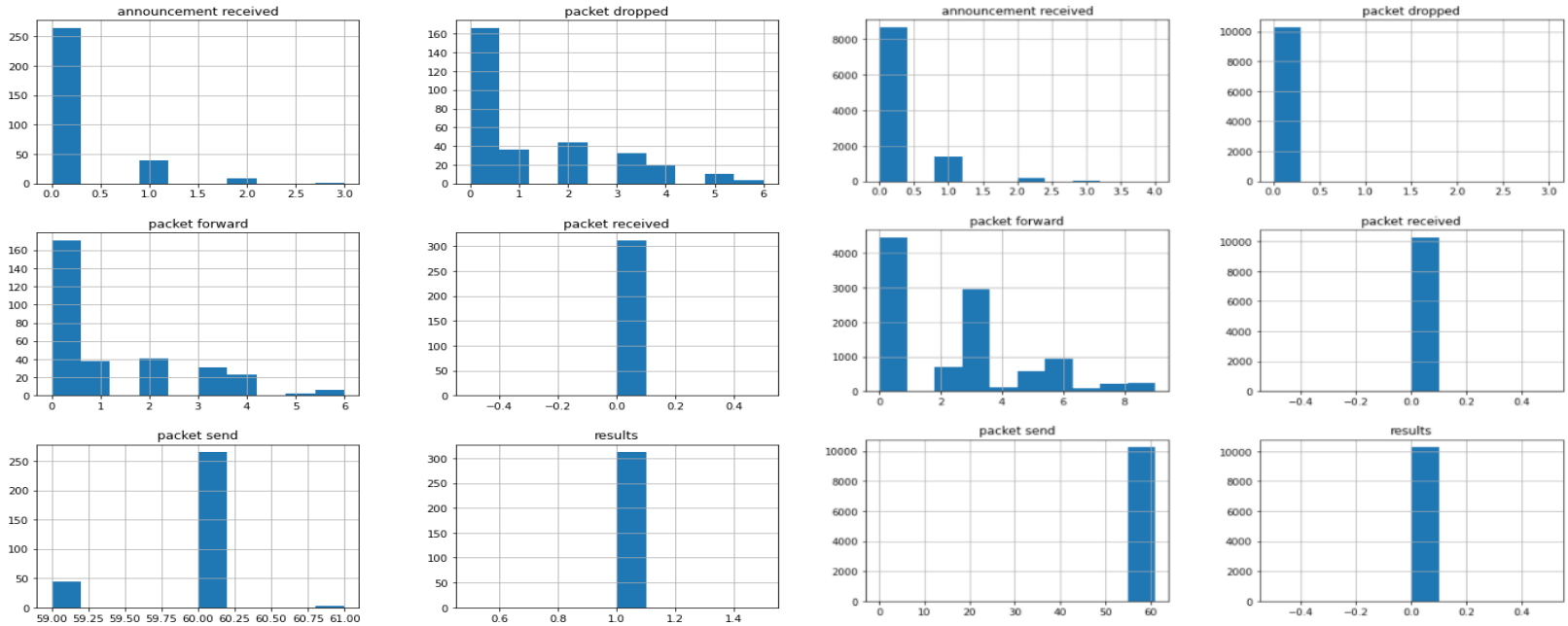
Middle topology

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2055
           1       0.93      0.22      0.36        63

    accuracy                           0.98      2118
   macro avg       0.96      0.61      0.67      2118
weighted avg       0.98      0.98      0.97      2118

confusion matrix
 [[2054    1]
 [  49   14]]

accuracy score
 0.9763928234183191
```

Σχήμα 7.58. SF FR middle network topology FIT results

Προχωρώντας τώρα στο network detection αναμένουμε τα αποτελέσματα να είναι ελαφρώς χειρότερα. Ο λόγος είναι επειδή προσθέτουμε στο data set μας περισσότερα δεδομένα benign (από τα άλλα benign nodes του δικτύου) και έτσι εκτός από το γεγονός ότι τα δεδομένα μας θα γίνουν πιο ανισόρροπα από ότι στο local, όπως είδαμε η συγκεκριμένη επίθεση είναι πιο δύσκολο να εντοπιστεί απ' ότι η blackHole. Ας δούμε τις τιμές των feature των benign nodes συμπεριλαμβανομένου και του sink node και του malicious node.

malicious benign



Σχήμα 7.59. Data points values for malicious and benign set

Όπως και στο local detection στο middle topology δεν έχουμε και την καλύτερη διασπορά των δεδομένων και αυτό αποδεικνύεται και από τα αποτελέσματα.

Top topology

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2057
           1       1.00      0.38      0.55        63

    accuracy                           0.98      2120
   macro avg       0.99      0.69      0.77      2120
weighted avg       0.98      0.98      0.98      2120

confusion matrix
 [[2057    0]
 [  39   24]]

accuracy score
 0.9816037735849057
```

Σχήμα 7.60. SF FR top network topology using middle model FIT results

Χρησιμοποιώντας το μοντέλο του middle topology τα αποτελέσματα που παίρνουμε στο top είναι καλά αλλά δυστυχώς έχουμε 39 FN. Αναμένουμε ότι με την εκπαίδευση νέου μοντέλου για το top network topology, να έχουμε καλύτερα δεδομένα αφού και στο local είδαμε ότι τα δεδομένα του malicious node για τα features packet dropped και packet forward διαφέρουν περισσότερο από τους benign nodes από ότι στο middle topology.

```
classification report
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      2057
           1       0.91      0.62      0.74        63

    accuracy                           0.99      2120
   macro avg       0.95      0.81      0.86      2120
weighted avg       0.99      0.99      0.99      2120

confusion matrix
 [[2053    4]
 [  24   39]]

accuracy score
 0.9867924528301887
```

Σχήμα 7.61. SF FR top network topology FIT results

Όντως τα αποτελέσματα είναι καλύτερα πετυχαίνοντας 98.6% accurate score και 24 FN. Τα δεδομένα είναι σχεδόν τα ίδια με αυτά του local detection απλά τώρα έχουμε περισσότερα benign δεδομένα αφού χρησιμοποιούμε όλους τους benign nodes του δικτύου. Όμως η διαφορά στην διασπορά των τιμών των features packet dropped & packet forward μεταξύ του malicious node και των άλλων nodes είναι μεγαλύτερη από ότι στο middle.

7.5 Sinkhole

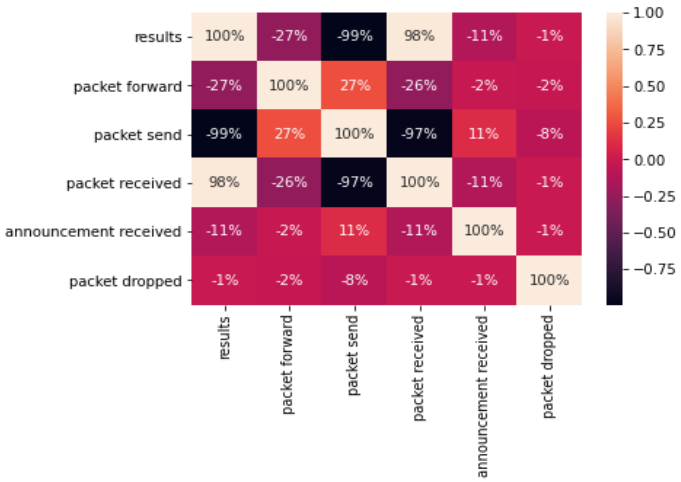
FIT DATA Local Detection

Middle topology

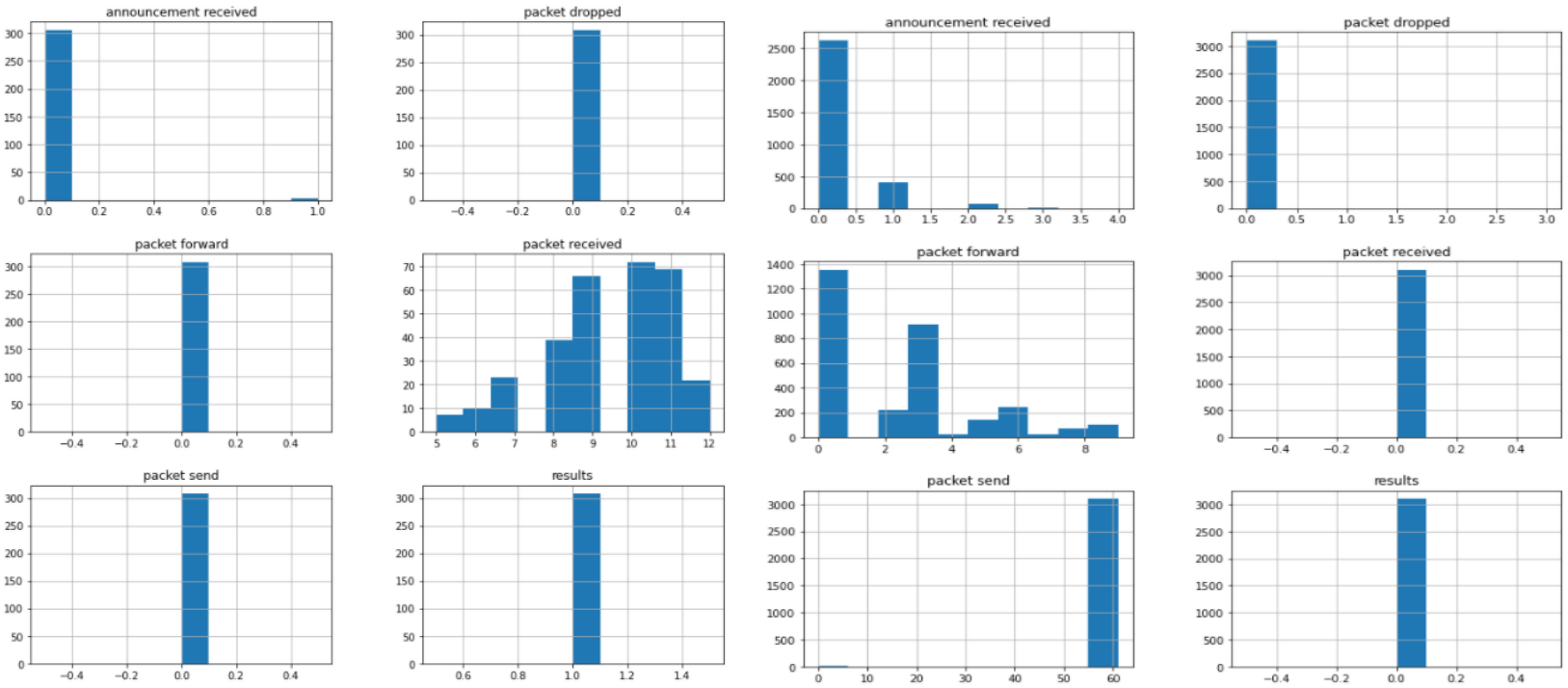
```
classification report
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       622
           1       0.74      0.90      0.81        62

    accuracy                           0.96       684
   macro avg       0.86      0.94      0.90       684
weighted avg       0.97      0.96      0.96       684

confusion matrix
 [[602  20]
 [  6  56]]

accuracy score
 0.9619883040935673
```

Σχήμα 7.62. Sinkhole middle local topology FIT results

Προχωρώντας στην τελευταία επίθεση (sinkhole) παρατηρούμε ότι στο middle topology local detection τα αποτελέσματα είναι αρκετά καλά. Έχουμε ένα accuracy score 94% με 6 FN και 20 FP.

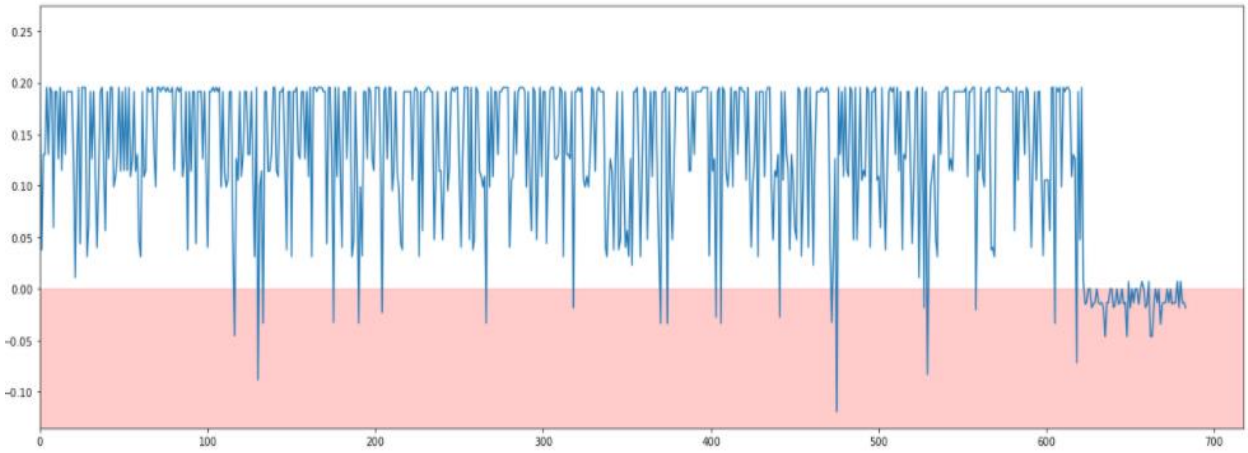


Σχήμα 7.63. Correlation Matrix



Σχήμα 7.64. Data points values for malicious and benign set

Παρατηρούμε από τα Σχήμα 7.63 & Σχήμα 7.64 ότι πλέον δεν έχουμε διαφορές στην διασπορά των τιμών των features packet dropped μεταξύ του malicious node με τους benign nodes. Όπως βλέπουμε στα δύο figures μελετούμε τα features packet received, packet send και ελαφρώς packet forward. Αυτό είναι λογικό εφόσον όπως είδαμε στο κεφάλαιο που αναλύουμε τις επιθέσεις, το sinkhole attack προσπαθεί να συμπεριφερθεί όπως τον sink node. Δηλαδή προσπαθεί να κάνει manipulate τα δεδομένα που λαμβάνει και τα δεδομένα που στέλνει (packet received & packet send). Άρα για αυτό το attack θα μας απασχολήσουν αυτά τα 2 κυρίως features και λιγότερο το packet forward. Στο Σχήμα 7.63 βλέπουμε και το correlation μεταξύ αυτών των features με το result (target variable).



Σχήμα 7.65. Health Plot showing the anomalies

Στο Σχήμα 7.65 παρατηρούμε ότι ο αλγόριθμος συμπεριφέρεται πολύ καλά στο sinkhole βρίσκοντας σχεδόν όλες τις ανωμαλίες (malicious data points).

Top topology

```
classification report
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       622
           1       0.62      0.62      0.62        63

    accuracy                           0.93       685
   macro avg       0.79      0.79      0.79       685
weighted avg       0.93      0.93      0.93       685

confusion matrix
 [[598  24]
 [ 24  39]]

accuracy score
 0.92992700729927
```

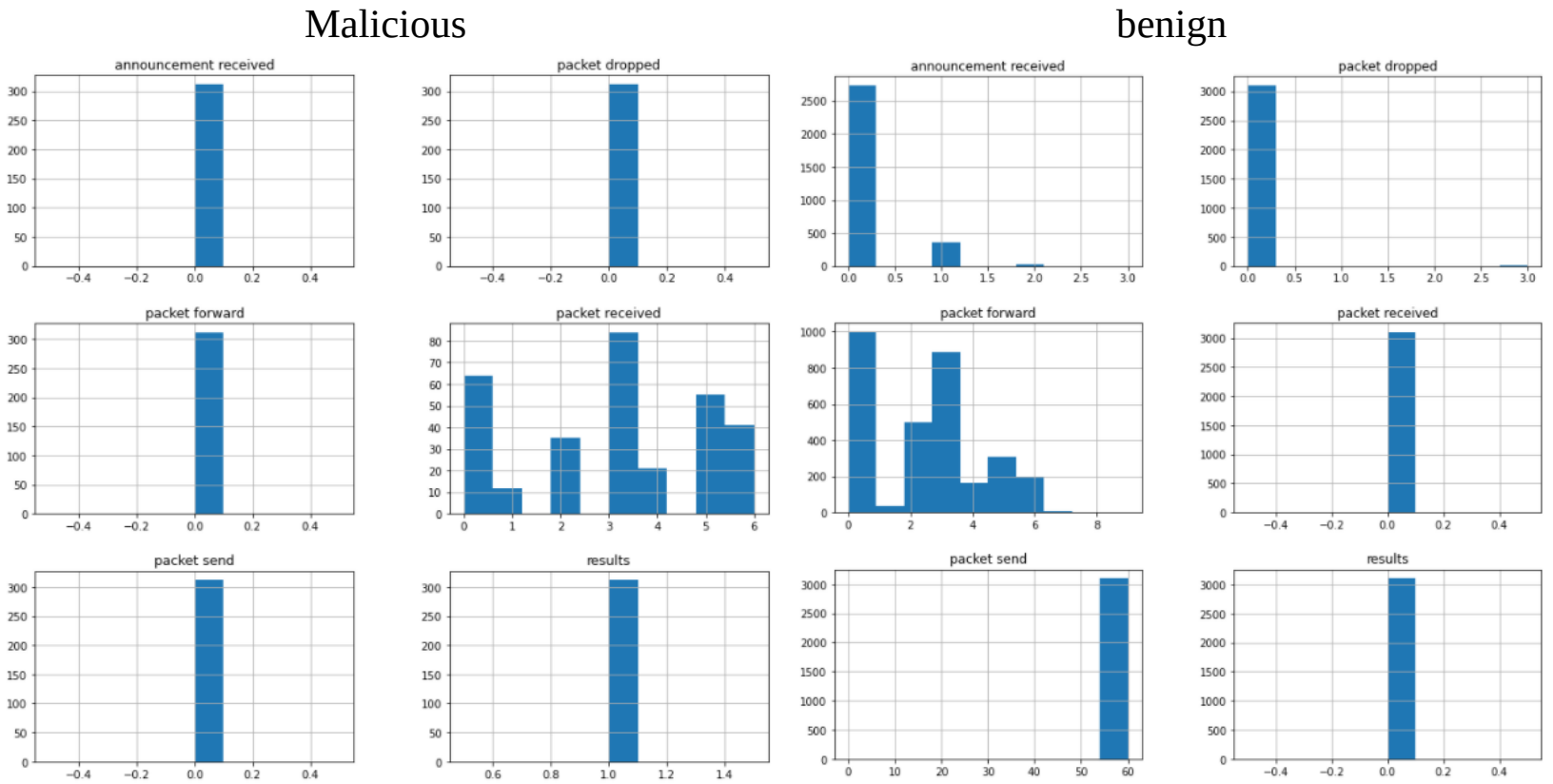
Σχήμα 7.66. Sinkhole top local topology using middle model FIT results

Βλέπουμε χειρότερα αποτελέσματα χρησιμοποιώντας το middle μοντέλο στο top topology. Θα δούμε τα αποτελέσματα ενός νέου εκπαιδευμένου μοντέλου.

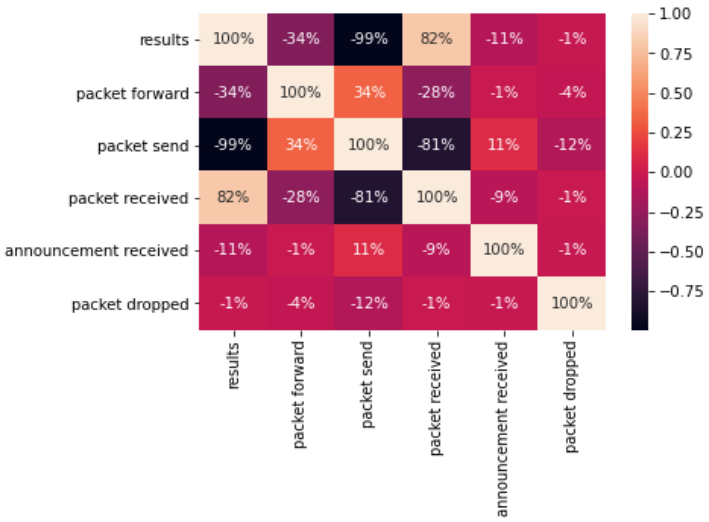
```
classification report
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       622
           1       0.87      0.83      0.85        63

    accuracy                           0.97       685
   macro avg       0.92      0.91      0.92       685
weighted avg       0.97      0.97      0.97       685

confusion matrix
 [[614   8]
 [ 11  52]]

accuracy score
 0.9722627737226277
```

Σχήμα 7.67. Sinkhole top local topology FIT results

Τα αποτελέσματα με το νέο εκπαιδευμένο μοντέλο βελτιώνονται κατά πολύ. Πετυχαίνουμε ένα 97.2% accuracy score. Έχουμε 11 FN data points αλλά δεν παύει να είναι καλό αποτέλεσμα.



Σχήμα 7.68. Data points values for malicious and benign set



Σχήμα 7.69. Correlation Matrix

Από τα Σχήμα 7.69 και 84 παρατηρούμε ότι οι τιμές των τριών features έχουν όπως και στο middle detection αρκετή διαφορά και ψηλό correlation. Αυτό εξηγεί και τα αποτελέσματα που δίνει ο αλγόριθμος.

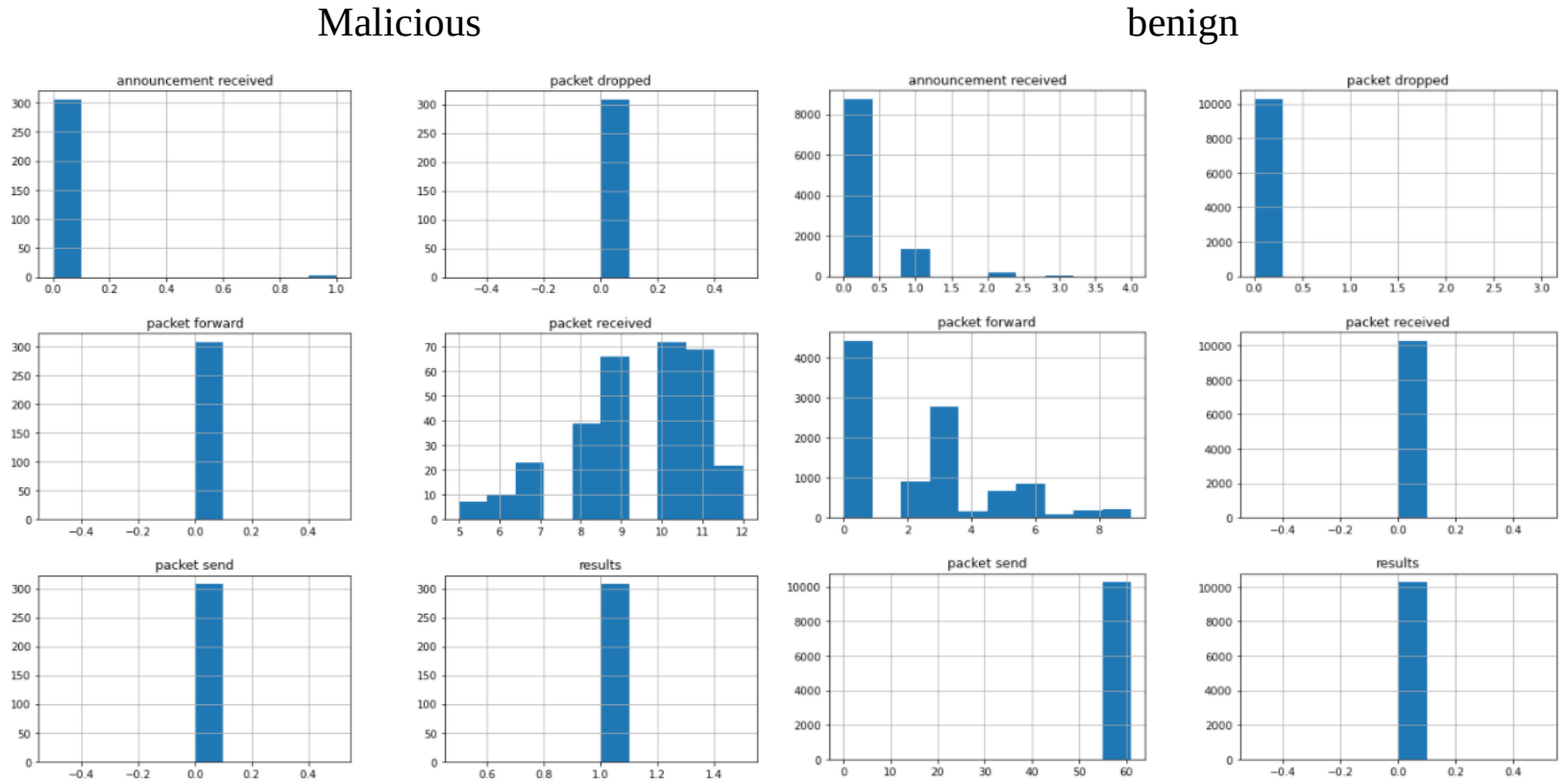
FIT DATA Network Detection

Middle topology

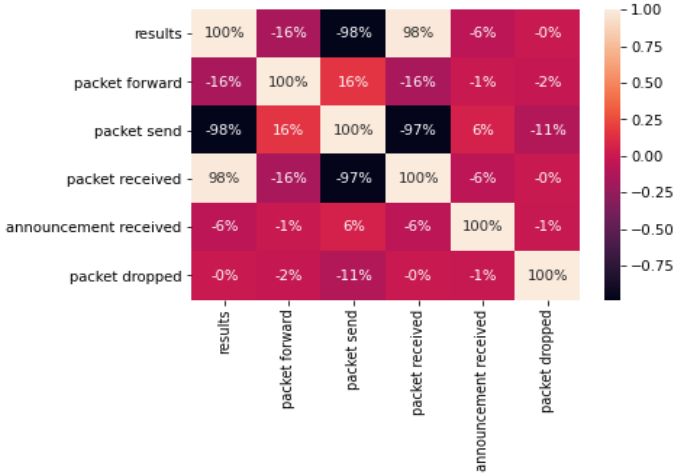
```
classification report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2056
           1       0.94      0.98      0.96        62

    accuracy                           1.00      2118
   macro avg       0.97      0.99      0.98      2118
weighted avg       1.00      1.00      1.00      2118

confusion matrix
 [[2052    4]
 [   1   61]]

accuracy score
 0.997639282341832
```

Σχήμα 7.70. Sinkhole middle network topology FIT results

Τα αποτελέσματα που έχουμε για το middle topology στο network detection είναι σχεδόν απόλυτα σχεδόν 100%. Βλέπουμε 99.7% accuracy score και 1 μόνο FN.

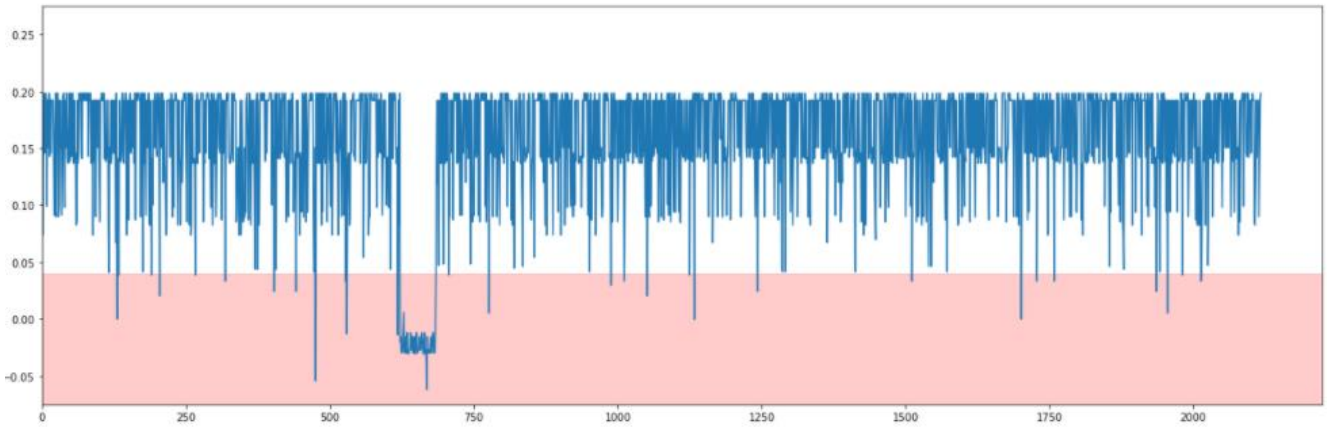


Σχήμα 7.71. Data points values for malicious and benign set



Σχήμα 7.72. Correlation Matrix

Ξεκάθαρα από τα 2 πιο πάνω figures βλέπουμε ότι τα δεδομένα είναι πολύ καλά κατανεμημένα. Η διαφορά στις τιμές των 3 features στον malicious node με τους benign nodes είναι μεγάλη.



Σχήμα 7.73. Health Plot showing the anomalies

Ο Isolation forest βλέπουμε ότι βρίσκει τα anomalies επιτυχώς.

Top topology

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2052
           1       1.00      0.46      0.63        63

    accuracy                           0.98      2115
   macro avg       0.99      0.73      0.81      2115
weighted avg       0.98      0.98      0.98      2115

confusion matrix
 [[2052    0]
 [  34   29]]

accuracy score
 0.9839243498817967
```

Σχήμα 7.74. Sinkhole top network topology using middle FIT results

Όχι και πολύ καλά αποτελέσματα στο top topology χρησιμοποιώντας το middle model. Τα FN αυξάνονται σε 34.

```
classification report
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      2052
           1       1.00      0.62      0.76        63

    accuracy                           0.99      2115
   macro avg       0.99      0.81      0.88      2115
weighted avg       0.99      0.99      0.99      2115

confusion matrix
 [[2052    0]
 [  24   39]]

accuracy score
 0.9886524822695035
```

Σχήμα 7.75. Sinkhole top network topology FIT results

Αν και τα αποτελέσματα βελτιώνονται σε σχέση με το middle trained model έχουμε 24 FN data points. Οι λόγοι είναι ίδιοι και για το middle topology.

7.6 COOJA DATA

COOJA DATA Local Detection

Middle topology

```
classification report
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       576
           1       0.77      0.40      0.52        58

    accuracy                           0.93       634
   macro avg       0.85      0.69      0.74       634
weighted avg       0.93      0.93      0.92       634

confusion matrix
 [[569   7]
 [ 35  23]]

accuracy score
 0.9337539432176656
```

BH-BN middle

```
classification report
              precision    recall  f1-score   support

           0       0.95      0.98      0.97       576
           1       0.72      0.50      0.59        58

    accuracy                           0.94       634
   macro avg       0.84      0.74      0.78       634
weighted avg       0.93      0.94      0.93       634

confusion matrix
 [[565  11]
 [ 29  29]]

accuracy score
 0.9369085173501577
```

BH-FR middle

```
classification report
              precision    recall  f1-score   support

           0       0.91      0.94      0.93       576
           1       0.16      0.10      0.12        58

    accuracy                           0.87       634
   macro avg       0.54      0.52      0.53       634
weighted avg       0.84      0.87      0.85       634

confusion matrix
 [[544  32]
 [ 52   6]]

accuracy score
 0.8675078864353313
```

SF-BN middle

```
classification report
              precision    recall  f1-score   support

           0       0.91      0.95      0.93       576
           1       0.16      0.09      0.11        58

    accuracy                           0.88       634
   macro avg       0.54      0.52      0.52       634
weighted avg       0.84      0.88      0.86       634

confusion matrix
 [[550  26]
 [ 53   5]]

accuracy score
 0.8753943217665615
```

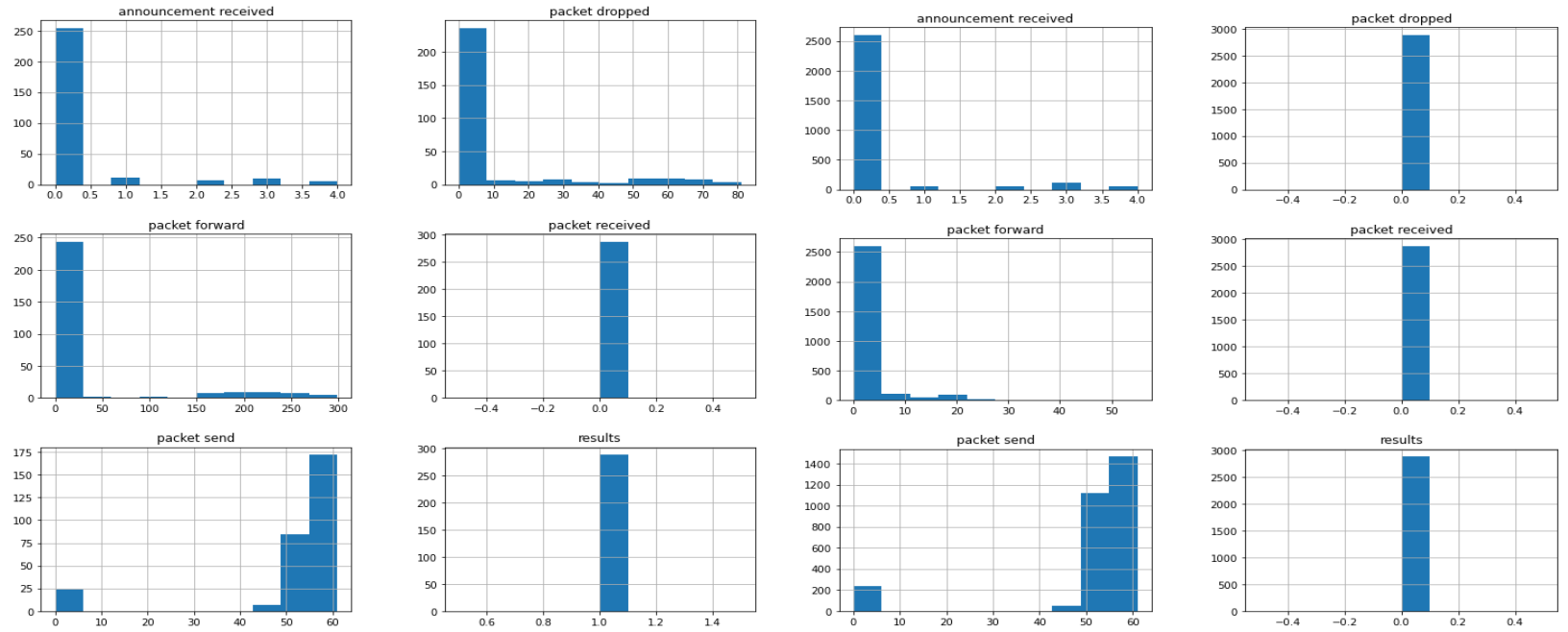
SF-FR middle

```
classification report
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       576
           1       0.41      0.19      0.26        58

    accuracy                           0.90       634
   macro avg       0.66      0.58      0.60       634
weighted avg       0.88      0.90      0.88       634

confusion matrix
 [[560  16]
 [ 47  11]]

accuracy score
 0.9006309148264984
```

Sinkhole middle

Προχωρώντας στα COOJA δεδομένα, θα παρατηρήσουμε αρκετά μεγάλη διαφορά απ' ότι στα FIT δεδομένα για το middle topology. Εξάλλου όπως βλέπουμε και τα αποτελέσματα πιο πάνω δυστυχώς δεν είναι τα αναμενόμενα έχοντας μεγάλη παρακμή από τα FIT δεδομένα από τα αποτελέσματα ήταν πολύ καλά σχεδόν τέλεια. Ο λόγος ήταν εύκολο να εντοπισθεί και είναι πολύ απλός να τον καταλάβουμε. Αυτή η παρακμή συμβαίνει στο middle topology κυρίως καθώς το top και το random topology, παρουσιάζουν παρόμοια και καλά αποτελέσματα όπως και στα FIT DATA.

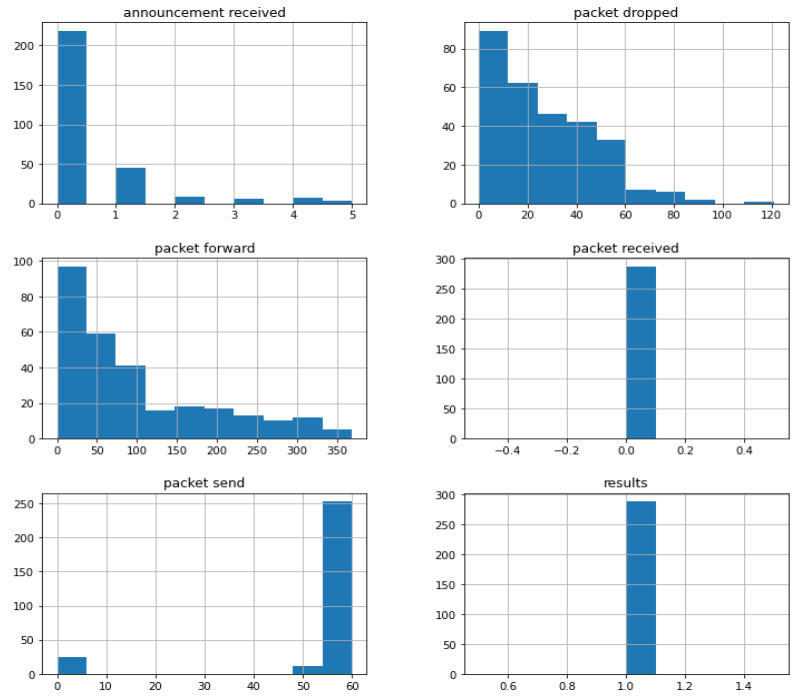
malicious **benign**



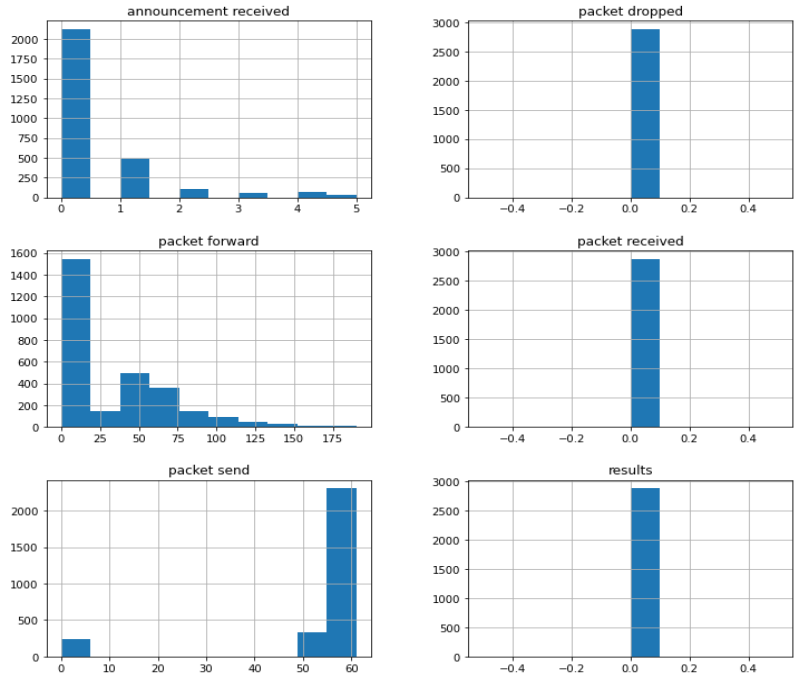
Πιο πάνω μπορούμε ξεκάθαρα να καταλάβουμε τα κακά αποτελέσματα του αλγορίθμου. Τα πάνω δεδομένα είναι για το BH-BN middle topology. Βλέπουμε ότι σε αντίθεση με τα FIT data τα 2 κύρια features που μας ενδιαφέρουν και γενικά όλα τα features του malicious node παίρνουν είναι σχεδόν πανομοιότητα με τα features του benign. Συγκεκριμένα το packet dropped και packet forward που όπως είδαμε στα FIT data μας ενδιαφέρουν περισσότερο σε αυτή την επίθεση, βλέπουμε ότι οι τιμές αυτών των features στον malicious node είναι σχεδόν ίδιες με του sink node. Αυτό καθιστά πολύ δύσκολο έως αδύνατο στον εντοπισμό των επιθέσεων από τον Isolation Forest καθώς δεν θα θεωρεί τα attacks ως anomalies λόγω της ομοιότητάς τους με τα benign. Αυτό που συμπεραίνουμε είναι ότι τα FIT data είναι πολύ καλύτερα και πιο ακριβής καθώς δημιουργούνται από εκτελέσεις σε πραγματικό περιβάλλον σε αντίθεση με τα COOJA που όπως βλέπουμε δεν είναι ξεκάθαρα.

Random topology

Malicious



benign



Προχωρώντας στο random topology παρατηρούμε ότι τα δεδομένα μας είναι όπως πρέπει όπως και στα FIT data. Φαίνεται η διαφορά στις τιμές των features packet forward και packet dropped του malicious node (BH-BN random topology) σε σχέση με τον sink node όπως ακριβώς βλέπαμε και στα FIT data άρα τα αποτελέσματα αναμένονται πολύ καλύτερα.

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       576
           1       0.98      0.81      0.89        58

    accuracy                           0.98       634
   macro avg       0.98      0.90      0.94       634
weighted avg       0.98      0.98      0.98       634

confusion matrix
 [[575   1]
 [ 11  47]]

accuracy score
 0.9810725552050473
```

BH-BN random

```
classification report
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       576
           1       0.94      0.81      0.87        58

    accuracy                           0.98       634
   macro avg       0.96      0.90      0.93       634
weighted avg       0.98      0.98      0.98       634

confusion matrix
 [[573   3]
 [ 11  47]]

accuracy score
 0.9779179810725552
```

BH-FR random

```
classification report
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       576
           1       0.91      0.52      0.66        58

    accuracy                           0.95       634
   macro avg       0.93      0.76      0.82       634
weighted avg       0.95      0.95      0.94       634

confusion matrix
 [[573   3]
 [ 28  30]]

accuracy score
 0.9511041009463722
```

SF-BN random

```
classification report
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       576
           1       0.87      0.57      0.69        58

    accuracy                           0.95       634
   macro avg       0.91      0.78      0.83       634
weighted avg       0.95      0.95      0.95       634

confusion matrix
 [[571   5]
 [ 25  33]]

accuracy score
 0.9526813880126183
```

SF-FR random

```
classification report
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       576
           1       0.60      0.36      0.45        58

    accuracy                           0.92       634
   macro avg       0.77      0.67      0.70       634
weighted avg       0.91      0.92      0.91       634

confusion matrix
 [[562  14]
 [ 37  21]]

accuracy score
 0.919558359621451
```

Sinkhole random

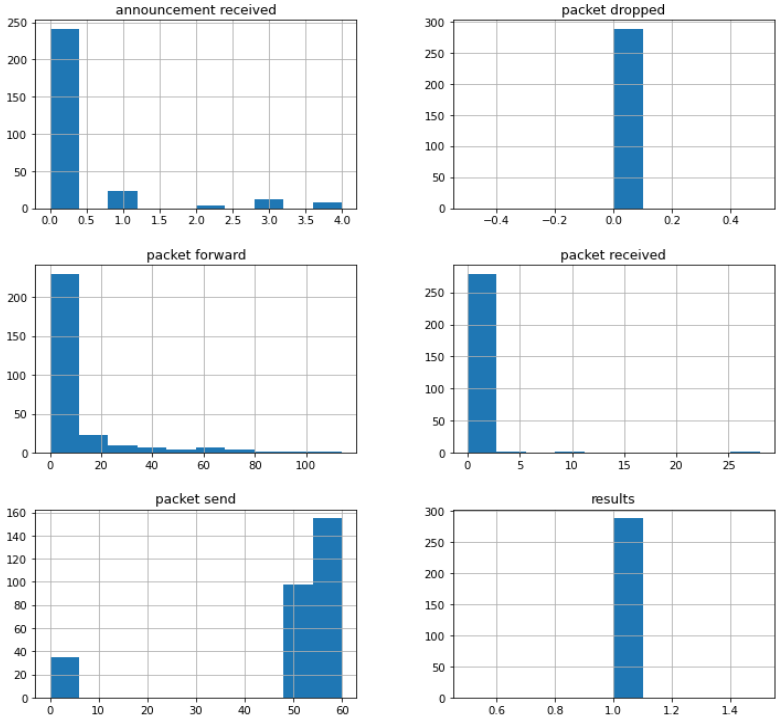
Top topology

Πολύ παρόμοια αποτελέσματα έχουμε και στο top topology όπως και στο random. Όμως στο top topology έχουμε κακά αποτελέσματα για το sinkhole attack.

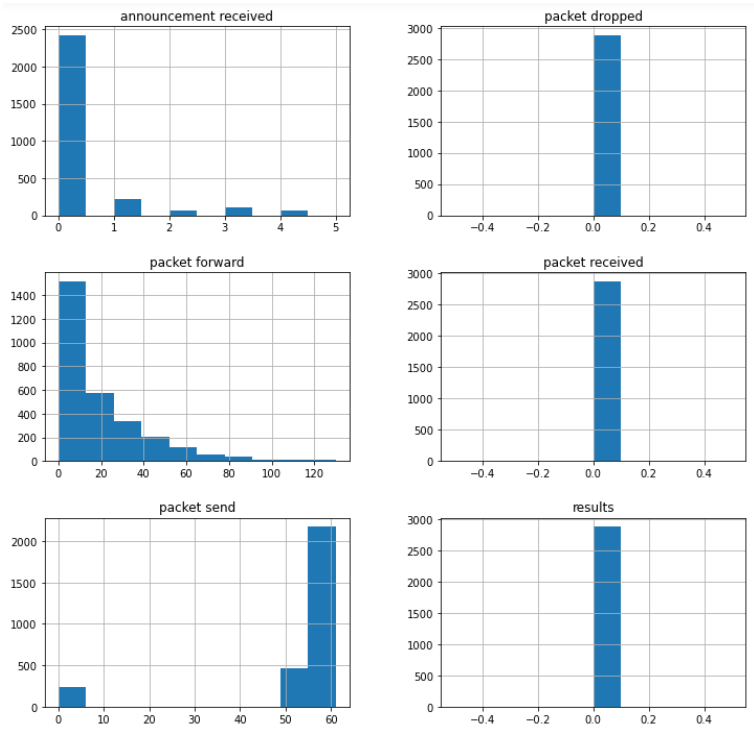
```
classification report
              precision    recall  f1-score   support

           0       0.91      0.94      0.93       576
           1       0.06      0.03      0.04        58

    accuracy                           0.86       634
   macro avg       0.48      0.49      0.48       634
weighted avg       0.83      0.86      0.84       634

confusion matrix
 [[544  32]
 [ 56   2]]

accuracy score
 0.861198738170347
```

Sinkhole attack top

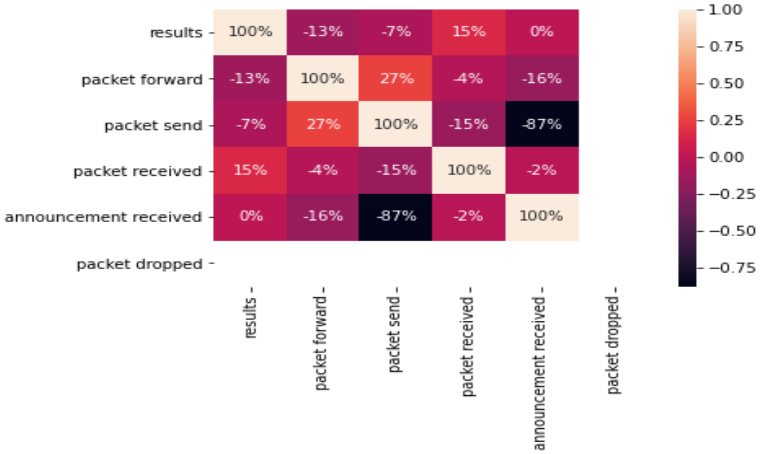
Malicious



benign



Αυτή είναι η διασπορά των τιμών των δεδομένων για τον malicious node και sink node στο sinkhole attack. Όπως βλέπουμε ο λόγος που έχουμε αυτά τα κακά αποτελέσματα είναι φανερός. Τα δεδομένα για ακόμη μια φορά είναι πανομοιότητα μεταξύ malicious node και sink node.



Στο πιο πάνω heatmap βλέπουμε πόσο πολύ έπεσε το correlation μεταξύ των features και του target variable. Ενώ στα FIT είχαμε 90-98% correlation στα features packet forward, packet send και packet received εδώ έχουμε 15% το μεγαλύτερο.

COOJA DATA Network Detection

Middle topology

```
classification report
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      1959
           1       0.64      0.12      0.20        58

    accuracy                           0.97      2017
   macro avg       0.81      0.56      0.59      2017
weighted avg       0.96      0.97      0.96      2017

confusion matrix
 [[1955    4]
 [  51    7]]

accuracy score
 0.9727317798710957
```

BH-BN middle

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1901
           1       0.91      0.34      0.50        58

    accuracy                           0.98      1959
   macro avg       0.94      0.67      0.74      1959
weighted avg       0.98      0.98      0.98      1959

confusion matrix
 [[1899    2]
 [  38   20]]

accuracy score
 0.9795814190913732
```

BH-FR middle

```
classification report
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1901
           1       0.00      0.00      0.00        58

    accuracy                           0.97      1959
   macro avg       0.49      0.50      0.49      1959
weighted avg       0.94      0.97      0.96      1959

confusion matrix
 [[1901    0]
 [  58    0]]

accuracy score
 0.9703930576824911
```

SF-BN middle

```
classification report
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1901
           1       0.00      0.00      0.00        58

    accuracy                           0.97      1959
   macro avg       0.49      0.50      0.49      1959
weighted avg       0.94      0.97      0.95      1959

confusion matrix
 [[1892    9]
 [  58    0]]

accuracy score
 0.96579887697805
```

SF-FR middle

```
classification report
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      1407
           1       0.00      0.00      0.00        34

    accuracy                           0.97      1441
   macro avg       0.49      0.50      0.49      1441
weighted avg       0.95      0.97      0.96      1441

confusion matrix
 [[1396   11]
 [  34    0]]

accuracy score
 0.9687716863289383
```

Sinkhole middle

Παρατηρούμε ότι τα αποτελέσματα στο middle topology για το network Detection στα COOJA data είναι παρόμοια και ακόμη χειρότερα απ' ότι στο local Detection. Οι λόγοι είναι σαφώς οι ίδιοι. Είναι ακόμη χειρότερα για το λόγο ότι στο network detection προστίθενται πολύ περισσότερα benign data points από όλους τους κόμβους του δικτύου και από την στιγμή που τα benign data είναι πολύ όμοια με τα malicious data για όλα τα attacks ο Isolation Forest αδυνατεί να εντοπίσει τις επιθέσεις.

Random topology

```
classification report
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1901
           1       0.95      0.72      0.82        58

    accuracy                           0.99      1959
   macro avg       0.97      0.86      0.91      1959
weighted avg       0.99      0.99      0.99      1959

confusion matrix
 [[1899    2]
 [  16   42]]

accuracy score
 0.9908116385911179
```

BH-BN random

```
classification report
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1901
           1       1.00      0.64      0.78        58

    accuracy                           0.99      1959
   macro avg       0.99      0.82      0.89      1959
weighted avg       0.99      0.99      0.99      1959

confusion matrix
 [[1901    0]
 [  21   37]]

accuracy score
 0.9892802450229708
```

BH-FR random

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1901
           1       0.74      0.45      0.56        58

    accuracy                           0.98      1959
   macro avg       0.86      0.72      0.77      1959
weighted avg       0.98      0.98      0.98      1959

confusion matrix
 [[1892    9]
 [  32   26]]

accuracy score
 0.9790709545686574
```

SF-BN random

```
classification report
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1901
           1       0.78      0.53      0.63        58

    accuracy                           0.98      1959
   macro avg       0.88      0.76      0.81      1959
weighted avg       0.98      0.98      0.98      1959

confusion matrix
 [[1892    9]
 [  27   31]]

accuracy score
 0.9816232771822359
```

SF-FR random

```
classification report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1901
           1       1.00      0.24      0.39        58

    accuracy                           0.98      1959
   macro avg       0.99      0.62      0.69      1959
weighted avg       0.98      0.98      0.97      1959

confusion matrix
 [[1901    0]
 [  44   14]]

accuracy score
 0.9775395610005104
```

Sinkhole random

Όπως και στο local detection βλέπουμε αρκετή βελτίωση στα αποτελέσματα και στο network detection. Οι λόγοι είναι ίδιοι με αυτούς που ανάφερα στο local detection. Στο random topology η διασπορά των τιμών στα features των δεδομένων μεταξύ benign και malicious node αυξάνεται κατά πολύ απ' ότι στο middle topology. Αυτό δημιουργεί μια φυσική διαφορά μεταξύ στα malicious data και benign data και έτσι ο Isolation Forest μπορεί να εντοπίσει αρκετά από τα attacks έχοντας την δυνατότητα να τα κάνει isolate.

Top topology

```
classification report
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       576
           1       0.98      0.74      0.84        58

    accuracy                           0.97       634
   macro avg       0.98      0.87      0.91       634
weighted avg       0.97      0.97      0.97       634

confusion matrix
 [[575   1]
 [ 15  43]]

accuracy score
 0.9747634069400631
```

BH-BN top

```
classification report
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       576
           1       1.00      0.69      0.82        58

    accuracy                           0.97       634
   macro avg       0.98      0.84      0.90       634
weighted avg       0.97      0.97      0.97       634

confusion matrix
 [[576   0]
 [ 18  40]]

accuracy score
 0.9716088328075709
```

BH-FR top

```
classification report
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       576
           1       0.79      0.38      0.51        58

    accuracy                           0.93       634
   macro avg       0.86      0.68      0.74       634
weighted avg       0.93      0.93      0.92       634

confusion matrix
 [[570   6]
 [ 36  22]]

accuracy score
 0.9337539432176656
```

SF-BN top

```
classification report
              precision    recall  f1-score   support

           0       0.97      0.98      0.97       576
           1       0.78      0.66      0.71        58

    accuracy                           0.95       634
   macro avg       0.87      0.82      0.84       634
weighted avg       0.95      0.95      0.95       634

confusion matrix
 [[565  11]
 [ 20  38]]

accuracy score
 0.9511041009463722
```

SF-FR top

```
classification report
              precision    recall  f1-score   support

           0       0.91      0.94      0.93       576
           1       0.06      0.03      0.04        58

    accuracy                           0.86       634
   macro avg       0.48      0.49      0.48       634
weighted avg       0.83      0.86      0.84       634

confusion matrix
 [[544  32]
 [ 56   2]]

accuracy score
 0.861198738170347
```

Sinkhole top

Παρόμοια ξανά τα αποτελέσματα και αναμενόμενα όπως και στο local top topology detection. Δυστυχώς για ακόμη μια φορά βλέπουμε πολύ μεγάλη παρακμή στα αποτελέσματα εντοπισμού του sinkhole attack. Ξανά ο λόγος είναι ο ίδιος με αυτό στο local detection.

Κεφάλαιο 8

Αποτελέσματα Self Organizing Map (SOM)

Ολοκληρώνοντας με τα αποτελέσματα του Isolation Forest θα προχωρήσουμε στον επόμενο αλγόριθμο SOM για να δούμε και να αναλύσουμε την συμπεριφορά του στον εντοπισμό των 5 διαφορετικών επιθέσεων.

Να σημειώσουμε ότι χρησιμοποιούμε τα ίδια ακριβώς δεδομένα που χρησιμοποιήσαμε και στον Isolation Forest άρα οι γραφικές παραστάσεις με την διασπορά των τιμών των features για τα benign και malicious nodes όπως επίσης και τα correlation matrix είναι ακριβώς οι ίδιες άρα δεν θα επαναληφθούν σε αυτό τον αλγόριθμο.

8.1 BlackHole BN

FIT DATA Local Detection

Middle topology

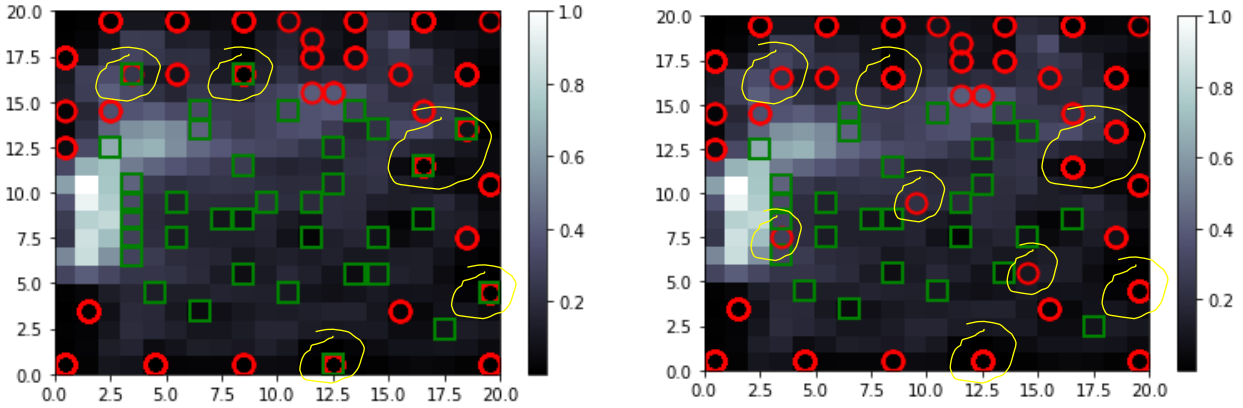
	precision	recall	f1-score	support
0	0.99	1.00	0.99	622
1	1.00	0.86	0.92	63
accuracy			0.99	685
macro avg	0.99	0.93	0.96	685
weighted avg	0.99	0.99	0.99	685

0.9868613138686131
[[622 0]
[9 54]]

Σχήμα 8.1. blackHole middle local topology FIT results

Τα αποτελέσματα για το blackhole BN σε local detection στο middle topology είναι εξαιρετικά πετυχαίνοντας 98.6% accuracy score με μόνο 9 FN data points. Πολύ καλύτερο από τον Isolation Forest.

Ας δούμε τα classifications που κάνει ο αλγόριθμος. Για όλα τα attacks θα μελετάμε τα classifications που έκανε για τα Test δεδομένα (X_test, Y_test) και θα τα συγκρίνουμε με τα classifications που έκανε για το X_test με τα predicted Y_test.



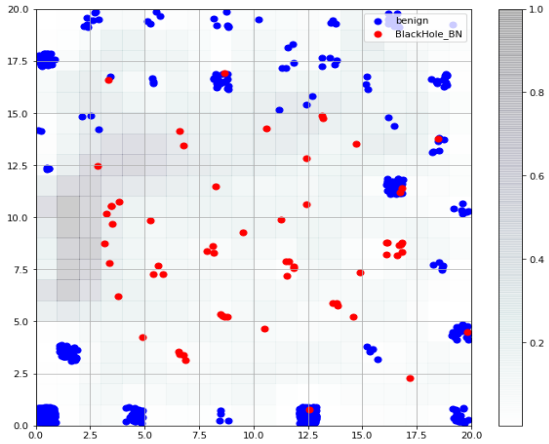
Σχήμα 8.2. Classifications of predicted Data

Ας μελετήσουμε το Σχήμα 8.2. Αυτά που θα αναφέρω ισχύουν για όλα τα classification graphs που θα δούμε στην συνέχεια. Στο αριστερό graph βλέπουμε τα classes που δίνει στα data points ο εκπαιδευμένος αλγόριθμος σε σχέση με τα actual classes που θα έπρεπε να είχε (Y_test). Αρχικά τα πράσινα classes είναι τα malicious ενώ τα κόκκινα είναι τα benign. Εκεί που υπάρχουν

κυκλωμένα classes με κίτρινο χρώμα είναι τα λάθη που κάνει ο αλγόριθμος δηλαδή τα FN. Σύμφωνα με τον εκπαιδευμένο neural network SOM εκείνα τα inputs γίνονται assign στους νευρώνες που είναι κυκλωμένοι δίνοντας τους λανθασμένα class benign ενώ το actual class τους είναι malicious, είναι δηλαδή τα FN. Να σημειώσουμε ότι σε κάθε νευρώνα δεν γίνεται assign ένα data point αλλά το πιο πιθανό πολλά. Τα όμοια data points Θα γίνουν assign στον ίδιο νευρώνα αφού έτσι δουλεύει ο SOM, βρίσκοντας την μικρότερη απόσταση μεταξύ του input από τον κοντινότερο νευρώνα. Άρα αν τα δεδομένα μας όπως είδαμε και στον Isolation Forest, δηλαδή αν τα malicious data είναι όμοια με τα benign data τότε θα γίνονται assign στον ίδιο ή σε παρόμοιο νευρώνα και θα τους γίνεται assign το ίδιο class, δηλαδή είτε benign είτε malicious. Έτσι θα προκύπτουν τα FP και FN. Και δυστυχώς επειδή στα δεδομένα μας έχουμε πολύ περισσότερα benign data points απ' ότι malicious θα έχουμε αρκετά FN.

Στο δεξιά graph τώρα βλέπουμε τα αποτελέσματα που μας δίνει ο αλγόριθμος πάνω στα test data για να δούμε τα FN. Τα κυκλωμένα neurons είναι τα FN. Βλέπουμε ότι ο αλγόριθμος τα κάνει assign σαν benign ενώ η πραγματική τους τιμή όπως βλέπουμε στο αριστερά graph είναι malicious.

Αυτά που ανάφερα πιο πάνω θα ισχύουν για όλα τα attacks που έχουμε να μελετήσουμε.



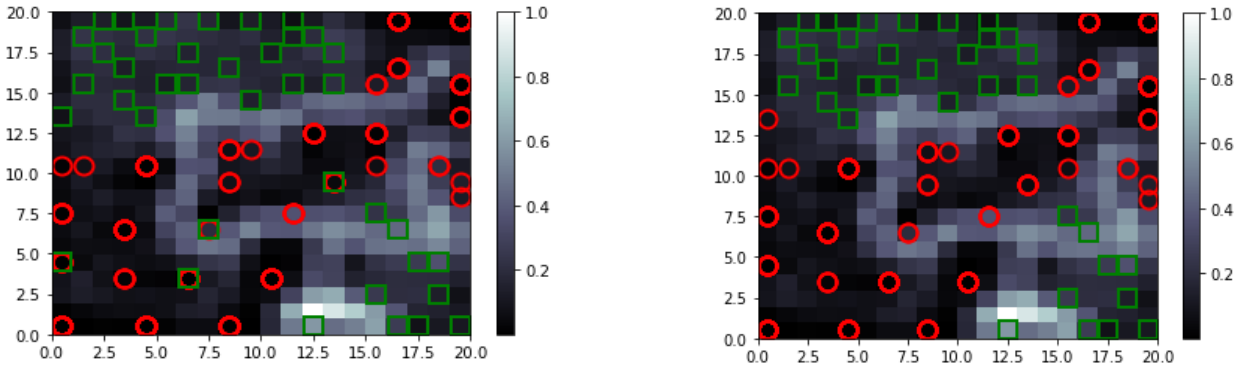
Σχήμα 8.3. Better classification graph

Στο Σχήμα 8.3 απλά βλέπουμε ένα πιο clear classification του αριστερού graph στο Σχήμα 8.2.

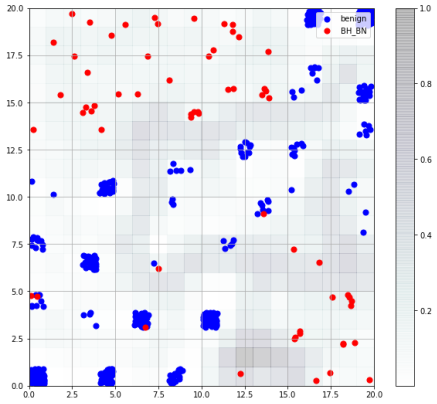
Τώρα οι λόγοι για τα αποτελέσματα είναι παρόμοιοι με αυτούς στο Isolation Forest. Όπως ανάφερα και πιο πάνω, σε κάθε νευρώνα δεν γίνεται assign ένα data point αλλά το πιο πιθανό πολλά. Τα όμοια data points Θα γίνουν assign στον ίδιο νευρώνα αφού έτσι δουλεύει ο SOM,

βρίσκοντας την μικρότερη απόσταση μεταξύ του input από τον κοντινότερο νευρώνα. Άρα αν τα δεδομένα μας όπως είδαμε και στον Isolation Forest, δηλαδή αν τα malicious data είναι όμοια με τα benign data τότε θα γίνονται assign στον ίδιο ή σε παρόμοιο νευρώνα και θα τους γίνεται assign το ίδιο class, δηλαδή είτε benign είτε malicious. Έτσι θα προκύπτουν τα FP και FN. Και δυστυχώς επειδή στα δεδομένα μας έχουμε πολύ περισσότερα benign data points απ' ότι malicious θα έχουμε αρκετά FN.

Top topology



Σχήμα 8.4. Classifications of predicted Data



Σχήμα 8.5. Better classification graph

Από τα Σχήμα 8.4 & Σχήμα 8.5 παρατηρούμε ότι για το top topology ο αλγόριθμος θα έχει καλύτερα αποτελέσματα από το middle αφού το classification που κάνει είναι πιο καθαρό και πιο ακριβές.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	622
1	1.00	0.89	0.94	62
accuracy			0.99	684
macro avg	0.99	0.94	0.97	684
weighted avg	0.99	0.99	0.99	684

```
0.9897660818713451
[[622   0]
 [  7  55]]
```

Σχήμα 8.6. blackHole top local topology using middle model FIT results

	precision	recall	f1-score	support
0	0.99	1.00	1.00	622
1	1.00	0.90	0.95	62
accuracy			0.99	684
macro avg	1.00	0.95	0.97	684
weighted avg	0.99	0.99	0.99	684

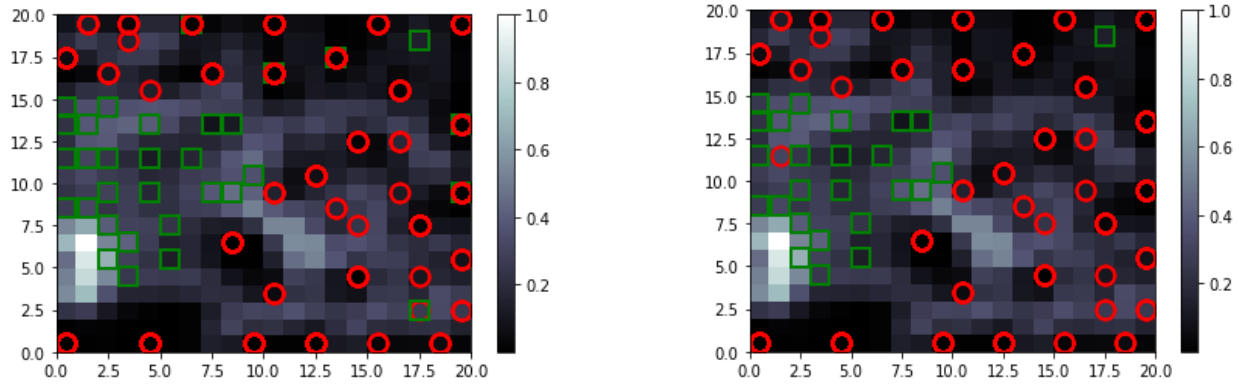
```
0.9912280701754386
[[622   0]
 [  6  56]]
```

Σχήμα 8.7. blackHole top local topology FIT results

Στο Σχήμα 8.6 βλέπουμε τα αποτελέσματα με την χρήση του middle μοντέλου. Είναι καλύτερα από ότι στο middle topology. Στο Σχήμα 8.7 βλέπουμε τα αποτελέσματα από νέο εκπαιδευμένο αλγόριθμο που είναι ακόμη καλύτερα.

FIT DATA Network Detection

Middle topology



Σχήμα 8.8. Classifications of predicted Data

Παρατηρούμε από το Σχήμα 8.8 ότι για ακόμα μια φορά θα έχουμε λογικά πολύ καλά αποτελέσματα.

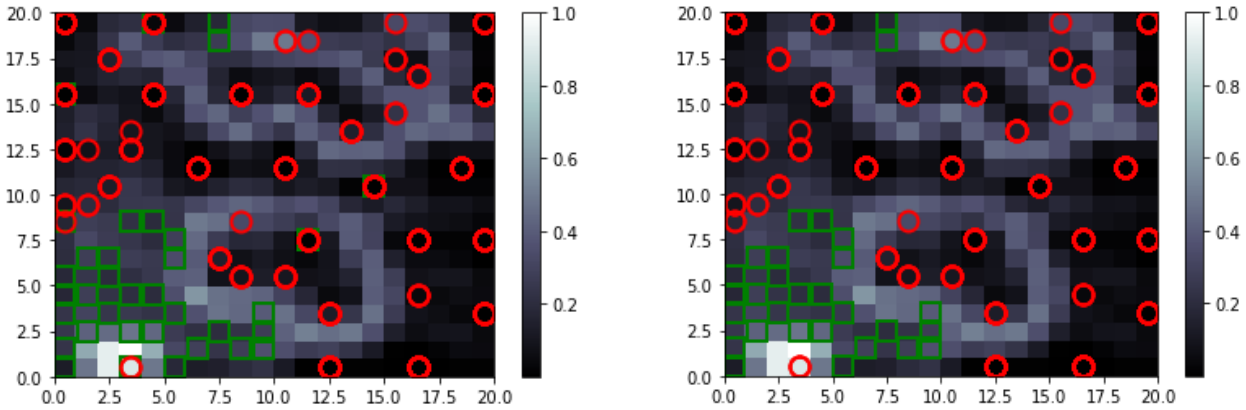
	precision	recall	f1-score	support
0	1.00	1.00	1.00	2057
1	1.00	0.87	0.93	63
accuracy			1.00	2120
macro avg	1.00	0.94	0.97	2120
weighted avg	1.00	1.00	1.00	2120

0.9962264150943396
[[2057 0]
[8 55]]

Σχήμα 8.9. blackHole middle network topology FIT results

Εξαιρετικά αποτελέσματα σχεδόν 100% accuracy score με μόνο 8 FN.

Top topology



Σχήμα 8.10. Classifications of predicted Data

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2054
1	1.00	0.90	0.95	62
accuracy			1.00	2116
macro avg	1.00	0.95	0.97	2116
weighted avg	1.00	1.00	1.00	2116

0.997164461247637
[[2054 0]
[6 56]]

Σχήμα 8.11. blackHole top network topology using middle model FIT results

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2054
1	1.00	0.89	0.94	62
accuracy			1.00	2116
macro avg	1.00	0.94	0.97	2116
weighted avg	1.00	1.00	1.00	2116

0.9966918714555766
[[2054 0]
[7 55]]

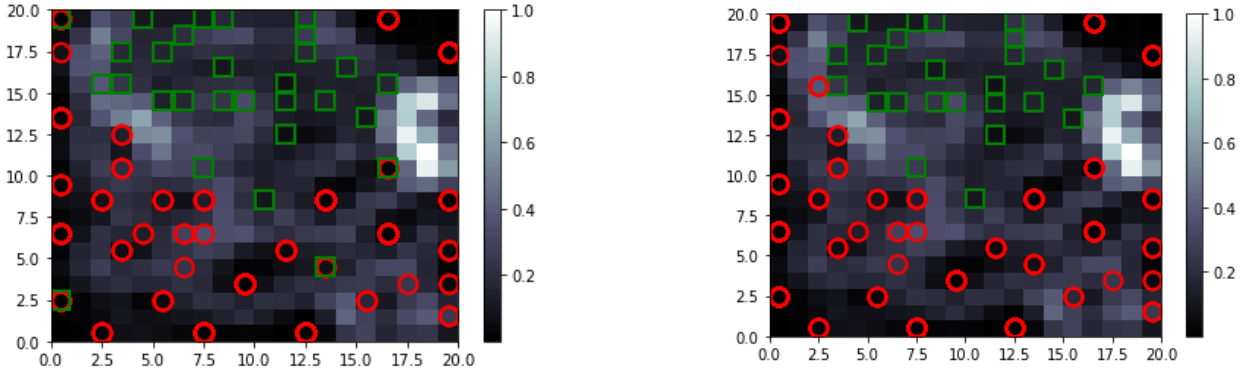
Σχήμα 8.12. blackHole top network topology FIT results

Βλέπουμε ότι και με το middle model και με νέο εκπαιδευμένο μοντέλο έχουμε εξαιρετικά αποτελέσματα στο top topology.

8.2 BlackHole FR

FIT DATA Local Detection

Middle topology



Σχήμα 8.13. Classifications of predicted Data

Από το Σχήμα 8.13 βλέπουμε ότι και για αυτό το blackhole attack θα έχουμε καλά αποτελέσματα.

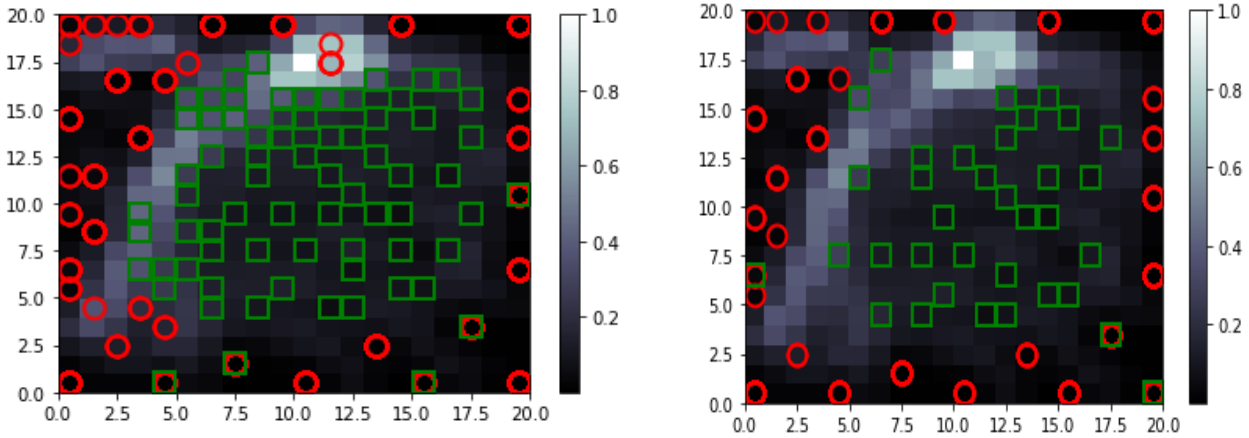
	precision	recall	f1-score	support
0	0.99	1.00	0.99	622
1	1.00	0.89	0.94	62
accuracy			0.99	684
macro avg	0.99	0.94	0.97	684
weighted avg	0.99	0.99	0.99	684

0.9897660818713451
[[622 0]
 [7 55]]

Σχήμα 8.14. blackHole middle local topology FIT results

Για ακόμη μια φορά έχουμε εξαιρετικά αποτελέσματα με 99% accuracy score και μόνο 7 FN.

Top topology



Σχήμα 8.15. Classifications of predicted Data

Από το Σχήμα 8.15 αναμένουμε ξανά πολύ καλά αποτελέσματα.

	precision	recall	f1-score	support
0	0.99	1.00	1.00	622
1	1.00	0.94	0.97	63
accuracy			0.99	685
macro avg	1.00	0.97	0.98	685
weighted avg	0.99	0.99	0.99	685

0.9941605839416059
[[622 0]
 [4 59]]

Σχήμα 8.16. blackHole top local topology using middle model FIT results

	precision	recall	f1-score	support
0	0.99	1.00	1.00	622
1	1.00	0.92	0.96	63
accuracy			0.99	685
macro avg	1.00	0.96	0.98	685
weighted avg	0.99	0.99	0.99	685

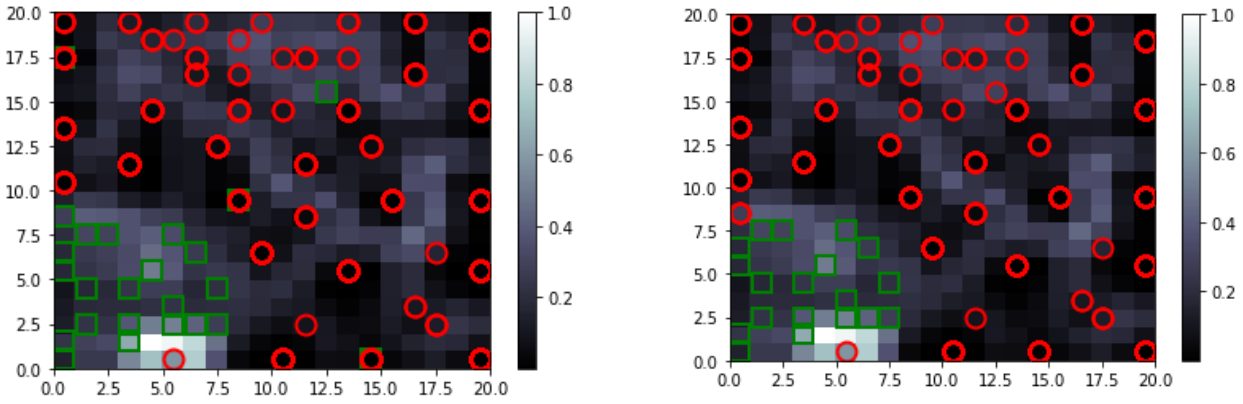
```
0.9927007299270073
[[622   0]
 [  5  58]]
```

Σχήμα 8.17. blackHole top local topology FIT results

Τόσο με το middle model όσο και με νέο μοντέλο έχουμε πολύ καλά αποτελέσματα όπως αναμέναμε.

FIT DATA Network Detection

Middle topology



Σχήμα 8.18. Classifications of predicted Data

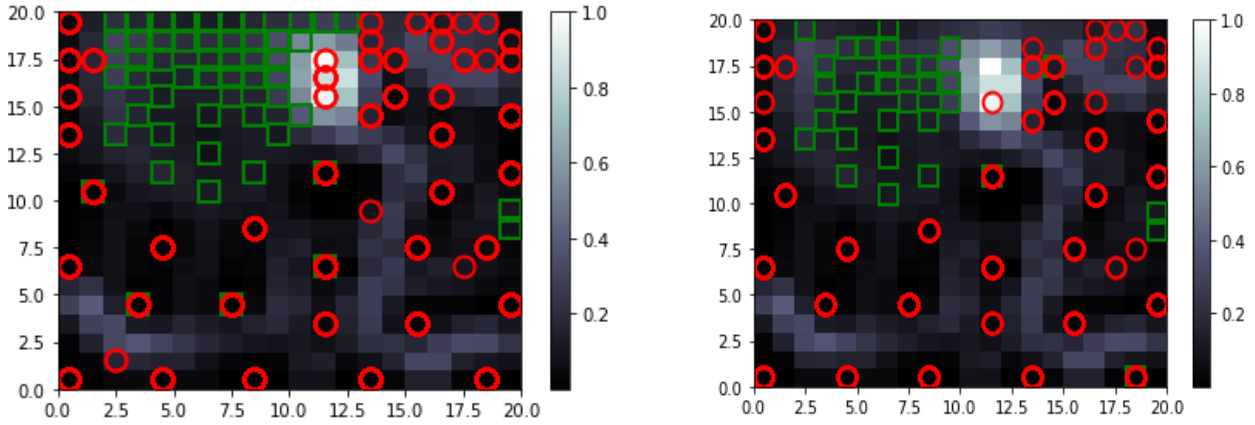
	precision	recall	f1-score	support
0	1.00	1.00	1.00	2054
1	1.00	0.89	0.94	62
accuracy			1.00	2116
macro avg	1.00	0.94	0.97	2116
weighted avg	1.00	1.00	1.00	2116

```
0.9966918714555766
[[2054    0]
 [   7   55]]
```

Σχήμα 8.19. blackHole middle network topology FIT results

Τα αποτελέσματα για το blackhole FR στο middle topology για network detection είναι ξανά σχεδόν τέλια με 99.7% accuracy score και 7 FN.

Top topology



Σχήμα 8.20. Classifications of predicted Data

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2055
1	1.00	0.94	0.97	63
accuracy			1.00	2118
macro avg	1.00	0.97	0.98	2118
weighted avg	1.00	1.00	1.00	2118

```
0.9981114258734656
[[2055    0]
 [   4   59]]
```

Σχήμα 8.21. blackHole top network topology using middle model FIT results

```
                precision    recall  f1-score   support

           0       1.00      1.00      1.00      2055
           1       1.00      0.95      0.98        63

    accuracy                           1.00      2118
   macro avg       1.00      0.98      0.99      2118
weighted avg       1.00      1.00      1.00      2118

0.9985835694050992
[[2055    0]
 [   3   60]]
```

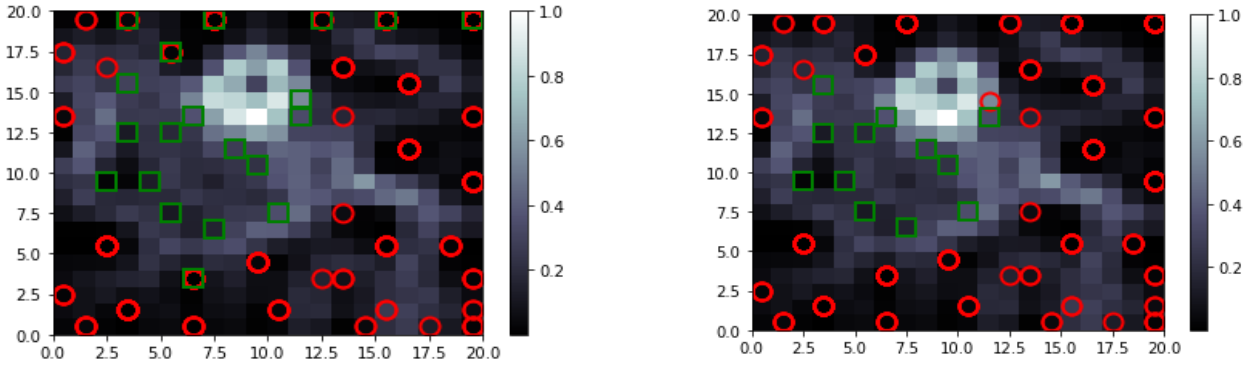
Σχήμα 8.22. blackHole top network topology FIT results

Τόσο με το middle model όσο και με νέο μοντέλο έχουμε πολύ καλά αποτελέσματα όπως αναμέναμε. Μέχρι στιγμής έχουμε το καλύτερο αποτέλεσμα με μόνο 3 FN όπως φαίνεται στο Σχήμα 8.22.

8.3 Selective forward BN

FIT DATA Local Detection

Middle topology



Σχήμα 8.23. Classifications of predicted Data

Παρατηρούμε όπως και αναμένουμε ότι τα αποτελέσματα για το selective forward τόσο για το BN όσο και για το FR δεν θα είναι τόσο καλά όσο τα blackhole. Τους λόγους τους αναλύσαμε και στον Isolation Forest αλλά και στην εισαγωγή αυτού του κεφαλαίου για τον SOM.

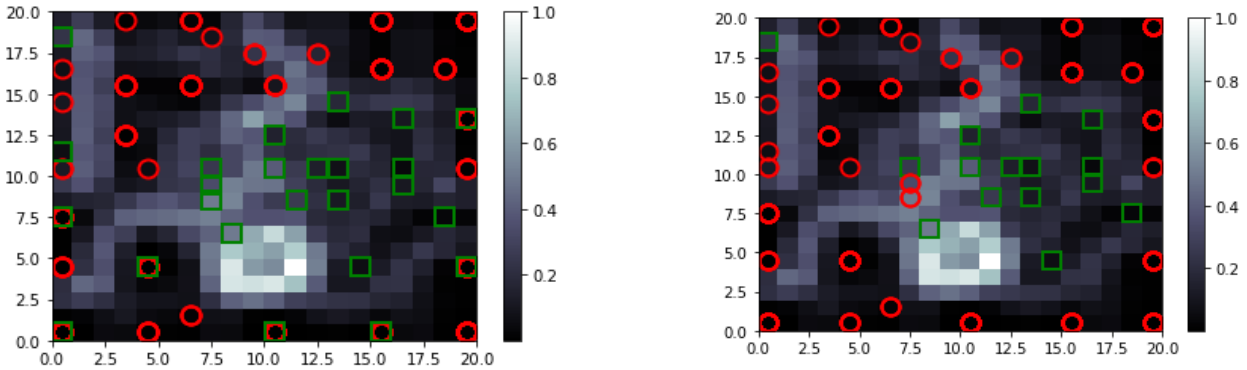
	precision	recall	f1-score	support
0	0.95	1.00	0.97	622
1	1.00	0.44	0.62	63
accuracy			0.95	685
macro avg	0.97	0.72	0.79	685
weighted avg	0.95	0.95	0.94	685

```
0.948905109489051
[[622   0]
 [ 35  28]]
```

Σχήμα 8.24. Selective Forward BN middle local topology FIT results

Όντως τα αποτελέσματα δεν είναι και τόσο καλά. Συγκεκριμένα πετυχαίνουμε ένα πολύ καλό accuracy score αλλά αυξάνονται τα FN που αυτό δεν το θέλουμε.

Top topology



Σχήμα 8.25. Classifications of predicted Data

	precision	recall	f1-score	support
0	0.94	1.00	0.97	622
1	1.00	0.41	0.58	63
accuracy			0.95	685
macro avg	0.97	0.71	0.78	685
weighted avg	0.95	0.95	0.94	685

```
0.945985401459854
[[622   0]
 [ 37  26]]
```

Σχήμα 8.25. SF BN top local topology using middle model FIT results

	precision	recall	f1-score	support
0	0.94	1.00	0.97	622
1	1.00	0.38	0.55	63
accuracy			0.94	685
macro avg	0.97	0.69	0.76	685
weighted avg	0.95	0.94	0.93	685

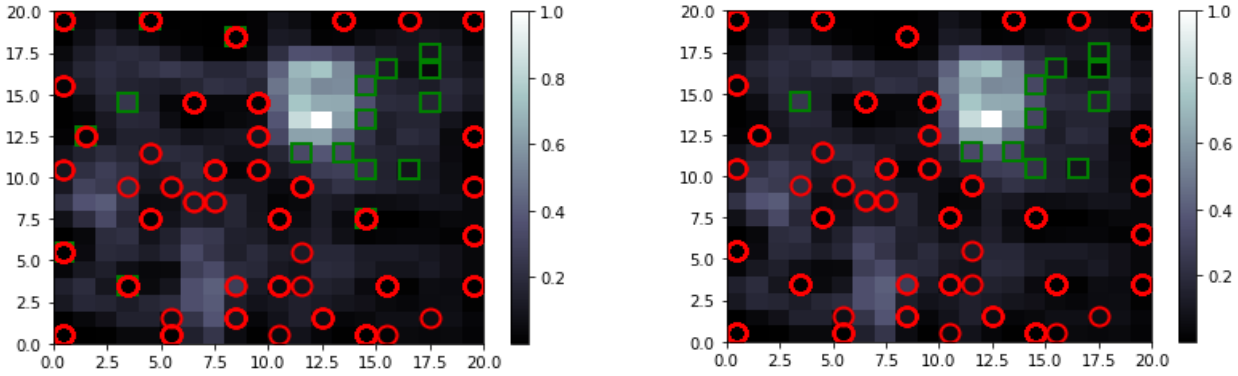
```
0.9430656934306569
[[622   0]
 [ 39  24]]
```

Σχήμα 8.26. SF BN top local topology FIT results

Αναμενόμενα αποτελέσματα χειρότερα από τα blackhole αλλά αυτή είναι η συμπεριφορά του αλγορίθμου για αυτή την επίθεση.

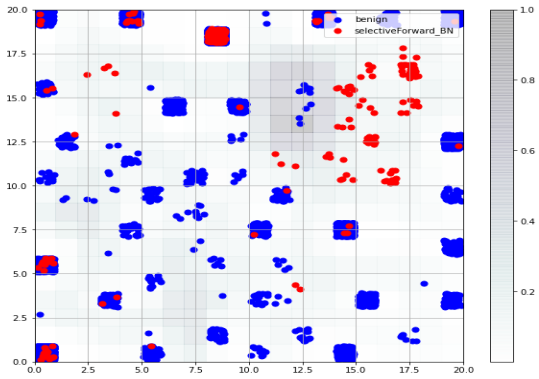
FIT DATA Network Detection

Middle topology



Σχήμα 8.27. Classifications of predicted Data

Σε αυτό το classification μπορεί το classification να φαίνεται καλό αλλά όπως είπα και πριν τα neurons παίρνουν πάνω από 1 data point. Αν τα data points είναι όμοια θα μπουν στο ίδιο neuron. Έτσι βλέπουμε και εδώ όπου κάποια λίγα neurons παίρνουν πολλά data points.



Σχήμα 8.28. Better classification

Όπως βλέπουμε στο Σχήμα 8.28 υπάρχουν νευρώνες που παίρνουν πολλά malicious inputs όπως επίσης και πολλά benign inputs. Αυτό επειδή η διασπορά στις τιμές των features του malicious node με τους benign είναι σχεδόν ίδια.

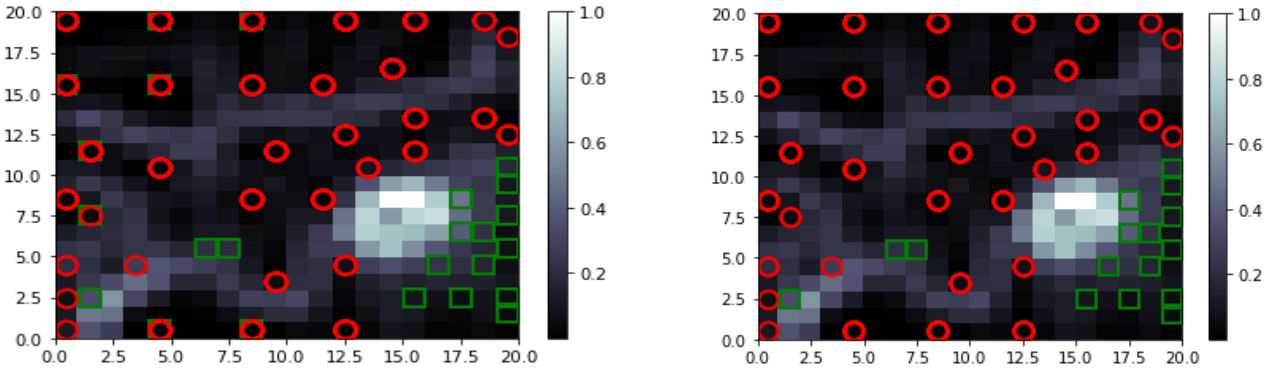
	precision	recall	f1-score	support
0	0.98	1.00	0.99	2056
1	1.00	0.46	0.63	63
accuracy			0.98	2119
macro avg	0.99	0.73	0.81	2119
weighted avg	0.98	0.98	0.98	2119

```
0.9839546956111374
[[2056    0]
 [  34   29]]
```

Σχήμα 8.29. SF BN middle network topology FIT results

Έτσι βλέπουμε και τα αρκετά FN που έχουμε. Αλλά παίρνουμε ένα πολύ καλό accuracy score.

Top topology



Σχήμα 8.30. Classifications of predicted Data

	precision	recall	f1-score	support
0	0.98	1.00	0.99	2056
1	1.00	0.41	0.58	63
accuracy			0.98	2119
macro avg	0.99	0.71	0.79	2119
weighted avg	0.98	0.98	0.98	2119

```
0.9825389334591789
[[2056    0]
 [  37   26]]
```

Σχήμα 8.31. SF BN top network topology using middle model FIT results

	precision	recall	f1-score	support
0	0.98	1.00	0.99	2056
1	1.00	0.44	0.62	63
accuracy			0.98	2119
macro avg	0.99	0.72	0.80	2119
weighted avg	0.98	0.98	0.98	2119

```
0.9834827748938179
[[2056    0]
 [  35   28]]
```

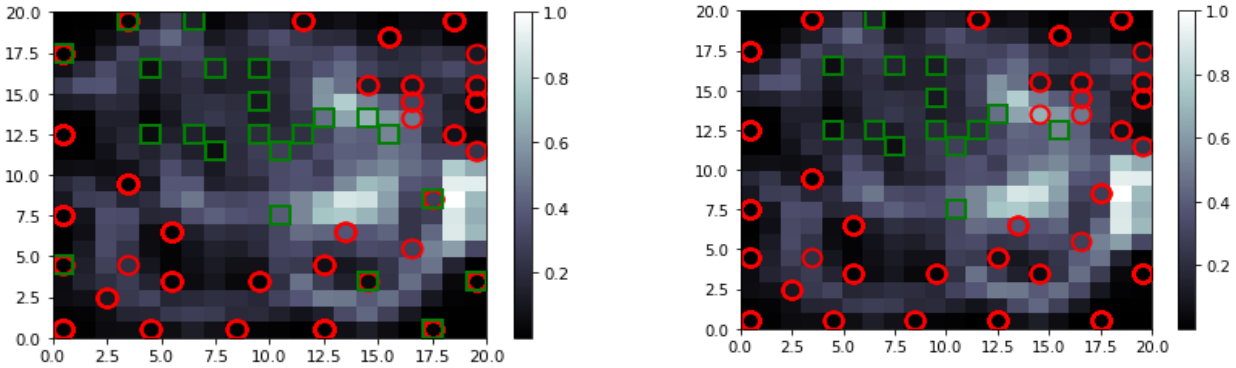
Σχήμα 8.32. SF BN top network topology FIT results

Όπως και στο middle έχουμε παρόμοια αποτελέσματα για το top topology για τους ίδιους λόγους.

8.4 Selective forward FR

FIT DATA Local Detection

Middle topology



Σχήμα 8.33. Classifications of predicted Data

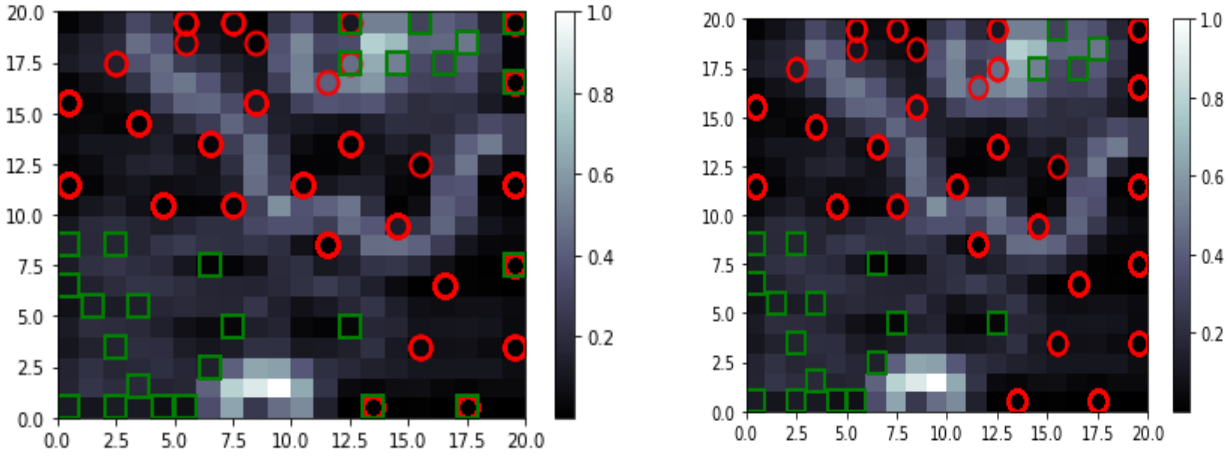
	precision	recall	f1-score	support
0	0.94	1.00	0.97	622
1	1.00	0.37	0.53	63
accuracy			0.94	685
macro avg	0.97	0.68	0.75	685
weighted avg	0.95	0.94	0.93	685

0.9416058394160584
[[622 0]
[40 23]]

Σχήμα 8.34. SF FR middle local topology FIT results

Αναμενόμενα αποτελέσματα παρόμοια με το selective forward BN για τους ίδιους λόγους.

Top topology



Σχήμα 8.35. Classifications of predicted Data

	precision	recall	f1-score	support
0	0.97	1.00	0.98	622
1	1.00	0.65	0.79	63
accuracy			0.97	685
macro avg	0.98	0.83	0.89	685
weighted avg	0.97	0.97	0.96	685

0.9678832116788321
[[622 0]
[22 41]]

Σχήμα 8.36. SF FR top local topology using middle model FIT results

	precision	recall	f1-score	support
0	0.97	1.00	0.98	622
1	1.00	0.65	0.79	63
accuracy			0.97	685
macro avg	0.98	0.83	0.89	685
weighted avg	0.97	0.97	0.96	685

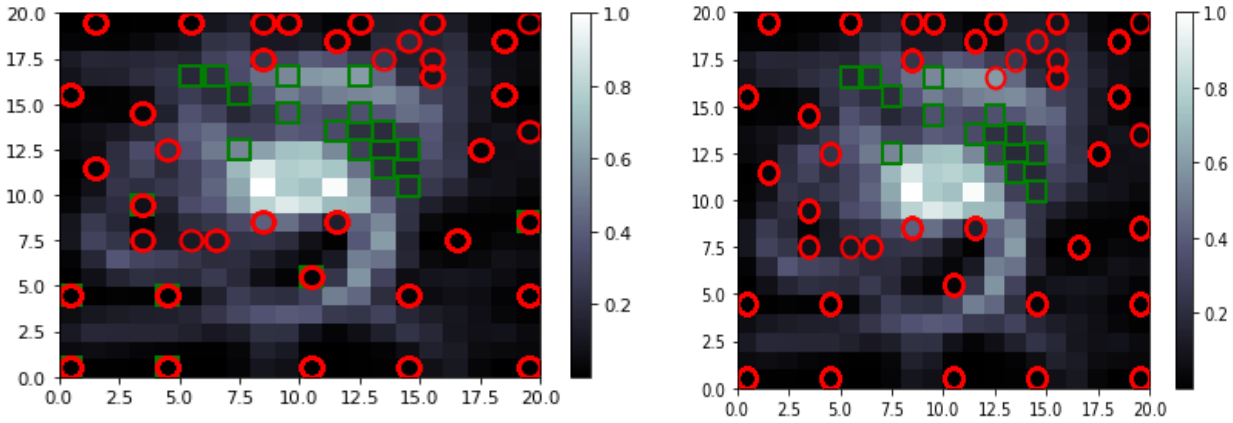
```
0.9678832116788321
[[622   0]
 [ 22  41]]
```

Σχήμα 8.37. SF FR top local topology FIT results

Από τα Σχήμα 8.36, 8.37 έχουμε βελτιωμένα αποτελέσματα για το top topology απ' ότι στο middle.

FIT DATA Network Detection

Middle topology



Σχήμα 8.38. Classifications of predicted Data

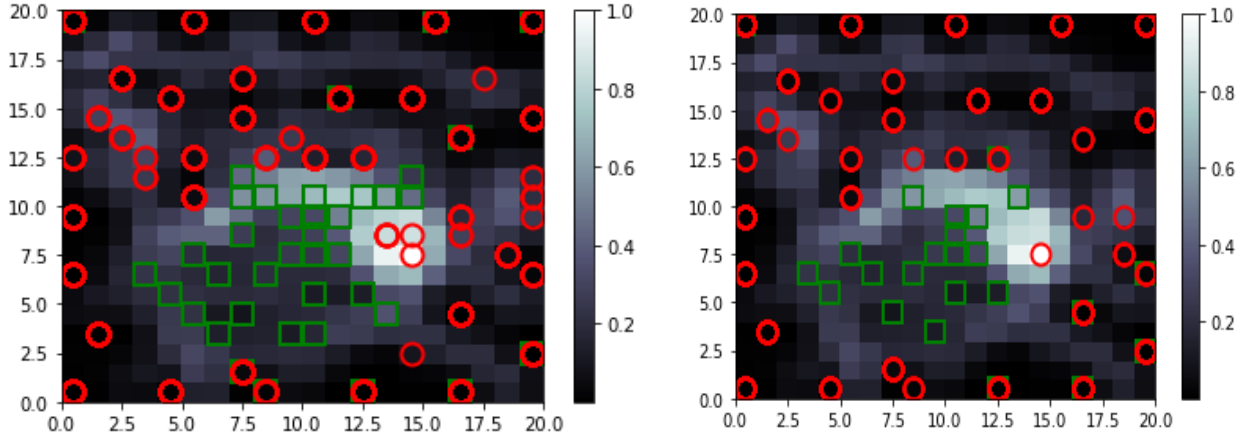
	precision	recall	f1-score	support
0	0.98	1.00	0.99	2055
1	1.00	0.37	0.53	63
accuracy			0.98	2118
macro avg	0.99	0.68	0.76	2118
weighted avg	0.98	0.98	0.98	2118

```
0.9811142587346553
[[2055    0]
 [  40   23]]
```

Σχήμα 8.39. SF FR middle network topology FIT results

Όπως και στο selective forward BN έτσι και στο FR έχουμε παρόμοια αποτελέσματα. Οι λόγοι είναι οι ίδιοι.

Top topology



Σχήμα 8.40. Classifications of predicted Data

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2057
1	1.00	0.65	0.79	63
accuracy			0.99	2120
macro avg	0.99	0.83	0.89	2120
weighted avg	0.99	0.99	0.99	2120

```
0.9896226415094339
[[2057    0]
 [  22   41]]
```

Σχήμα 8.41. SF FR top network topology using middle model FIT results

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2057
1	1.00	0.65	0.79	63
accuracy			0.99	2120
macro avg	0.99	0.83	0.89	2120
weighted avg	0.99	0.99	0.99	2120

```
0.9896226415094339
[[2057    0]
 [  22   41]]
```

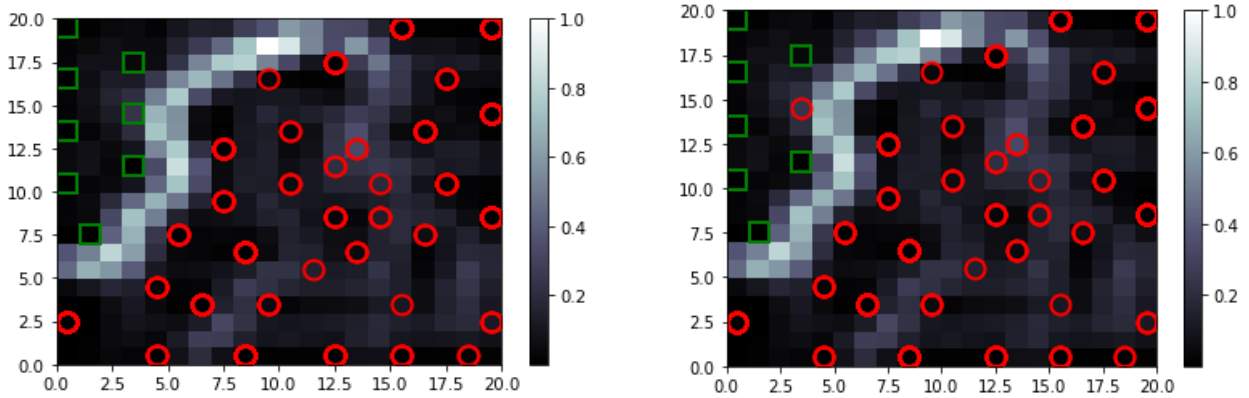
Σχήμα 8.42. SF FR top network topology FIT results

Όπως και στο top topology του SF BN έτσι και στο SF FR. Βλέπουμε μια βελτίωση στα FN και στο accuracy score για τους ίδιους λόγους.

8.5 Sinkhole

FIT DATA Local Detection

Middle topology



Σχήμα 8.43. Classifications of predicted Data

Για τα sinkhole attacks αναμένουμε τα καλύτερα αποτελέσματα όπως και με τον Isolation Forest. Από το Σχήμα 8.43 βλέπουμε ένα πολύ καλό και καθαρό classification και καταλαβαίνουμε ότι λογικά τα αποτελέσματα θα είναι εξαιρετικά.

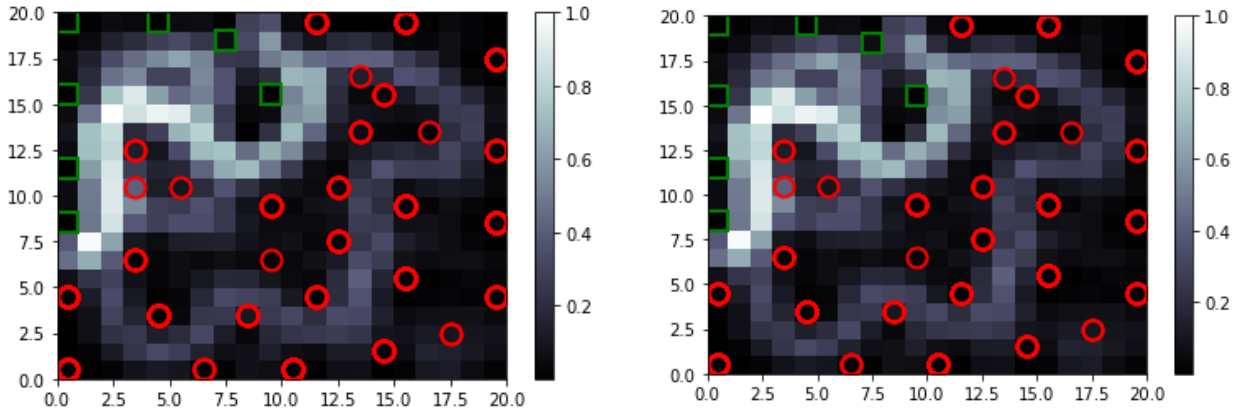
	precision	recall	f1-score	support
0	1.00	1.00	1.00	622
1	1.00	0.98	0.99	62
accuracy			1.00	684
macro avg	1.00	0.99	1.00	684
weighted avg	1.00	1.00	1.00	684

```
0.9985380116959064
[[622   0]
 [  1  61]]
```

Σχήμα 8.44. Sinkhole middle local topology FIT results

Όπως αναμέναμε έχουμε τα καλύτερα αποτελέσματα για το sinkhole attack με 1 μόνο FN άρα 100% επιτυχία.

Top topology



Σχήμα 8.45. Classifications of predicted Data

	precision	recall	f1-score	support
0	1.00	1.00	1.00	622
1	1.00	1.00	1.00	63
accuracy			1.00	685
macro avg	1.00	1.00	1.00	685
weighted avg	1.00	1.00	1.00	685

```
1.0
[[622   0]
 [  0  63]]
```

Σχήμα 8.46. Sinkhole top local topology using middle topology FIT results

	precision	recall	f1-score	support
0	1.00	1.00	1.00	622
1	1.00	1.00	1.00	63
accuracy			1.00	685
macro avg	1.00	1.00	1.00	685
weighted avg	1.00	1.00	1.00	685

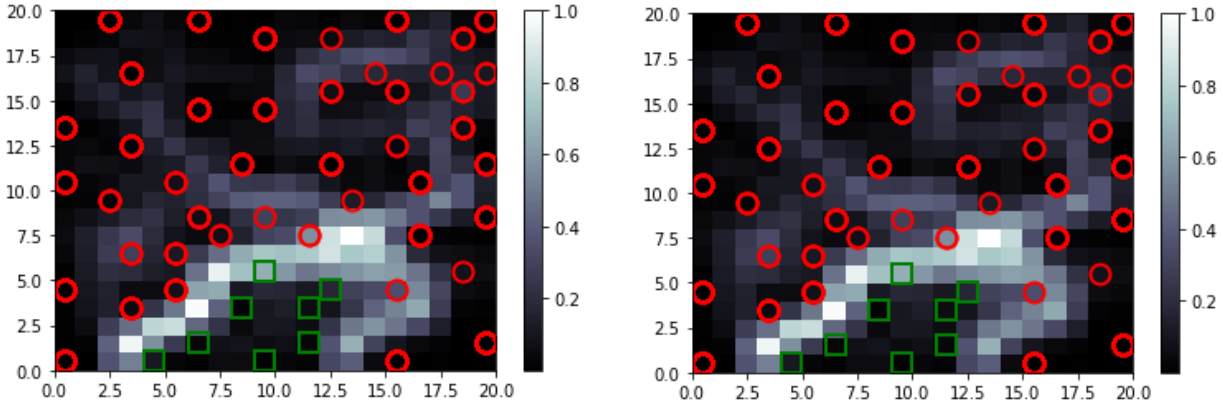
```
1.0
[[622   0]
 [  0  63]]
```

Σχήμα 8.47. Sinkhole top local topology FIT results

Στο top topology ο αλγόριθμος κατάφερε να πετύχει 100% accuracy βρίσκοντας όλα τα attacks. Αυτά είναι και τα ιδανικά αποτελέσματα τα οποία είναι πολύ σπάνιο να προκύψουν.

FIT DATA Network Detection

Middle topology



Σχήμα 8.48. Classifications of predicted Data

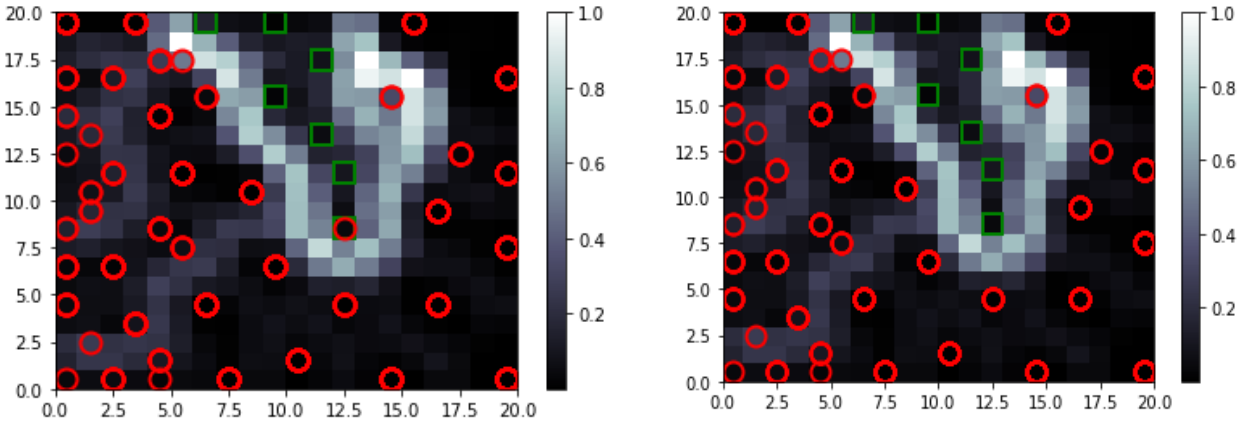
	precision	recall	f1-score	support
0	1.00	1.00	1.00	2056
1	1.00	1.00	1.00	62
accuracy			1.00	2118
macro avg	1.00	1.00	1.00	2118
weighted avg	1.00	1.00	1.00	2118

```
1.0
[[2056    0]
 [   0   62]]
```

Σχήμα 8.49. Sinkhole top local topology FIT results

Για ακόμη μια φορά ο αλγόριθμος πετυχαίνει 100% accuracy score για το middle topology στο network detection.

Top topology



Σχήμα 8.50. Classifications of predicted Data

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2052
1	0.98	1.00	0.99	63
accuracy			1.00	2115
macro avg	0.99	1.00	1.00	2115
weighted avg	1.00	1.00	1.00	2115

```
0.9995271867612293
[[2051    1]
 [   0   63]]
```

Σχήμα 8.51. Sinkhole top network topology using middle topology FIT results

Στην top topology ο αλγόριθμος βρίσκει 1 FP μόνο το οποίο δεν είναι καθόλου πρόβλημα. Συγκεκριμένα όταν βρίσκει FP αντί FN δεν είναι τόσο μεγάλο το πρόβλημα για το λόγο ότι τα FN είναι data points που θεωρεί σαν benign ενώ στην πραγματικότητα είναι malicious και αυτό δεν το θέλουμε.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2052
1	0.98	1.00	0.99	63
accuracy			1.00	2115
macro avg	0.99	1.00	1.00	2115
weighted avg	1.00	1.00	1.00	2115

```
0.9995271867612293
[[2051    1]
 [   0   63]]
```

Σχήμα 8.52. Sinkhole top network topology FIT results

8.6 COOJA DATA

COOJA DATA Local Detection

Middle topology

	precision	recall	f1-score	support
0	0.94	1.00	0.97	576
1	0.95	0.33	0.49	58
accuracy			0.94	634
macro avg	0.94	0.66	0.73	634
weighted avg	0.94	0.94	0.92	634

0.9369085173501577
[[575 1]
[39 19]]

BH-BN middle

	precision	recall	f1-score	support
0	0.95	1.00	0.97	576
1	1.00	0.43	0.60	58
accuracy			0.95	634
macro avg	0.97	0.72	0.79	634
weighted avg	0.95	0.95	0.94	634

0.9479495268138801
[[576 0]
[33 25]]

BH-FR middle

	precision	recall	f1-score	support
0	0.92	1.00	0.96	576
1	1.00	0.09	0.16	58
accuracy			0.92	634
macro avg	0.96	0.54	0.56	634
weighted avg	0.92	0.92	0.88	634

0.916403785488959
[[576 0]
[53 5]]

SF-BN middle

	precision	recall	f1-score	support
0	0.92	1.00	0.96	576
1	1.00	0.10	0.19	58
accuracy			0.92	634
macro avg	0.96	0.55	0.57	634
weighted avg	0.92	0.92	0.89	634

0.917981072555205
[[576 0]
[52 6]]

SF-FR middle

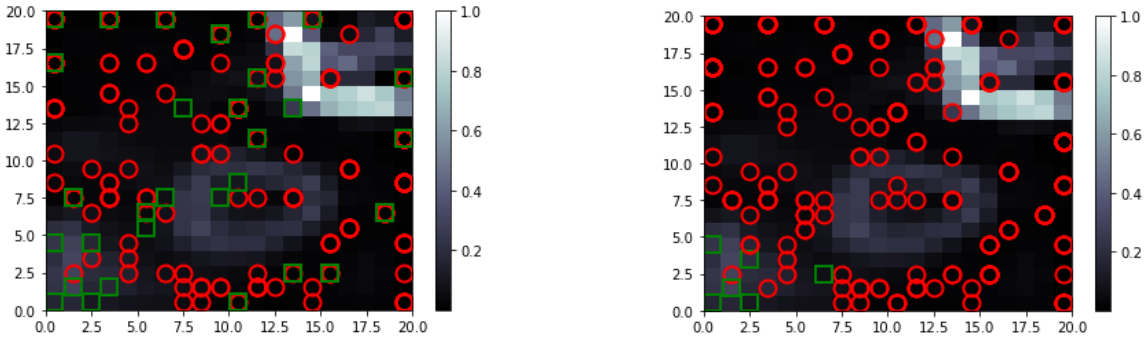
	precision	recall	f1-score	support
0	0.92	1.00	0.95	576
1	0.71	0.09	0.15	58
accuracy			0.91	634
macro avg	0.81	0.54	0.55	634
weighted avg	0.90	0.91	0.88	634

0.9132492113564669
[[574 2]
[53 5]]

Sinkhole middle

Όπως και στον Isolation Forest τόσο και στον SOM βλέπουμε την παρακμή στα αποτελέσματα όταν δοκιμάζουμε τα COOJA DATA. Τα αποτελέσματα και σε αυτό τον αλγόριθμο είναι

χειρότερα από τα FIT DATA που είχαμε εξαιρετικά αποτελέσματα. Οι λόγοι είναι σαφώς οι ίδιοι όπως και στον Isolation Forest που αναφέραμε πολλές φορές.



Classifications of predicted Data

Βλέπουμε ξεκάθαρα από τα πάνω classification (για το sinkhole attack) του αλγορίθμου ότι δεν μπορεί να ξεχωρίσει καθαρά τα benign από τα malicious data set και αυτό έχει αντίκτυπο στα αποτελέσματα που δεν είναι καλά.

random topology

	precision	recall	f1-score	support
0	0.98	1.00	0.99	576
1	0.98	0.79	0.88	58
accuracy			0.98	634
macro avg	0.98	0.90	0.93	634
weighted avg	0.98	0.98	0.98	634

```
0.9794952681388013
[[575   1]
 [ 12  46]]
```

BH-BN random

	precision	recall	f1-score	support
0	0.94	1.00	0.97	576
1	1.00	0.41	0.59	58
accuracy			0.95	634
macro avg	0.97	0.71	0.78	634
weighted avg	0.95	0.95	0.94	634

```
0.9463722397476341
[[576   0]
 [ 34  24]]
```

	precision	recall	f1-score	support
0	0.98	0.99	0.99	576
1	0.92	0.83	0.87	58
accuracy			0.98	634
macro avg	0.95	0.91	0.93	634
weighted avg	0.98	0.98	0.98	634

```
0.9779179810725552
[[572   4]
 [ 10  48]]
```

BH-FR random

	precision	recall	f1-score	support
0	0.95	1.00	0.98	576
1	1.00	0.52	0.68	58
accuracy			0.96	634
macro avg	0.98	0.76	0.83	634
weighted avg	0.96	0.96	0.95	634

```
0.9558359621451105
[[576   0]
 [ 28  30]]
```

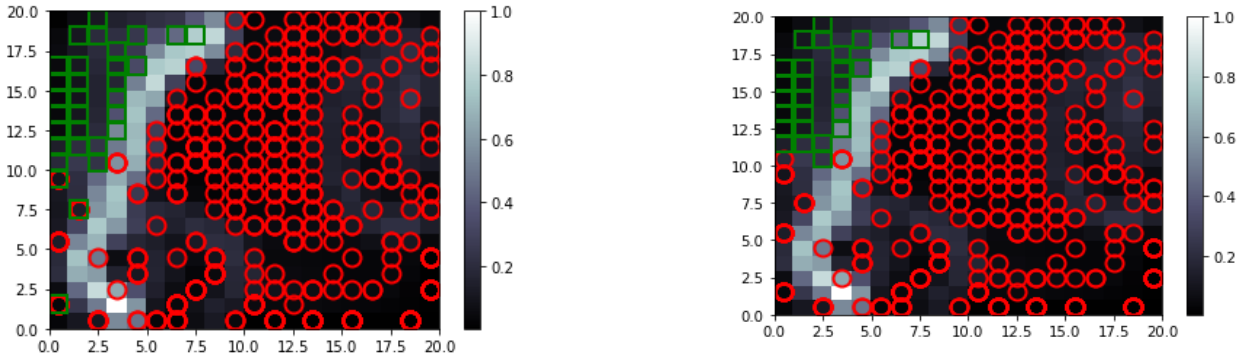
SF-BN random *SF-FR random*

	precision	recall	f1-score	support
0	0.99	1.00	1.00	576
1	1.00	0.93	0.96	58
accuracy			0.99	634
macro avg	1.00	0.97	0.98	634
weighted avg	0.99	0.99	0.99	634

0.9936908517350158
[[576 0]
[4 54]]

Sinkhole random

Ξεκάθαρη η βελτίωση στο random topology όπου τα αποτελέσματα είναι εξαιρετικά όπως και στα FIT DATA. Αυτό γιατί όπως είδαμε πολλές φορές, τα δεδομένα μεταξύ benign και malicious αποκλίνουν σε συγκεκριμένα features αναλόγως της επίθεσης.



Classifications of predicted Data

Βλέπουμε από το πιο πάνω classification την δυνατότητα του αλγορίθμου να κάνει καθαρό και ξεκάθαρο classify των δεδομένων benign και malicious.

top topology

	precision	recall	f1-score	support
0	0.96	1.00	0.98	576
1	0.97	0.64	0.77	58
accuracy			0.97	634
macro avg	0.97	0.82	0.88	634
weighted avg	0.97	0.97	0.96	634

0.9652996845425867
[[575 1]
[21 37]]

	precision	recall	f1-score	support
0	0.96	0.99	0.98	576
1	0.90	0.64	0.75	58
accuracy			0.96	634
macro avg	0.93	0.82	0.86	634
weighted avg	0.96	0.96	0.96	634

0.9605678233438486
[[572 4]
[21 37]]

BH-BN top

	precision	recall	f1-score	support
0	0.94	1.00	0.97	576
1	1.00	0.41	0.59	58
accuracy			0.95	634
macro avg	0.97	0.71	0.78	634
weighted avg	0.95	0.95	0.94	634

0.9463722397476341
[[576 0]
[34 24]]

BH-FR top

	precision	recall	f1-score	support
0	0.96	1.00	0.98	576
1	1.00	0.64	0.78	58
accuracy			0.97	634
macro avg	0.98	0.82	0.88	634
weighted avg	0.97	0.97	0.96	634

0.9668769716088328
[[576 0]
[21 37]]

SF-BN top

SF-FR top

	precision	recall	f1-score	support
0	0.92	0.99	0.96	576
1	0.75	0.16	0.26	58
accuracy			0.92	634
macro avg	0.84	0.57	0.61	634
weighted avg	0.91	0.92	0.89	634

0.917981072555205
[[573 3]
[49 9]]

Sinkhole top

Όπως και στον Isolation Forest στο local detection top topology βλέπουμε ότι τα αποτελέσματα είναι παρομοίως καλά όπως και στο random topology και σαφώς καλύτερα από το middle topology όπως παρατηρούμε όπως και στον Isolation Forest τα αποτελέσματα του sinkhole attack είναι χειρότερα. Ο λόγος όπως τον εξηγήσαμε πάρα πολλές φορές είναι ο ίδιος.

COOJA DATA Network Detection

Middle topology

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1959
1	1.00	0.38	0.55	58
accuracy			0.98	2017
macro avg	0.99	0.69	0.77	2017
weighted avg	0.98	0.98	0.98	2017

0.9821517104610809
[[1959 0]
[36 22]]

BH-BN middle

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1901
1	0.97	0.50	0.66	58
accuracy			0.98	1959
macro avg	0.98	0.75	0.83	1959
weighted avg	0.98	0.98	0.98	1959

0.9846860643185299
[[1900 1]
[29 29]]

BH-FR middle

	precision	recall	f1-score	support
0	0.97	1.00	0.99	1901
1	1.00	0.09	0.16	58
accuracy			0.97	1959
macro avg	0.99	0.54	0.57	1959
weighted avg	0.97	0.97	0.96	1959

0.9729453802960695
[[1901 0]
[53 5]]

SF-BN middle

	precision	recall	f1-score	support
0	0.97	1.00	0.99	1901
1	1.00	0.14	0.24	58
accuracy			0.97	1959
macro avg	0.99	0.57	0.61	1959
weighted avg	0.98	0.97	0.96	1959

0.9744767738642164
[[1901 0]
[50 8]]

SF-FR middle

	precision	recall	f1-score	support
0	0.97	1.00	0.98	1901
1	0.00	0.00	0.00	58
accuracy			0.97	1959
macro avg	0.49	0.50	0.49	1959
weighted avg	0.94	0.97	0.96	1959

0.9703930576824911
[[1901 0]
[58 0]]

Sinkhole attack

random topology

	precision	recall	f1-score	support
0	0.98	1.00	0.99	576
1	0.98	0.79	0.88	58
accuracy			0.98	634
macro avg	0.98	0.90	0.93	634
weighted avg	0.98	0.98	0.98	634

0.9794952681388013
[[575 1]
[12 46]]

BH-BN random

	precision	recall	f1-score	support
0	0.99	1.00	1.00	1901
1	0.94	0.81	0.87	58
accuracy			0.99	1959
macro avg	0.97	0.90	0.93	1959
weighted avg	0.99	0.99	0.99	1959

0.9928534966819806
[[1898 3]
[11 47]]

BH-FR random

	precision	recall	f1-score	support
0	0.94	1.00	0.97	576
1	1.00	0.41	0.59	58
accuracy			0.95	634
macro avg	0.97	0.71	0.78	634
weighted avg	0.95	0.95	0.94	634

0.9463722397476341
[[576 0]
[34 24]]

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1901
1	1.00	0.53	0.70	58
accuracy			0.99	1959
macro avg	0.99	0.77	0.84	1959
weighted avg	0.99	0.99	0.98	1959

0.9862174578866769
[[1901 0]
[27 31]]

SF-BN random

SF-FR random

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1901
1	1.00	0.93	0.96	58
accuracy			1.00	1959
macro avg	1.00	0.97	0.98	1959
weighted avg	1.00	1.00	1.00	1959

0.9979581419091373

[[1901 0]
[4 54]]

Sinkhole random

Top topology

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1901
1	1.00	0.57	0.73	58
accuracy			0.99	1959
macro avg	0.99	0.78	0.86	1959
weighted avg	0.99	0.99	0.99	1959

0.9872383869321082

[[1901 0]
[25 33]]

BH-BN top

	precision	recall	f1-score	support
0	0.99	1.00	1.00	1901
1	1.00	0.71	0.83	58
accuracy			0.99	1959
macro avg	1.00	0.85	0.91	1959
weighted avg	0.99	0.99	0.99	1959

0.9913221031138336

[[1901 0]
[17 41]]

BH-FR top

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1901
1	1.00	0.40	0.57	58
accuracy			0.98	1959
macro avg	0.99	0.70	0.78	1959
weighted avg	0.98	0.98	0.98	1959

0.9821337417049515

[[1901 0]
[35 23]]

SF-BN top

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1901
1	0.92	0.62	0.74	58
accuracy			0.99	1959
macro avg	0.96	0.81	0.87	1959
weighted avg	0.99	0.99	0.99	1959

0.9872383869321082

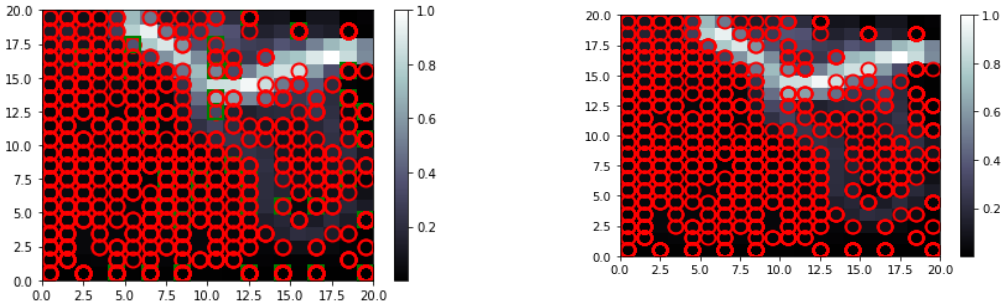
[[1898 3]
[22 36]]

SF-FR top

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1901
           1       0.00      0.00      0.00        58

    accuracy                           0.97      1959
   macro avg       0.49      0.50      0.49      1959
weighted avg       0.94      0.97      0.96      1959

0.9703930576824911
[[1901    0]
 [  58    0]]
```

Sinkhole top

Τα αποτελέσματα που βλέπουμε στις 3 τοπολογίες middle, top, random για το network topology είναι τα αναμενόμενα. Έχουμε αναλόγως καλά αποτελέσματα εκτός από το sinkhole attack για τους ξεκάθαρους λόγους. Βλέπουμε πόσο πολύ αδυνατεί ο αλγόριθμος να εντοπίσει τις επιθέσεις.



Classifications of predicted Data

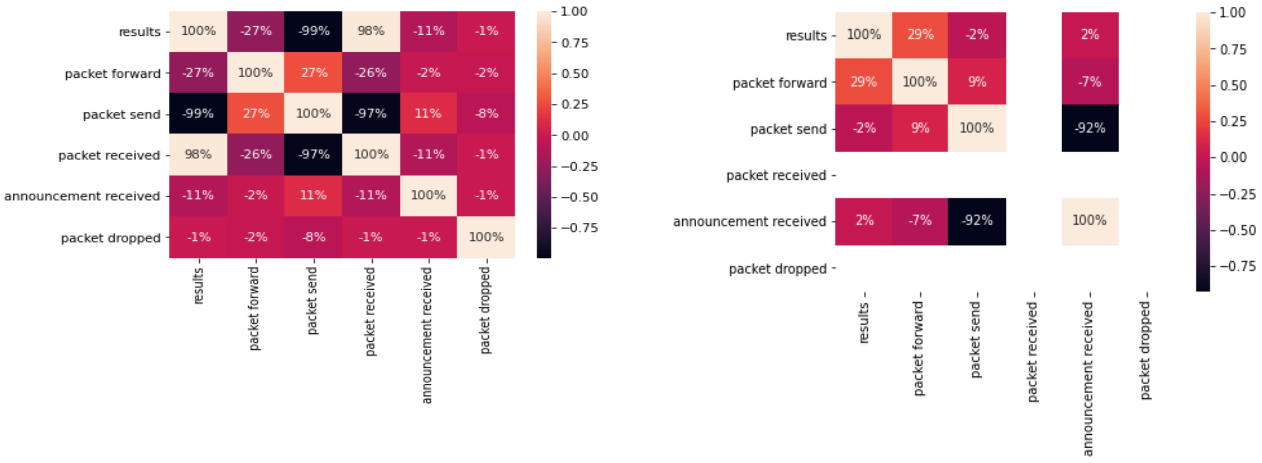
Κεφάλαιο 9

Συμπεράσματα

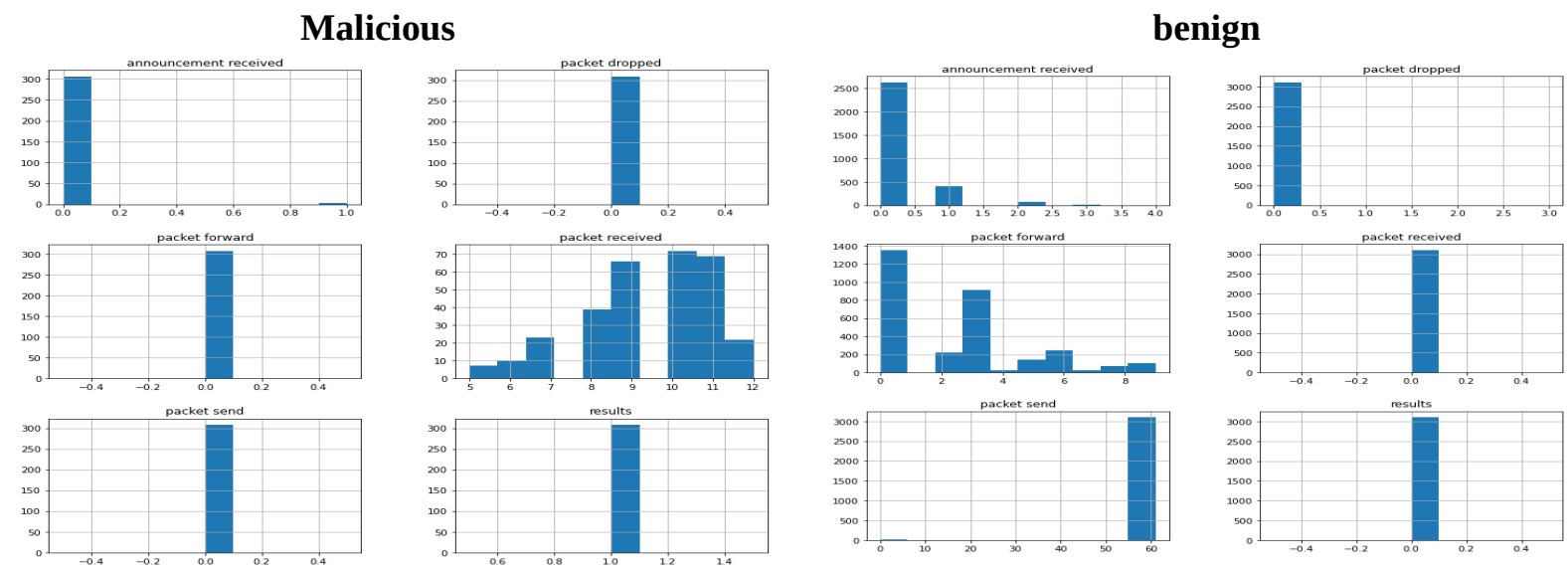
Φτάνουμε στο τελευταίο κεφάλαιο στο οποίο θα περιγράψω τα συμπεράσματα που πήρα όπως συμπεράσματα στα αποτελέσματα των αλγορίθμων, συμπεράσματα μεταξύ των 2 αλγορίθμων SOM και Isolation Forest, θα συγκρίνουμε τα αποτελέσματα μου με τα αποτελέσματα του project που αναπτύχθηκε από την Δρ. Χριστιάνα Ιωάννου χρησιμοποιώντας τον BLR αλγόριθμο και τέλος μια μελλοντική δουλειά.

9.1 COOJA vs FIT

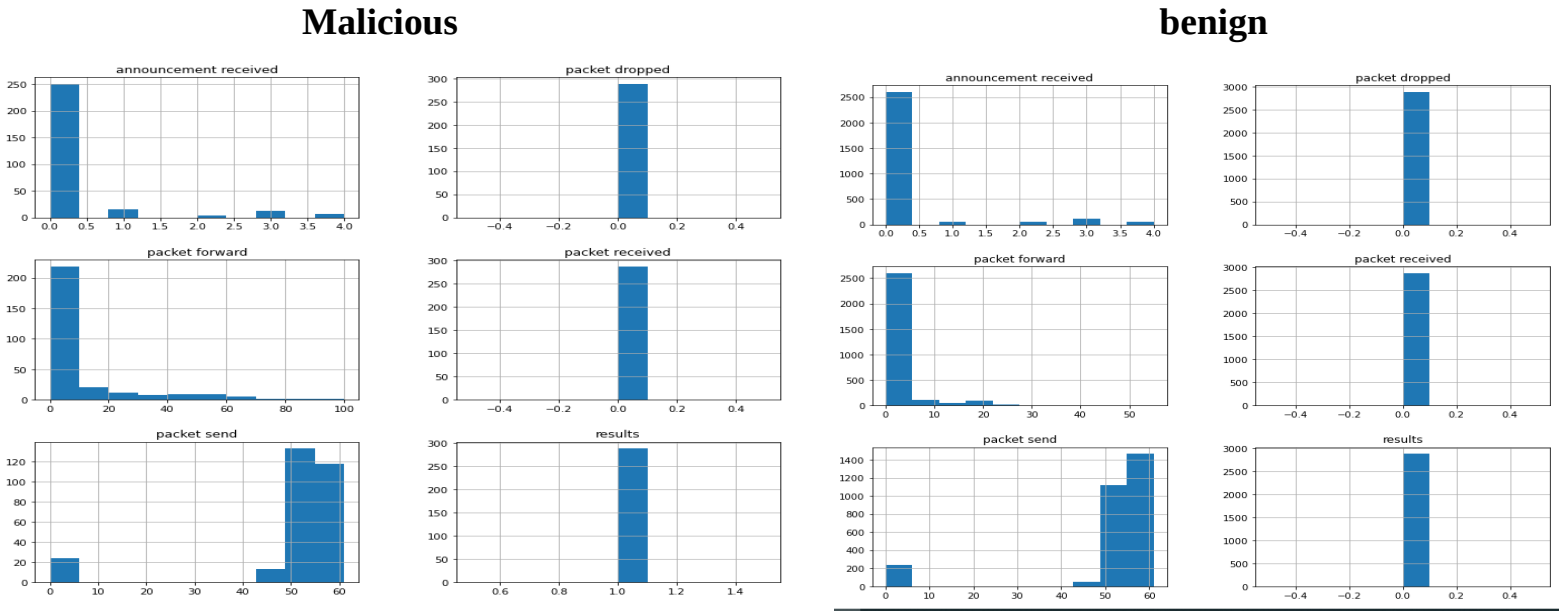
Είναι ξεκάθαρο και από το προηγούμενο κεφάλαιο στο οποίο είδαμε τα αποτελέσματα για όλα τα σενάρια και όπου ανάλυσα λεπτομερώς τους λόγους που πήρα αυτά τα αποτελέσματα, ότι τα αποτελέσματα που έχουν οι αλγόριθμοι στα FIT δεδομένα είναι καλύτερα από τα αποτελέσματα που έχουν στα COOJA δεδομένα. Ας θυμηθούμε ότι τα FIT data παράγονται σε ένα real environment ενώ τα COOJA δεδομένα παράγονται με την χρήση του COOJA Simulator. Είναι γεγονός λοιπόν ότι μας ενδιαφέρουν περισσότερο τα FIT data τα οποία είναι πιο ξεκάθαρα και ακριβής απ' ότι τα COOJA data. Διαπίστωσα μέσα από τα αποτελέσματα ότι υπάρχουν κάποιες περίεργες κατανομές των τιμών των features στα malicious nodes σε σχέση με τα benign στα COOJA data.



Σχήμα 9.1 correlation matrix FIT vs COOJA sinkhole attack



Σχήμα 9.2 Data points values for malicious and benign set FIT



Σχήμα 9.3 Data points values for malicious and benign set COOJA

Όπως είχαμε δει και στο προηγούμενο κεφάλαιο η sinkhole επίθεση είναι η επίθεση στην οποία ο malicious node προσπαθεί να μιμηθεί τον sink node. Από το Σχήμα 9.1. βλέπουμε ότι στα FIT δεδομένα έχουμε ένα πολύ μεγάλο correlation μεταξύ packet send, packet received και packet forward με το target variable (results). Επίσης στο Σχήμα 9.2 βλέπουμε για άλλη μια φορά την καθαρή διαφορά στις τιμές των features packet send, packet received και packet forward του

malicious node σε σχέση με τους benign nodes. Είναι λογικό να διαφέρουν οι τιμές σε αυτά τα τρία features αφού όπως είπαμε ο malicious node με sinkhole attack προσπαθεί να μιμηθεί τον sink node. Λόγω των λογικών ξεκάθαρων και καλών δεδομένων που έχουμε στα FIT data οι δύο αλγόριθμοι δίνουν εξαιρετικά αποτελέσματα, όπως είδαμε και στο προηγούμενο κεφάλαιο όπου ο SOM δίνει μέχρι και 100% με πλήρη εντοπισμό των επιθέσεων.

Από την άλλη μεριά όμως παρατηρούμε πολύ περίεργα δεδομένα στα COOJA. Και από τον correlation matrix και από τα σχήματα 9.2, 9.3 δεν βλέπουμε καθόλου αυτή την διαφορά στα 3 αυτά features μεταξύ malicious και benign nodes. Για την ακρίβεια βλέπουμε μια γενική ψηλή ομοιότητα μεταξύ των δεδομένων του malicious node και των benign nodes γεγονός το οποίο είναι περίεργο. Δυστυχώς αυτό, καθιστά σχεδόν αδύνατο στους συγκεκριμένους δύο αλγορίθμους να εντοπίσουν τα sinkhole attacks για τα COOJA δεδομένα, το οποίο είναι λογικό αφού τα δεδομένα είναι σχεδόν τα ίδια.

Έτσι από την μια έχουμε λογικά δεδομένα με καλή κατανομή στις τιμές τους (FIT) στα οποία έχουμε εξαιρετικά αποτελέσματα και από την άλλη μεριά έχουμε τα COOJA τα οποίο για κάποιο περίεργο λόγο είναι πολύ όμοια μεταξύ τους και αυτό δημιουργεί την ανικανότητα στους αλγορίθμους να έχουν καλά αποτελέσματα.

Αυτή η διαφορά στις τιμές των δεδομένων βρίσκεται γενικά και σε άλλα attacks. Στο middle topology στα COOJA τόσο σε local όσο και στο network detection τα αποτελέσματα δεν ήταν καλά. Στην random τοπολογία τα αποτελέσματα βελτιώθηκαν κατά πολύ και ήταν τα αναμενόμενα και στην top τοπολογία είχαμε πάλι καλά αποτελέσματα εκτός από το sinkhole attack και το λόγο μας τον δίνουν τα πιο πάνω features.

Άρα το τελικό συμπέρασμα είναι ότι τα αποτελέσματα που παίρνουμε με τα FIT data είναι πολύ καλά και καλύτερα από τα COOJA data.

Κλείνοντας δεν μπορώ να μην αναφέρω τις διαστάσεις των δεδομένων. Έχουμε να κάνουμε με μόλις 5 features στα δεδομένα μας. Σε προβλήματα τα οποία θέλουμε να λύσουμε με τεχνικές μηχανική μάθηση, τα dataset αποτελούνται από εκατοντάδες και περισσότερα features ενώ εμείς έχουμε ένα dataset με μόλις 5. Αυτό αυξάνει κατά πολύ και την ομοιότητα των δεδομένων. Έτσι οι αλγόριθμοι εκπαιδεύονται με μόλις 5 features και συγκεκριμένα 4 επειδή στις περισσότερες

περιπτώσεις αφαιρέσαμε το packet received (εκτός από τα sinkhole), με αποτέλεσμα να κατηγοριοποιούν τα data points κρίνοντας τα και βασιζόμενοι σε μόλις 4-5 features.

9.2 SOM vs Isolation Forest

Όταν φτάνουμε σε σύγκριση μεταξύ δύο αλγορίθμων, η σύγκριση δεν πρέπει να γίνεται μόνο στις μετρικές των αλγορίθμων. Μεγάλο ποσοστό της σύγκρισης είναι το κόστος και ο χρόνος εκτέλεσης του κάθε αλγόριθμου. Πόσο συνολικό χρόνο χρειάζονται και πόρους υπολογιστικούς πόρους απαιτούν.

Ξεκινώντας συγκρίνοντας τους δύο αλγόριθμους στις 3 μετρικές που είδαμε (accuracy score, precision, recall), είδαμε και από το προηγούμενο κεφάλαιο ότι ο SOM αντιδρά αρκετά καλύτερα απ' ότι ο Isolation Forest. Σχεδόν αν όχι σε όλα τα σενάρια ο SOM πετυχαίνει καλύτερο accuracy score, precision και recall από τον Isolation Forest κρατώντας τα FP όσο πιο λίγα γίνεται και αυξάνοντας τα TN και TP. Και οι δύο αλγόριθμοι έχουν ελάχιστα και στις περισσότερες φορές μηδενικά FN. Όσο αφορά όμως το κόστος και τον χρόνο εκτέλεσης είναι ξεκάθαρο ότι ο Isolation Forest είναι γρηγορότερος και απαιτεί λιγότερους υπολογιστικούς πόρους απ' ότι ο SOM.

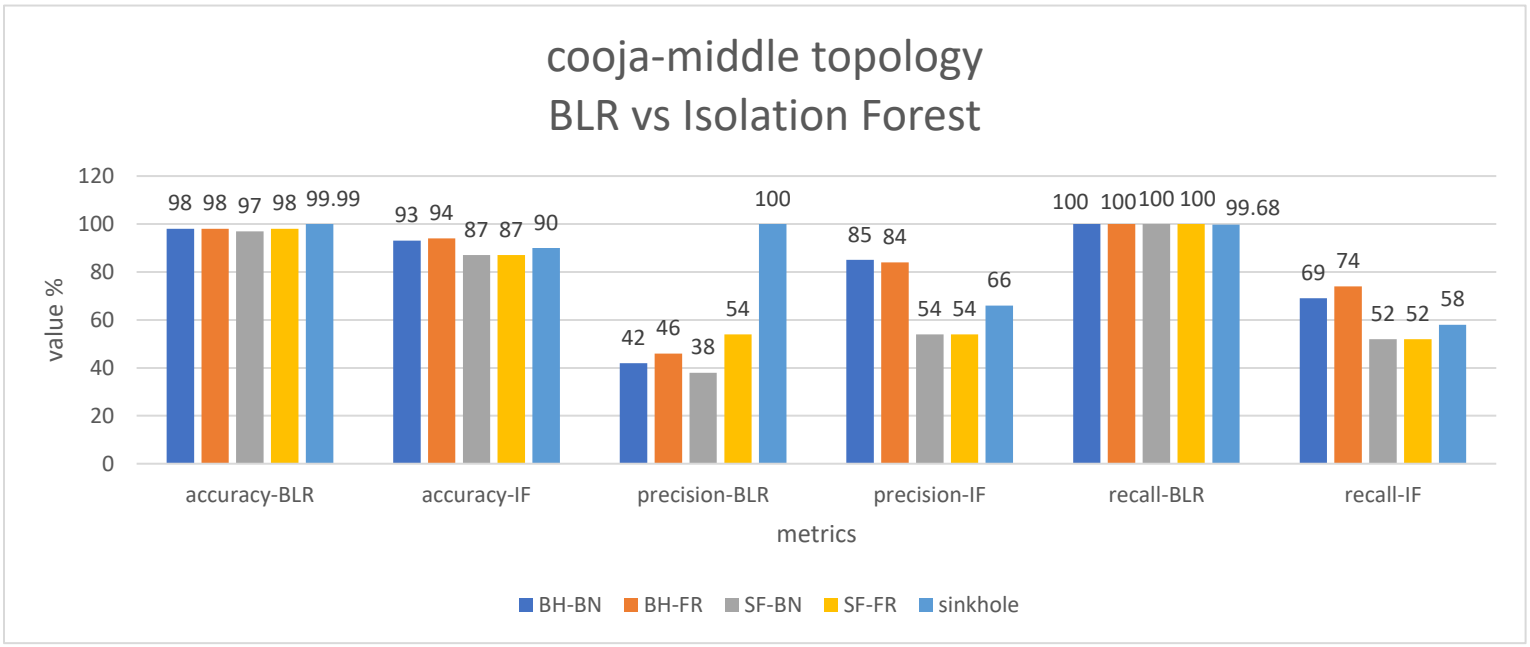
Έτσι έχουμε ένα δίλημμα να επιλέξουμε μεταξύ ακρίβειας και κόστους. Αυτό το trade off μπορεί να επιλυθεί με τον εξής τρόπο. Για το συγκεκριμένο πρόβλημα ίσως δεν είμαστε τόσο αυστηροί σε κόστος και χρόνο εφόσον ο όγκος των δεδομένων που έχουμε δεν είναι αρκετά μεγάλος φτάνοντας μέχρι και 10000-12000 data points (network detection). Αν όμως είχαμε τεράστιο όγκο δεδομένων και έπρεπε να λάβουμε σοβαρά υπόψιν τον χρόνο και το κόστος των αλγορίθμων, θα ήταν καλό να μελετήσουμε περισσότερο την απόφαση που θα πάρουμε μεταξύ των δύο αλγορίθμων. Έτσι αποφάσισα ότι ο SOM αλγόριθμος είναι καλύτερος και προτιμότερος για το συγκεκριμένο πρόβλημα που έχουμε να λύσουμε αν και κοστίζει περισσότερο από τον Isolation Forest.

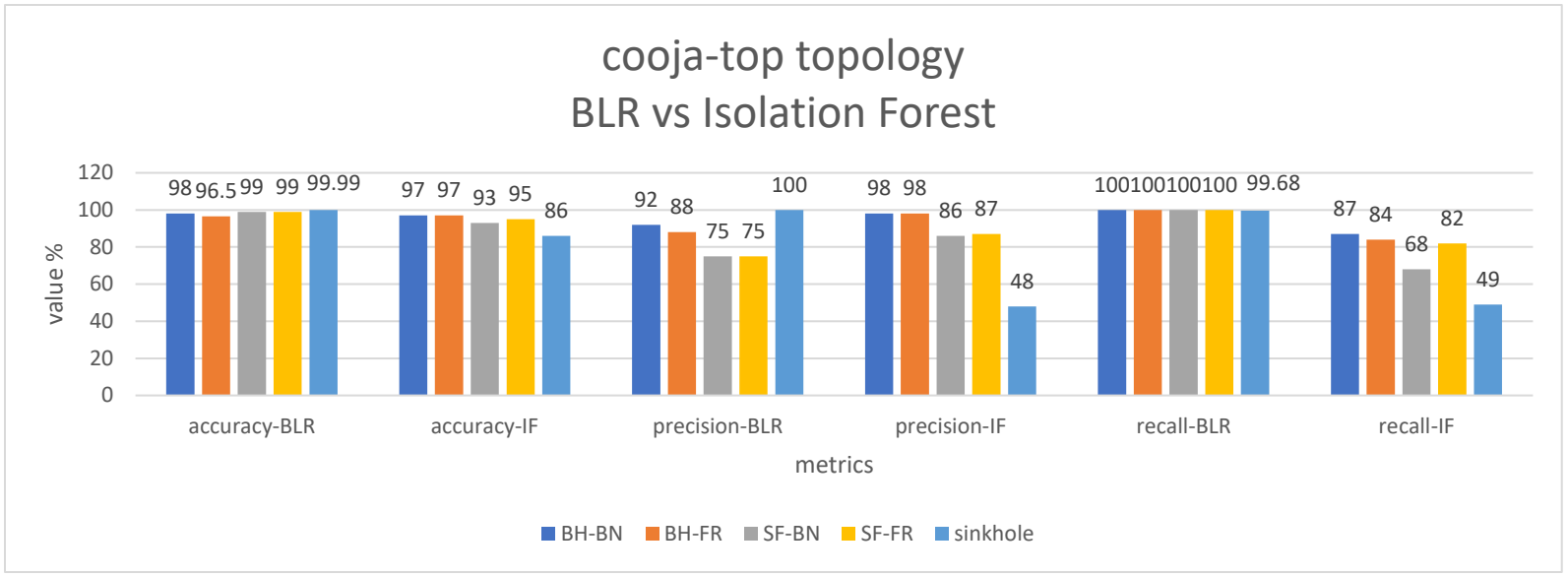
Ο λόγος που ο SOM αντιδράει καλύτερα σε precision, recall και accuracy score από τον Isolation Forest είναι επειδή χρησιμοποιεί ποιο ακριβές μετρικές, χρησιμοποιώντας πολλούς νευρώνες και υπολογίζοντας ακριβές αποστάσεις μεταξύ νευρώνων και vectors (data points). Από την άλλη μεριά ο Isolation Forest είναι ένας αλγόριθμος ειδικός στο να βρίσκει ανωμαλίες μέσα σε δεδομένα αλλά με γρήγορο τρόπο. Άρα αν τα δεδομένα ενός malicious node έχουν κάποιες

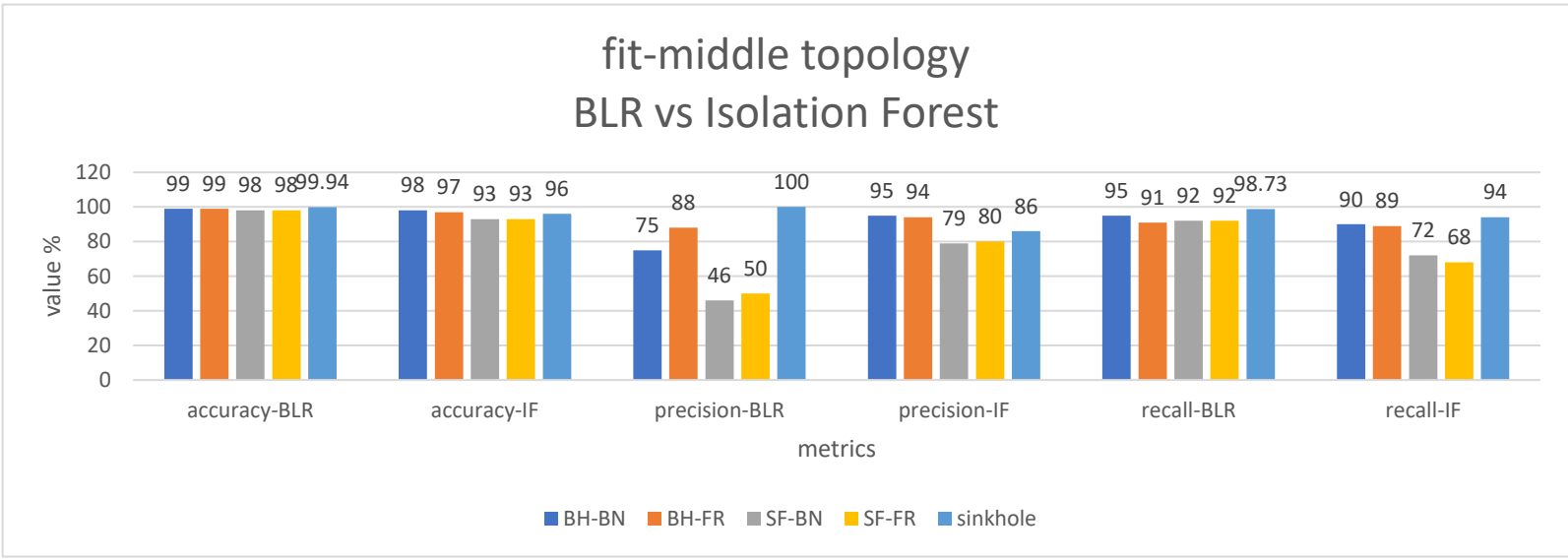
διαφορές αλλά όχι τεράστιες διαφορές από τα δεδομένα ενός benign node, ο Isolation Forest ίσως δυσκολευτεί να εντοπίσει τις ανωμαλίες σε αντίθεση με τον SOM. Γενικά είναι ένας αρκετά ευαίσθητος αλγόριθμος του οποίου οι παράμετροι και κυρίως το contamination πρέπει να είναι optimized για να δίνει όσο το δυνατό καλύτερα αποτελέσματα.

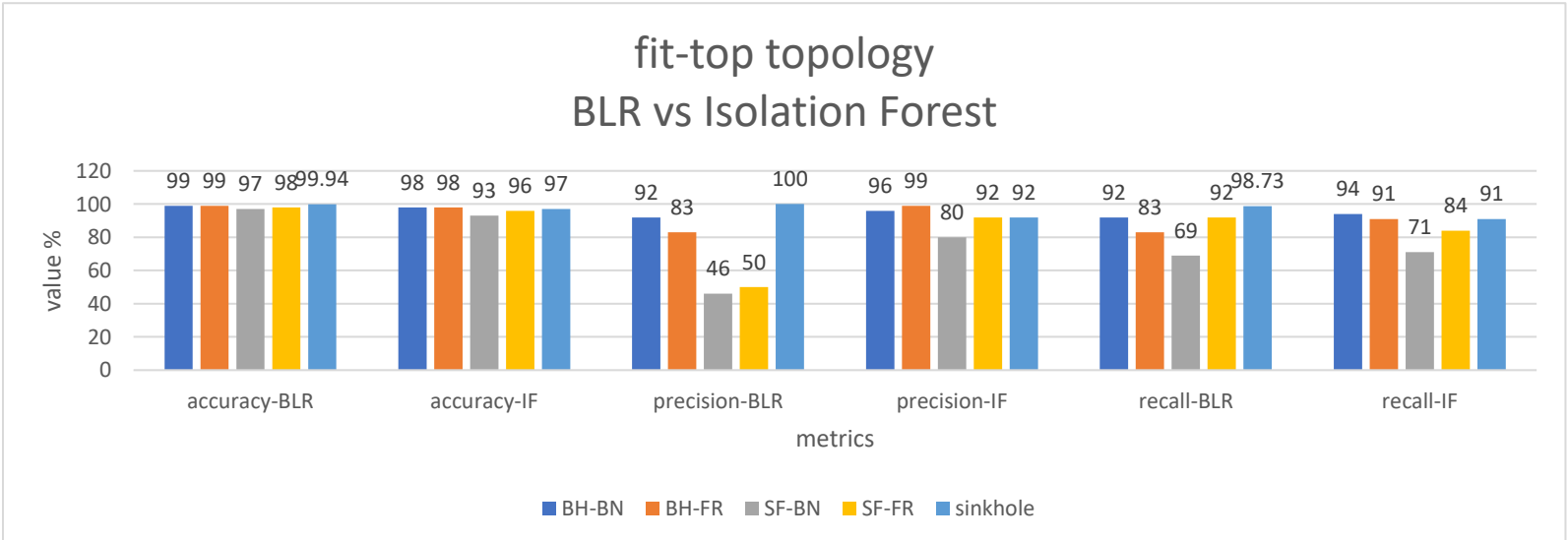
Ένα γενικό συμπέρασμα που πήρα από τους δύο αλγόριθμους είναι ότι δεν περίμενα να συμπεριφερθούν με τόσο καλό τρόπο στο συγκεκριμένο πρόβλημα. Όπως ανάφερα και στο κεφάλαιο 4, αυτοί οι αλγόριθμοι είναι unsupervised και η χρήση τους γίνεται σε εντελώς διαφορετικής φύσεως προβλήματα. Για το συγκεκριμένο πρόβλημα, δηλαδή ένα binary classification πρόβλημα με labeled dataset, χρησιμοποιούνται supervised learning αλγόριθμοι και συγκεκριμένα αλγόριθμοι οι οποίοι είναι ειδικοί στο να λύνουν προβλήματα binary classification, όπως ο SVM, Logistic Regression, Random Forest, KNN, Decision Tree και άλλοι πολλοί.

9.3 Σύγκριση με BLR



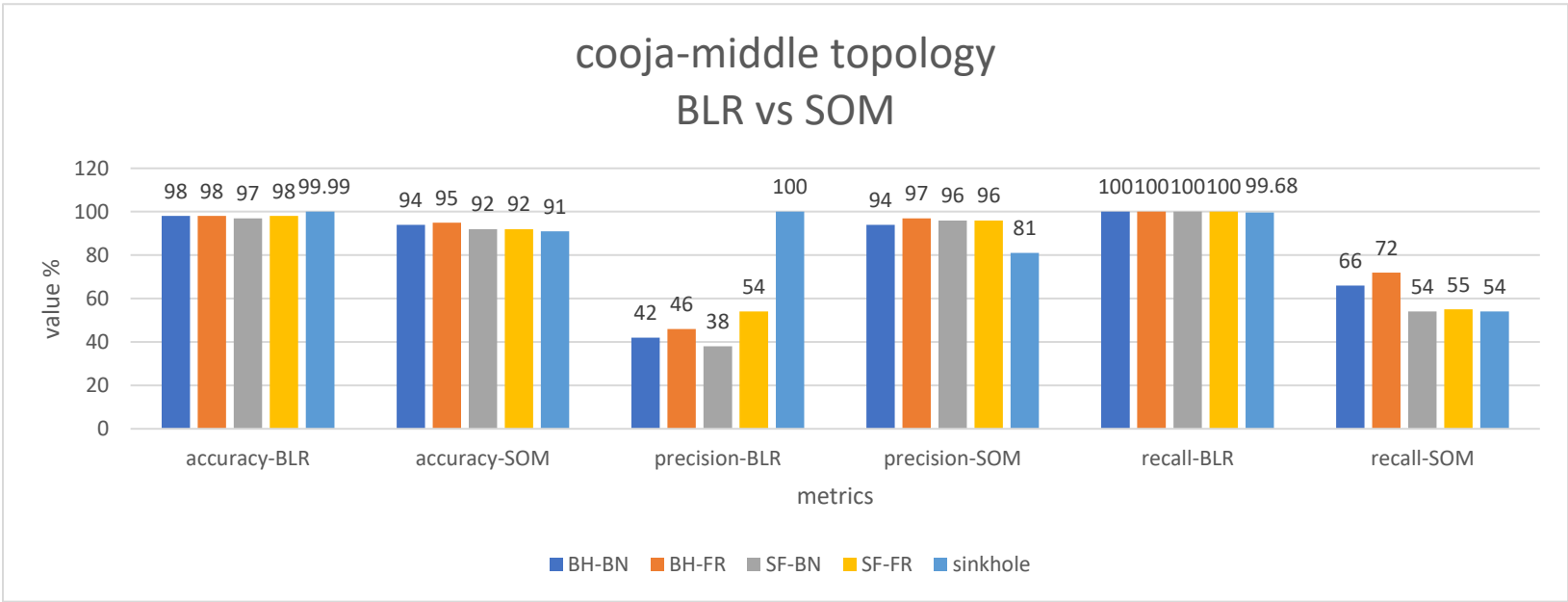

cooja-top topology
BLR vs Isolation Forest
value %
120
100
80
60
40
20
0
98 96.5 99 99 99.99
97 97 93 95 86
92 88 75 75 100
98 98 86 87 48
100 100 100 100 99.68
87 84 68 82 49
accuracy-BLR
accuracy-IF
precision-BLR
precision-IF
recall-BLR
recall-IF
metrics
BH-BN
BH-FR
SF-BN
SF-FR
sinkhole

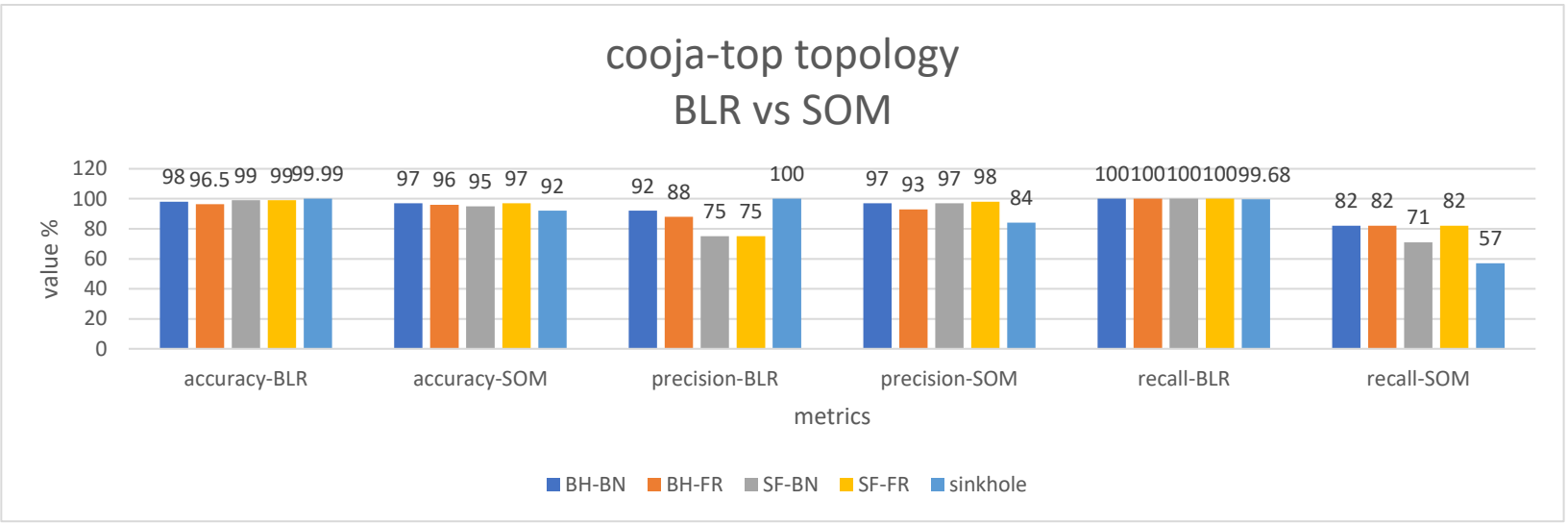

fit-middle topology
BLR vs Isolation Forest
value %
120
100
80
60
40
20
0
99 99 98 98 99.94
98 97 93 93 96
75 88 46 50 100
95 94 79 80 86
95 91 92 92 98.73
90 89 72 68 94
accuracy-BLR
accuracy-IF
precision-BLR
precision-IF
recall-BLR
recall-IF
metrics
BH-BN
BH-FR
SF-BN
SF-FR
sinkhole

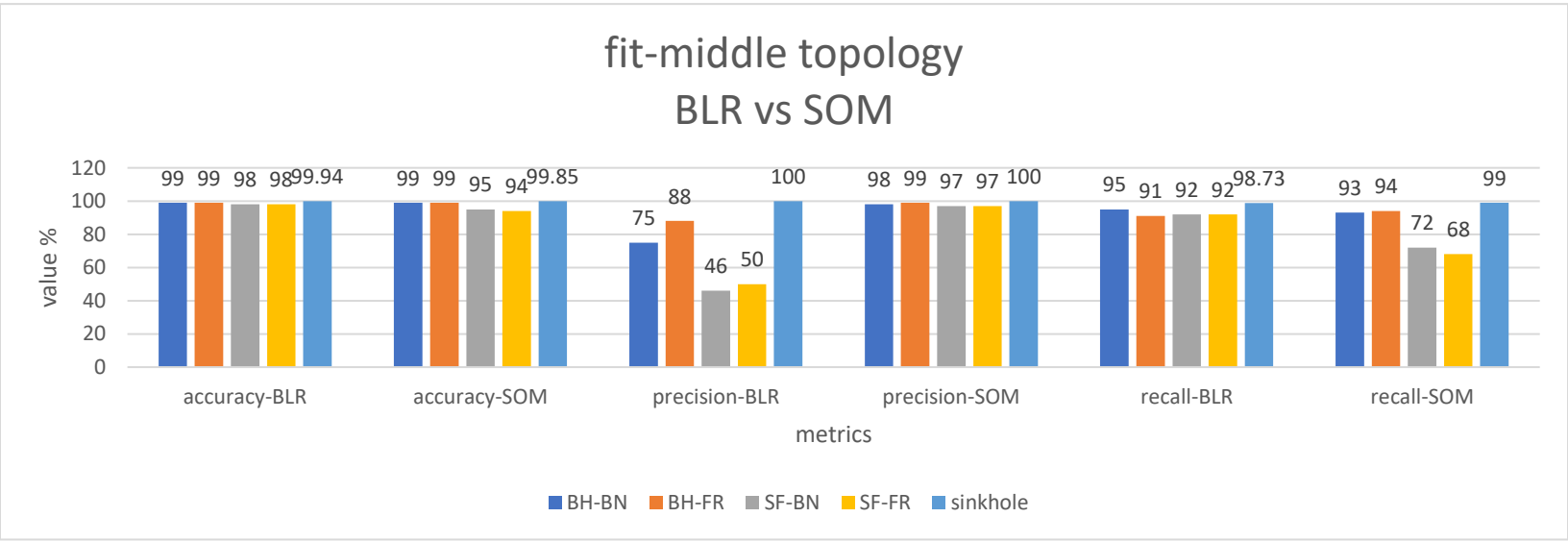

fit-top topology
BLR vs Isolation Forest
value %
120
100
80
60
40
20
0
99 99 97 98 99.94
98 98 93 96 97
92 83 46 50 100
96 99 80 92 92
92 83 69 92 98.73
94 91 71 84 91
accuracy-BLR
accuracy-IF
precision-BLR
precision-IF
recall-BLR
recall-IF
metrics
BH-BN
BH-FR
SF-BN
SF-FR
sinkhole

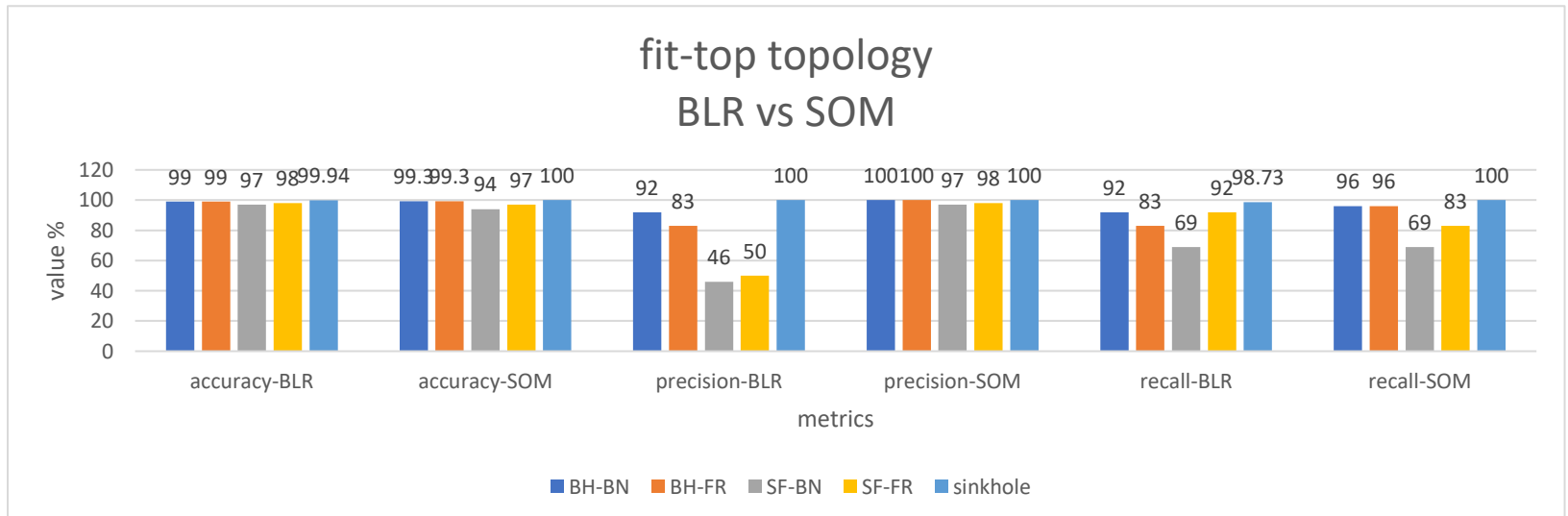
Στα πιο πάνω charts βλέπουμε την σύγκριση μεταξύ αποτελέσματα BLR και Isolation Forest. Για τα COOJA data middle topology παρατηρούμε ότι ο BLR έχει πολύ καλά αποτελέσματα ενώ ο Isolation Forest αν και πετυχαίνει ψηλότερα precision και ψηλά accuracy scores για τα attacks, έχει χαμηλό recall. Για την top topology τα αποτελέσματα βελτιώνονται. Γνωρίζουμε ήδη τον λόγο όπως τον είδαμε και στα αποτελέσματα του Isolation Forest. Εκτός από το γεγονός ότι οι τιμές στα δεδομένα του malicious node και του benign είναι όμοιες και αυτό κάνει δύσκολο τον Isolation Forest να εντοπίσει τα attacks με μεγάλη επιτυχία βλέπουμε ότι ο BLR αντιμετωπίζει το ίδιο πρόβλημα στο precision. Ο BLR γενικά είναι αλγόριθμος supervised και πολύ καλός για binary classification γι' αυτό είναι και λογικό να δουλεύει καλύτερα. Προχωρώντας στα FIT data έχουμε πολύ καλά αποτελέσματα και στους δύο αλγορίθμους. Για αυτά τα δεδομένα και οι δύο αλγόριθμοι δίνουν καλά accuracy score, precision και recall.

Σε γενικές γραμμές συγκρίνοντας τον Isolation Forest που είναι unsupervised με τον BLR που είναι supervised και ειδικός σε binary classification προβλήματα, είναι έκπληξη ότι ο Isolation Forest δίνει καλά αποτελέσματα όσο και ο BLR, κάποιες φορές χειρότερα, κάποιες καλύτερα.




cooja-top topology
BLR vs SOM
value %
120
100
80
60
40
20
0
98 96.5 99 99 99.99
97 96 95 97 92
92 88 75 75 100
97 93 97 98 84
100 100 100 100 99.68
82 82 71 82 57
accuracy-BLR
accuracy-SOM
precision-BLR
precision-SOM
recall-BLR
recall-SOM
metrics
BH-BN
BH-FR
SF-BN
SF-FR
sinkhole


fit-middle topology
BLR vs SOM
value %
120
100
80
60
40
20
0
99 99 98 98 99.94
99 99 95 94 99.85
75 88 46 50 100
98 99 97 97 100
95 91 92 92 98.73
93 94 72 68 99
accuracy-BLR
accuracy-SOM
precision-BLR
precision-SOM
recall-BLR
recall-SOM
metrics
BH-BN
BH-FR
SF-BN
SF-FR
sinkhole


fit-top topology
BLR vs SOM
value %
120
100
80
60
40
20
0
99 99 97 98 99.94
99.3 99.3 94 97 100
92 83 46 50 100
100 100 97 98 100
92 83 69 92 98.73
96 96 69 83 100
accuracy-BLR
accuracy-SOM
precision-BLR
precision-SOM
recall-BLR
recall-SOM
metrics
BH-BN
BH-FR
SF-BN
SF-FR
sinkhole

Για τα COOJA data παρατηρούμε τα ίδια όπως και στον Isolation Forest. Στα FIT data βλέπουμε ότι ο SOM έχει καλύτερα αποτελέσματα από τον Isolation Forest και σε αρκετές περιπτώσεις και από τον BLR. Όπως είπαμε είναι ένας πιο ακριβής αλγόριθμος αλλά δυστυχώς απαιτεί πολύ περισσότερους πόρους.

Τα συμπεράσματά μου από την σύγκριση των δύο αλγορίθμων με τον BLR είναι τα εξής. Αρχικά φαίνεται ότι αν και οι δύο αλγόριθμοι δεν είναι και οι πιο suitable για το πρόβλημα που έχουμε να αντιμετωπίσουμε, τα αποτελέσματα που έχουν είναι πάρα πολύ καλά. Αν εξαιρέσουμε το middle topology στα COOJA γενικά τα αποτελέσματά τους είναι πολύ καλά, ειδικά στα FIT data. Ο BLR, ένας αλγόριθμος suitable για το πρόβλημα έχει πολύ καλά αποτελέσματα. Έχει κάποιο πρόβλημα στο precision σε κάποιες περιπτώσεις όπως και οι άλλοι δύο στο recall. Τέλος όσο αφορά τους πόρους και τα κόστη των αλγορίθμων, ο πιο ακριβός και ο πιο ακριβός είναι ο SOM ενώ ο BLR και ο Isolation Forest είναι δύο γρήγοροι αλγόριθμοι με χαμηλό υπολογιστικό κόστος.

9.4 Μελλοντική δουλειά

Μια μελλοντική δουλειά που θα μπορούσε να επιτευχθεί είναι η δοκιμή πολλαπλών αλγορίθμων για το συγκεκριμένο πρόβλημα. Θα έστρεφα την προσοχή μου σε αλγόριθμους supervised learning ειδικούς για προβλήματα binary classification όπως ανάφερα και προηγουμένως. Εννοείται πάντα υπάρχει χώρος για βελτίωση των αποτελεσμάτων (accuracy score, precision, recall) όπως επίσης και βελτίωση στο κόστος και χρόνο που απαιτούν οι αλγόριθμοι. Μέσα στα πλαίσια της πτυχιακής μου εργασίας προσπάθησα να πετύχω όσο το δυνατό καλύτερα αποτελέσματα μπορούσα και να φέρω εις πέρας δύο αλγορίθμους που να συμπεριφέρονται όσο καλύτερα γίνεται στο πρόβλημα που είχαμε να αντιμετωπίσουμε.

Μια άλλη επέκταση της δουλειάς θα ήταν η ανίχνευση και η αναγνώριση των επιθέσεων. Δηλαδή αν ανιχνευθεί επίθεση, ποιο είναι το είδος της επίθεσης που ανιχνεύθηκε. Επίσης θα ήταν ενδιαφέρον η υλοποίηση και ανάπτυξη τεχνικών αντιμετώπισης των επιθέσεων και πως θα μπορούσαμε να αντιμετωπίσουμε την ανίχνευση αυτών των επιθέσεων σε ένα IoT δίκτυο.

Τέλος θα θέλαμε να πραγματοποιηθεί ανίχνευση σε ένα πραγματικό περιβάλλον IoT, καθώς αυτό είναι σε λειτουργία. Δηλαδή να υλοποιηθούν μοντέλα που λαμβάνουν πληροφορία μέσω ενός ζωντανού data stream και να την αναλύουν σε πραγματικό χρόνο.

Βιβλιογραφία

[1] Elie Kfoury, Julien Saab, Paul Younes & Roger Achkar, “A Self Organizing Map Intrusion Detection System for RPL Protocol Attacks”, Department of Computer and Communications Engineering American University of Science and Technology, Beirut, January 2019.

[2] Christiana Nicolaou and Vasos Vassiliou, “Experimentation with Local Intrusion Detection in IoT Networks Using Supervised Learning”, Department of Computer Science, University of Cyprus, and RISE - Research Center on Interactive Media, Smart Systems and Emerging Technologies, Nicosia, Cyprus, 2020.

[3] Kiswar Sadaf & Jabeen Sultana, “Intrusion Detection Based on Autoencoder and Isolation Forest in Fog Computing”, Department of Computer Science, College of Computer and Information Sciences, Majmaah University, Al Majma'ah 11952, Saudi Arabia, Sep 23, 2020.

[4] Christiana Ioannou and Vasos Vassiliou, “Accurate Detection of Sinkhole Attacks in IoT Networks Using Local Agents”, Department of Computer Science, University of Cyprus and RISE - Research Center on Interactive Media, Smart Systems and Emerging Technologies Nicosia, Cyprus, pp 3-4.

[5] C. Ioannou, V. Vassiliou, and C. Sergiou, “RMT: A Wireless Sensor Network Monitoring Tool,” in Proceedings of the 13th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, ser. PE-WASUN ’16, 2016, pp. 45–49.

[6] A.-u. Rehman, S. U. Rehman, and H. Raheem, “Sinkhole attacks in wireless sensor networks: A survey,” Wireless Personal Communications, vol. 106, no. 4, pp. 2291–2313, Jun 2019.

[7] C. Ioannou and V. Vassiliou, “The Impact of Network Layer Attacks in Wireless Sensor Networks,” in International Workshop on Secure Internet of Things (SIoT 2016), Crete, Greece, Sep. 2016.

[8] Shoukat Ali, Dr. Muazzam A Khan, Jawad Ahmad, Asad W. Malik, and Anis ur Rehman, “Detection and Prevention of Black Hole Attacks in IOT & WSN”, Department of Computing, SEECS, National University of Sciences and Technology, Islamabad, Pakistan, 2018, pp 2-3.

[9] Fatma Gara, Leila Ben Saad, Rahma Ben Ayed, “An Intrusion Detection System for Selective Forwarding Attack in IPv6-based Mobile WSNs”, University of Tunis El Manar, ENIT, RISC, BP. 37 Le Belvedere, 1002 Tunis, Tunisia, pp 1-2.

[10] Kiswar Sadaf & Jabeen Sultana, “Intrusion Detection Based on Autoencoder and Isolation Forest in Fog Computing”, Department of Computer Science, College of Computer and Information Sciences, Majmaah University, Al Majma'ah 11952, Saudi Arabia, Sep 23, 2020, pp 5-6.

[11] F. Tony Liu, K. Ming Ting, and Z.-H. Zhou, ``Isolation forest ICDM08," in Proc. ICDM, 2008,pp.413-422.[Online].Available: https://cs.nju.edu.cn/zhouzh/zhouzh.les/publication/icdm08b.pdf%0Ahttps://cs.nju.edu.cn/zhouzh/zhouzh.les/publication/icdm08b.pdf?Pq=isolation forest.

Παράρτημα Α

Σε αυτό το παράρτημα θα δούμε τον κώδικα για τον Isolation Forest για τα FIT DATA. Δεν χρειάζεται να παραθέσω τους κώδικες για τα COOJA DATA εφόσον είναι ο ίδιος με απλά διαφορετικά input files.

Για το λόγο ότι ο κώδικας είναι ο ίδιος για τις επιθέσεις, το μόνο που αλλάζει είναι το file που διαβάζει τα δεδομένα, θα παραθέτω μόνο το BlackHole BN attack. Επίσης ο κώδικας για τις ανιχνεύσεις σε local και network detection είναι σχεδόν ίδιος με την διαφορά ότι στο network detection προθέτονται και τα δεδομένα των άλλων benign nodes. Εξάλλου οι κώδικες θα παραδοθούν οπόταν μπορείτε να τους ελέγξετε.

Άρα πιο κάτω θα δούμε τον κώδικα ανίχνευσης BlackHole_BN σε τοπολογίες middle και top στο local detection για τα FIT DATA. Θεωρώ σπατάλη χρόνου και τόπου να παραθέσω και τους κώδικες για τις άλλες επιθέσεις επειδή είναι ίδιοι οπόταν θα πάρει πάρα πολλές σελίδες για να βάλω όλους τους κώδικες από όλα τα σενάρια.

Local Detection middle Topology

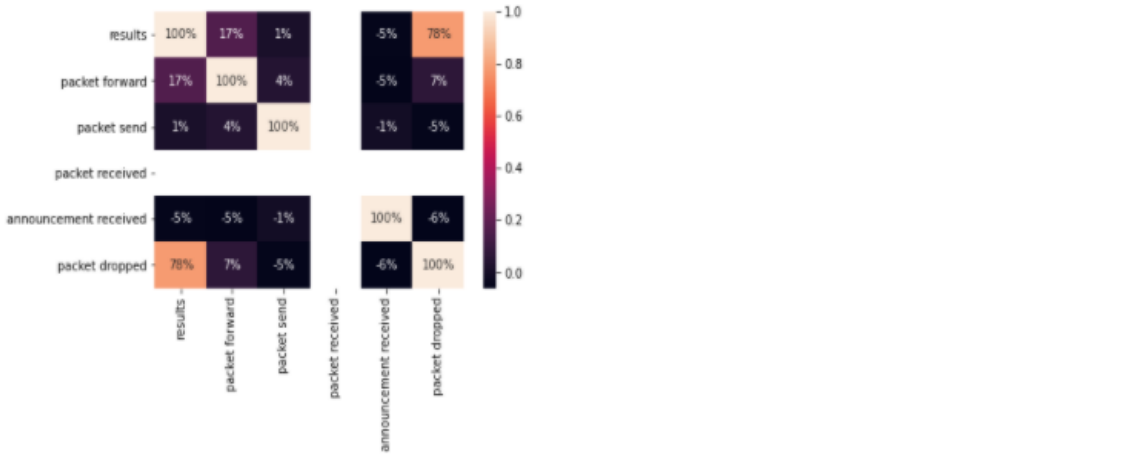
BlackHole_BN_evaluation.py

In [1]:
```
#import the liabries we need
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt #for charts
import seaborn as sns
from sklearn.ensemble import IsolationForest
import pickle
import shutil
```

In [2]:
```
#Take the 80% of benign and malicious txt and concatenate them. Then take 20% of each again and concatenate
from sklearn.model_selection import train_test_split
benign = np.loadtxt("data/benign_middle.txt", dtype='int')
malicious = np.loadtxt("data/BlackHole/SF_BH_BN_middle_malicious.txt", dtype='int')
X_benign = benign[:, 1:]
Y_benign = benign[:, 0]
X_malicious = malicious[:, 1:]
Y_malicious = malicious[:, 0]

X = np.vstack((X_benign, X_malicious))
y = np.hstack((Y_benign, Y_malicious))

X_trainB, X_testB, Y_trainB, Y_testB = train_test_split(X_benign, Y_benign, test_size=0.2,random_state=0)
X_trainM, X_testM, Y_trainM, Y_testM = train_test_split(X_malicious, Y_malicious, test_size=0.2,random_state=0)

X_train = np.concatenate((X_trainB,X_trainM),axis=0)
X_test = np.concatenate((X_testB,X_testM), axis=0)
Y_train = np.concatenate((Y_trainB,Y_trainM), axis=0)
Y_test= np.concatenate((Y_testB,Y_testM), axis=0)

X_train = np.delete(X_train,2,1)
X_test = np.delete(X_test,2,1)
Y_test = Y_test.reshape(-1,1)
```

In [3]:
```
#correlation matrix showing the correlation between features
df = pd.DataFrame(np.concatenate((benign,malicious),axis=0), columns = ['results','packet forward','packet send',
                                                                         'packet received','announcement received',
                                                                        'packet dropped'])
sns.heatmap(df.corr(),annot = True, fmt= ".0%")
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x1b759dfab80>



In [4]:
```
df_malicious = pd.DataFrame(malicious, columns = ['results','packet forward','packet send',
                                                  'packet received','announcement received',
                                                  'packet dropped'])
```

In [5]:
```
df_benign = pd.DataFrame(benign, columns = ['results','packet forward','packet send',
                                            'packet received','announcement received',
                                            'packet dropped'])
```

In [6]:
```
df_malicious
```

Out[6]:

	results	packet forward	packet send	packet received	announcement received	packet dropped
0	1	5	59	0	0	0
1	1	2	60	0	0	4
2	1	0	60	0	0	6
3	1	1	60	0	0	4
4	1	2	60	0	0	4
...	...	...	...	...	...	...
307	1	1	59	0	0	5
308	1	3	60	0	0	2
309	1	3	59	0	0	3
310	1	0	60	0	0	5
311	1	3	60	0	0	3

312 rows × 6 columns

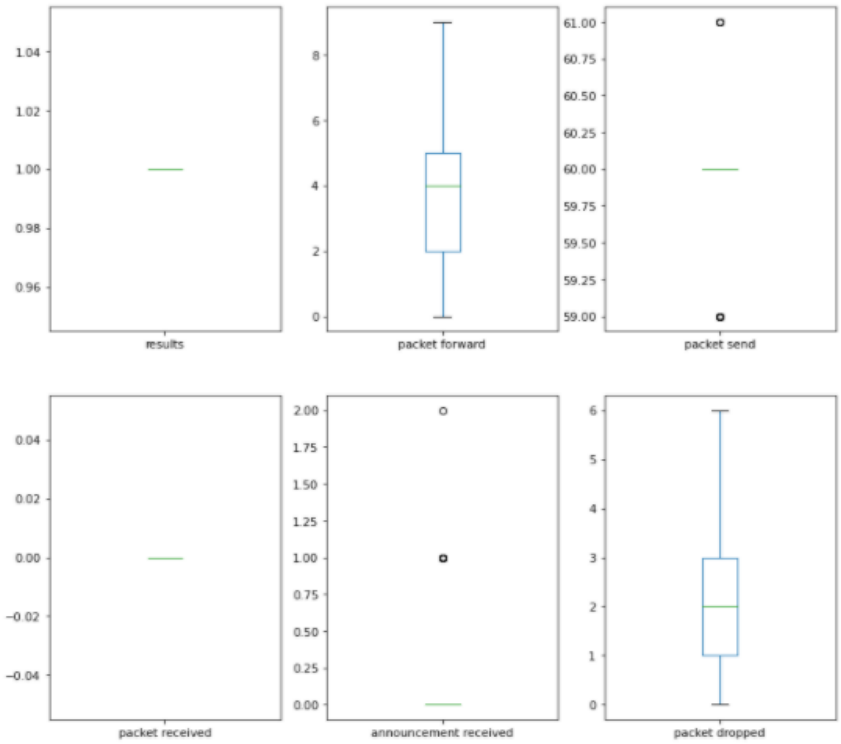
In [7]:
```
df_benign
```

Out[7]:

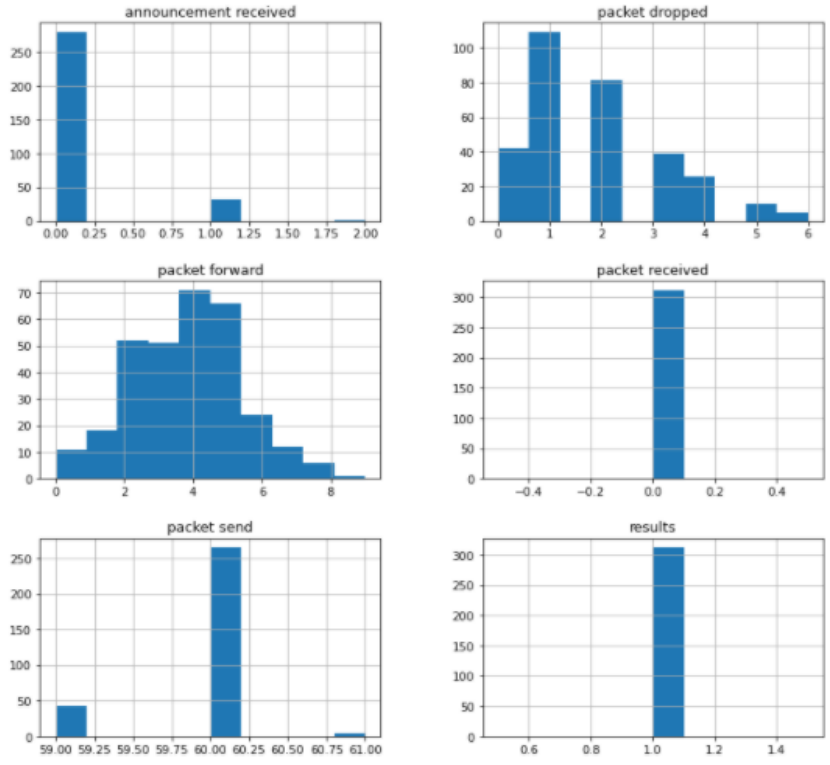
	results	packet forward	packet send	packet received	announcement received	packet dropped
0	0	3	59	0	1	0
1	0	0	59	0	0	0
2	0	3	59	0	1	0
3	0	3	59	0	0	0
4	0	3	60	0	1	0
...	...	...	...	...	...	...
3102	0	3	60	0	0	0
3103	0	0	60	0	1	0
3104	0	0	59	0	0	0
3105	0	6	59	0	0	0
3106	0	0	60	0	0	0

3107 rows × 6 columns

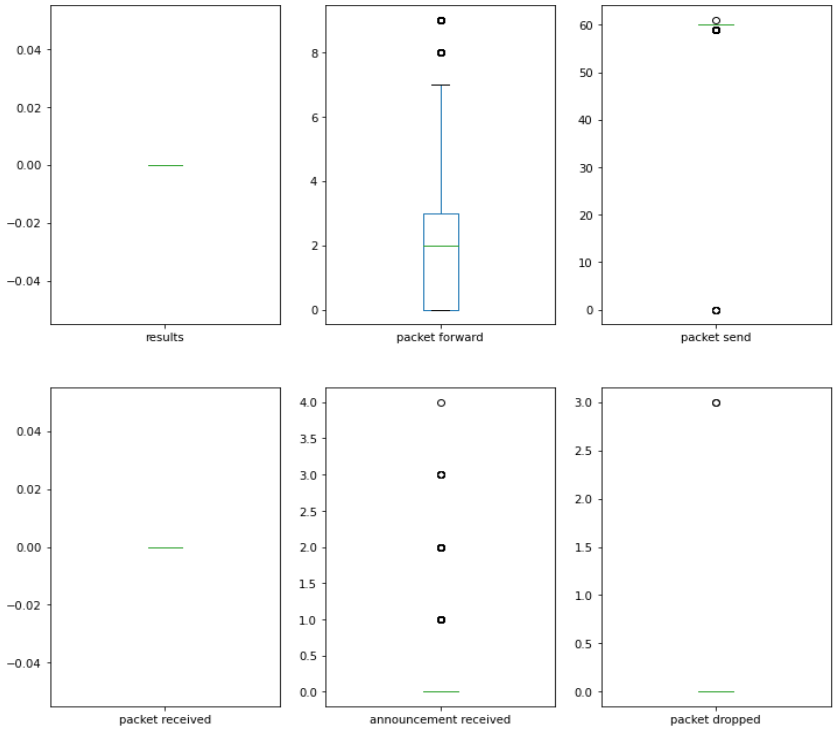
In [8]:
```
# scatter plot matrix showing if we have outliers and the data values
df_malicious.plot(kind='box', subplots=True,layout=(2,3), sharex = False, sharey = False, figsize=(12,12))
plt.show()
```



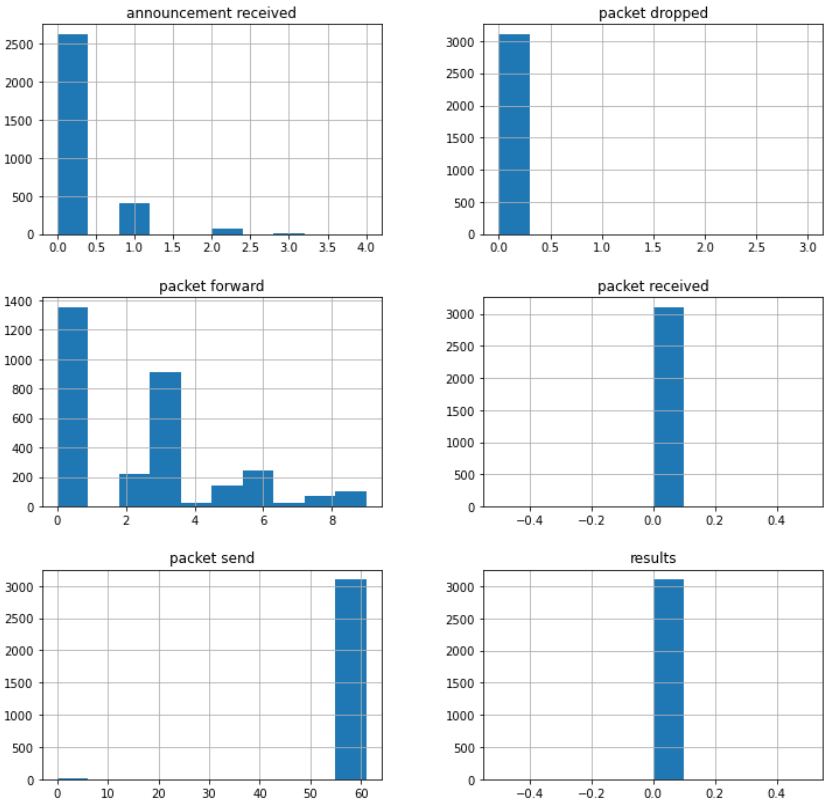
In [9]:
```
# plot for malicious node showing the value of the features of the data points. This plots are important in order to understand
# the results we will take at the end
df_malicious.hist(figsize = (12,12))
plt.show()
```



In [10]:
```
# scatter plot matrix
df_benign.plot(kind='box', subplots=True,layout=(2,3), sharex = False, sharey = False, figsize=(12,12))
plt.show()
```



In [11]:
```
df_benign.hist(figsize = (12,12))
plt.show()
```



In [12]:
```
df
```

Out[12]:

	results	packet forward	packet send	packet received	announcement received	packet dropped
0	0	3	59	0	1	0
1	0	0	59	0	0	0
2	0	3	59	0	1	0
3	0	3	59	0	0	0
4	0	3	60	0	1	0
...	...	...	...	...	...	...
3414	1	1	59	0	0	5
3415	1	3	60	0	0	2
3416	1	3	59	0	0	3
3417	1	0	60	0	0	5
3418	1	3	60	0	0	3

3419 rows × 6 columns

In [13]:
```
# Check if we have any empty values in our dateset
print(df.isnull().sum())
```

```
results                 0
packet forward          0
packet send             0
packet received         0
announcement received   0
packet dropped          0
dtype: int64
```

In [14]:
```
# Check for outliers
print(df.min())
```

```
results                 0
packet forward          0
packet send             0
packet received         0
announcement received   0
packet dropped          0
dtype: int32
```

In [15]:
```
print(df.max())
```

```
results                  1
packet forward           9
packet send             61
packet received          0
announcement received    4
packet dropped           6
dtype: int32
```

In [16]:
```
print(df.mean())
```

```
results                  0.091255
packet forward           2.439895
packet send             59.765136
packet received          0.000000
announcement received    0.175782
packet dropped           0.169055
```

In [17]:
```
print((df == 0).sum())
```

```
results                3107
packet forward         1365
packet send               6
packet received        3419
announcement received  2902
packet dropped         3147
dtype: int64
```

In [10]:
```
df[df['packet send'] == 0]
```

Out[10]:

	results	packet forward	packet send	packet received	announcement received	packet dropped
1678	0	0	0	0	0	3
1702	0	0	0	0	0	3
1816	0	0	0	0	0	0
1839	0	0	0	0	0	0
2964	0	0	0	0	0	0
2988	0	0	0	0	0	0

In [11]:
```
X_train
```

Out[11]:
```
array([[ 6, 60,  0,  0],
       [ 3, 60,  0,  0],
       [ 0, 60,  0,  0],
       ...,
       [ 8, 60,  0,  1],
       [ 5, 60,  0,  2],
       [ 5, 60,  0,  0]])
```

In [12]:
```
# Feature scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [13]:
```
X_train
```

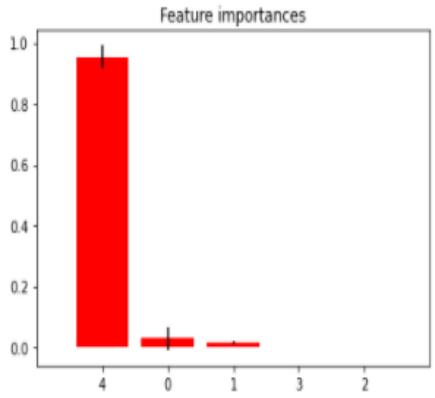
Out[13]:
```
array([[0.66666667, 0.98360656, 0.        , 0.        ],
       [0.33333333, 0.98360656, 0.        , 0.        ],
       [0.        , 0.98360656, 0.        , 0.        ],
       ...,
       [0.88888889, 0.98360656, 0.        , 0.16666667],
       [0.55555556, 0.98360656, 0.        , 0.33333333],
       [0.55555556, 0.98360656, 0.        , 0.        ]])
```

In [14]:
```
# Feature Selection
from sklearn.ensemble import ExtraTreesClassifier
estimator = ExtraTreesClassifier(n_estimators=100, max_features=5, random_state=0)
estimator.fit(X, y)
importances = estimator.feature_importances_
importances
```

Out[14]:
```
array([0.02906064, 0.01467845, 0.        , 0.00204962, 0.9542113 ])
```

In [40]:
```
# Feature Importance 1
# Use ensemble method: The goal of ensemble methods is to combine the
# predictions of several base estimators built with a given learning algorithm
# in order to improve generalizability / robustness over a single estimator.
# http://scikit-learn.org/stable/modules/ensemble.html
# Build an estimator (forest of trees) and compute the feature importances
# n_estimators = number of trees in forest

X = df.iloc[:, 1:].values
y = df.iloc[:, 0].values

estimator = ExtraTreesClassifier(n_estimators=100, max_features=5, random_state=0)
estimator.fit(X,y)
# Lets get the feature importances. Features with high importance score higher.
importances = estimator.feature_importances_
std = np.std([tree.feature_importances_ for tree in estimator.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")
for f in range(X.shape[1]):
    print("%d. feature %s (%f)" % (f + 1, df.drop(columns=['results']).columns[indices[f]], importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
       color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.show()
```

```
Feature ranking:
1. feature packet dropped (0.954211)
2. feature packet forward (0.029061)
3. feature packet send (0.014678)
4. feature announcement received (0.002050)
5. feature packet received (0.000000)
```



In [33]:
```
#hyperparameter tuning
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import make_scorer, f1_score

Y_train = np.where(Y_train==1,-1,Y_train)
Y_train = np.where(Y_train==0,1,Y_train)
Y_test = np.where(Y_test==1,-1,Y_test)
Y_test = np.where(Y_test==0,1,Y_test)

f1sc = make_scorer(f1_score, average='micro')


param_grid = {'contamination': [0.06, 0.07, 0.08, 0.09, 0.1, 0.11],
              'max_samples': [100,150,200,250],
              'n_estimators': [100,150,200,250],
              'max_features': [1,2,3]
             }

iforest = IsolationForest(random_state=0)
clf = GridSearchCV(estimator=iforest, param_grid=param_grid, scoring=f1sc, cv=5,
                   return_train_score=False)

clf.fit(X_train, Y_train)
```

Out[33]:
```
GridSearchCV(cv=5, estimator=IsolationForest(random_state=0),
             param_grid={'contamination': [0.06, 0.07, 0.08, 0.09, 0.1, 0.11],
                         'max_features': [1, 2, 3],
                         'max_samples': [100, 150, 200, 250],
                         'n_estimators': [100, 150, 200, 250]},
             scoring=make_scorer(f1_score, average=micro))
```

In [34]:
```
df_param = pd.DataFrame(clf.cv_results_)
df_param.head(7)
```

Out[34]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_contamination	param_max_features	param_max_samples	param_n_estimators	
0	0.138626	0.001342	0.028026	1.168008e-07	0.06	1	100	100	{'contan 'max_f
1	0.206807	0.000811	0.041641	3.738288e-04	0.06	1	100	150	{'contan 'max_f
2	0.278527	0.003374	0.055469	4.830166e-04	0.06	1	100	200	{'contan 'max_f
3	0.345924	0.002886	0.069947	8.661851e-04	0.06	1	100	250	{'contan 'max_f
4	0.139526	0.002156	0.027825	4.004479e-04	0.06	1	150	100	{'contan 'max_f
5	0.213798	0.006323	0.042238	7.489782e-04	0.06	1	150	150	{'contan 'max_f
6	0.286990	0.008844	0.057553	2.239681e-03	0.06	1	150	200	{'contan 'max_f

In [35]:
```
df_param[['param_contamination','param_max_features', 'param_max_samples', 'param_n_estimators', 'mean_test_score']]
```

Out[35]:

	param_contamination	param_max_features	param_max_samples	param_n_estimators	mean_test_score
0	0.06	1	100	100	0.913195
1	0.06	1	100	150	0.915391
2	0.06	1	100	200	0.911730
3	0.06	1	100	250	0.911730
4	0.06	1	150	100	0.915393
...	...	...	...	...	...
283	0.11	3	200	250	0.868590
284	0.11	3	250	100	0.870787
285	0.11	3	250	150	0.869322
286	0.11	3	250	200	0.869321
287	0.11	3	250	250	0.868956

288 rows × 5 columns

In [36]:
```
# Best score
clf.best_score_
```

Out[36]:
```
0.9403017457862065
```

In [37]:
```
# Best parameters
clf.best_params_
```

Out[37]:
```
{'contamination': 0.06,
 'max_features': 1,
 'max_samples': 250,
 'n_estimators': 100}
```

In [76]:
```
df_param.loc[(df_param['param_contamination'] == 0.07) & (df_param['param_max_samples'] == 250) &
             (df_param['param_n_estimators'] == 150) & (df_param['param_max_features'] == 1)]
```

Out[76]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_contamination	param_max_features	param_max_samples	param_n_estimators	
61	0.20979	0.00273	0.041638	0.000801	0.07	1	250	150	{'conta 'max_

In [66]:
```
df_param.loc[(df_param['param_contamination'] == 0.08) & (df_param['param_max_samples'] == 250) &
             (df_param['param_n_estimators'] == 150) & (df_param['param_max_features'] == 1)]
```

Out[66]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_contamination	param_max_features	param_max_samples	param_n_estimators	
109	0.214681	0.005893	0.043113	0.001769	0.08	1	250	150	{'cont 'ma

In [67]:
```
df_param.loc[(df_param['param_contamination'] == 0.09) & (df_param['param_max_samples'] == 250) &
             (df_param['param_n_estimators'] == 150) & (df_param['param_max_features'] == 1)]
```

In [15]:
```
# Train the model
forest = IsolationForest(contamination=0.08,n_estimators=150,random_state=0,max_features=1, max_samples=250)
forest.fit(X_train)
```

Out[15]:
```
IsolationForest(contamination=0.08, max_features=1, max_samples=250,
                n_estimators=150, random_state=0)
```

In [56]:
```
#save the trained model to be used for top topology
filename = 'blackHoleBN_iforest.sav'
pickle.dump(forest, open(filename, 'wb'))

#move the file for top topology
original = "blackHoleBN_iforest.sav"
target =  r"C:\Users\steve\OneDrive\jupyter\Essay\Isolation Forest\FIT_DATA\localDetection\top"
shutil.move(original,target)
```

Out[56]:
```
'C:\\Users\\steve\\OneDrive\\jupyter\\Essay\\Isolation Forest\\FIT_DATA\\localDetection\\top\\blackHoleBN_iforest.sav'
```

In [41]:
```
Y_test = np.where(Y_test==1,0,Y_test)
Y_test = np.where(Y_test==-1,1,Y_test)
```

In [16]:
```
y_pred = forest.predict(X_test)
y_pred = np.where(y_pred==1,0,y_pred)
y_pred = np.where(y_pred==-1,1,y_pred)
```

In [17]:
```
# Take the results
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm = confusion_matrix(Y_test,y_pred)
ac = accuracy_score(Y_test,y_pred)
cr = classification_report(Y_test, y_pred)
print("classification report\n", cr)
print("confusion matrix\n", cm)
print("\naccuracy score\n", ac)
```

```
classification report
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       622
           1       0.91      0.81      0.86        63

    accuracy                           0.98       685
   macro avg       0.95      0.90      0.92       685
weighted avg       0.97      0.98      0.97       685

confusion matrix
 [[617   5]
 [ 12  51]]

accuracy score
 0.9751824817518249
```

In [66]:
```
#create a dataframe for test datas
df_test = pd.DataFrame(np.concatenate((X_test,Y_test),axis=1), columns = ['packet forward','packet received',
                                                                          'announcement received',
                                                                              'packet dropped', 'results'])
df_test.describe()
```

Out[66]:

	packet forward	packet received	announcement received	packet dropped	results
count	685.000000	685.000000	685.000000	685.000000	685.000000
mean	0.272019	0.981237	0.046715	0.029197	0.091971
std	0.277141	0.005836	0.117105	0.119728	0.289196
min	0.000000	0.967213	0.000000	0.000000	0.000000
25%	0.000000	0.983607	0.000000	0.000000	0.000000
50%	0.333333	0.983607	0.000000	0.000000	0.000000
75%	0.333333	0.983607	0.000000	0.000000	0.000000
max	1.000000	1.000000	0.750000	1.000000	1.000000

In [67]:
```
df_test['health'] = forest.decision_function(X_test)
df_test['anomaly'] = y_pred
df_test
```

Out[67]:

	packet forward	packet received	announcement received	packet dropped	results	health	anomaly
0	0.000000	0.983607	0.00	0.000000	0.0	0.076226	0
1	1.000000	0.983607	0.00	0.000000	0.0	0.033690	0
2	0.666667	0.983607	0.00	0.000000	0.0	0.060504	0
3	0.666667	0.983607	0.00	0.000000	0.0	0.060504	0
4	0.333333	0.983607	0.00	0.000000	0.0	0.083373	0
...	...	...	...	...	...	...	...
680	0.333333	0.983607	0.25	0.000000	1.0	0.065860	0
681	0.555556	0.983607	0.00	0.166667	1.0	-0.011154	1
682	0.555556	0.983607	0.00	0.166667	1.0	-0.011154	1
683	0.555556	0.983607	0.00	0.166667	1.0	-0.011154	1
684	0.888889	0.983607	0.00	0.166667	1.0	-0.036889	1

685 rows × 7 columns

In [68]:
```
#Check the false negatives
false_positive = df_test[(df_test['anomaly'] == 0) & (df_test['results'] == 1)]
false_positive.describe()
```

Out[68]:

	packet forward	packet received	announcement received	packet dropped	results	health	anomaly
count	12.000000	12.000000	12.000000	12.000000	12.0	12.000000	12.0
mean	0.453704	0.982240	0.020833	0.125000	1.0	0.036601	0.0
std	0.167227	0.004732	0.072169	0.160885	0.0	0.027025	0.0
min	0.333333	0.967213	0.000000	0.000000	1.0	0.011356	0.0
25%	0.333333	0.983607	0.000000	0.000000	1.0	0.011356	0.0
50%	0.333333	0.983607	0.000000	0.000000	1.0	0.023859	0.0
75%	0.583333	0.983607	0.000000	0.333333	1.0	0.060504	0.0
max	0.777778	0.983607	0.250000	0.333333	1.0	0.083373	0.0

In [71]:
```
#Check the false positives
false_positive = df_test[(df_test['anomaly'] == 0) & (df_test['results'] == 1)]
false_positive
```

Out[71]:

	packet forward	packet received	announcement received	packet dropped	results	health	anomaly
642	0.666667	0.983607	0.00	0.000000	1.0	0.060504	0
644	0.777778	0.983607	0.00	0.000000	1.0	0.028460	0
652	0.333333	0.983607	0.00	0.000000	1.0	0.083373	0
653	0.333333	0.983607	0.00	0.333333	1.0	0.011356	0
655	0.666667	0.983607	0.00	0.000000	1.0	0.060504	0
657	0.555556	0.983607	0.00	0.000000	1.0	0.059450	0
658	0.444444	0.967213	0.00	0.000000	1.0	0.019259	0
670	0.333333	0.983607	0.00	0.166667	1.0	0.016382	0
672	0.333333	0.983607	0.00	0.333333	1.0	0.011356	0
676	0.333333	0.983607	0.00	0.333333	1.0	0.011356	0
678	0.333333	0.983607	0.00	0.333333	1.0	0.011356	0
680	0.333333	0.983607	0.25	0.000000	1.0	0.065860	0

In [74]:
```
#Compare with benign (True negatives)
true_negatives = df_test[df_test['results'] == 0]
true_negatives
```

Out[74]:

	packet forward	packet received	announcement received	packet dropped	results	health	anomaly
0	0.000000	0.983607	0.00	0.0	0.0	0.076226	0
1	1.000000	0.983607	0.00	0.0	0.0	0.033690	0
2	0.666667	0.983607	0.00	0.0	0.0	0.060504	0
3	0.666667	0.983607	0.00	0.0	0.0	0.060504	0
4	0.333333	0.983607	0.00	0.0	0.0	0.083373	0
...	...	...	...	...	...	...	...
617	0.222222	0.983607	0.00	0.0	0.0	0.060430	0
618	0.888889	0.967213	0.25	0.0	0.0	-0.013522	1
619	0.000000	0.983607	0.00	0.0	0.0	0.076226	0
620	0.888889	0.983607	0.00	0.0	0.0	0.037092	0
621	0.333333	0.983607	0.00	0.0	0.0	0.083373	0

622 rows × 7 columns

In [62]:
```
#Check the true negatives
true_positive = df_test[(df_test['anomaly'] == 0) & (df_test['results'] == 0)]
true_positive.describe()
```

Out[62]:

	packet forward	packet received	announcement received	packet dropped	results	health	anomaly
count	617.000000	617.000000	617.000000	617.0	617.0	617.000000	617.0
mean	0.256798	0.981189	0.044976	0.0	0.0	0.064911	0.0
std	0.276703	0.005818	0.111727	0.0	0.0	0.018535	0.0
min	0.000000	0.967213	0.000000	0.0	0.0	0.000000	0.0
25%	0.000000	0.983607	0.000000	0.0	0.0	0.058431	0.0
50%	0.333333	0.983607	0.000000	0.0	0.0	0.076226	0.0
75%	0.333333	0.983607	0.000000	0.0	0.0	0.076226	0.0
max	1.000000	0.983607	0.750000	0.0	0.0	0.083373	0.0

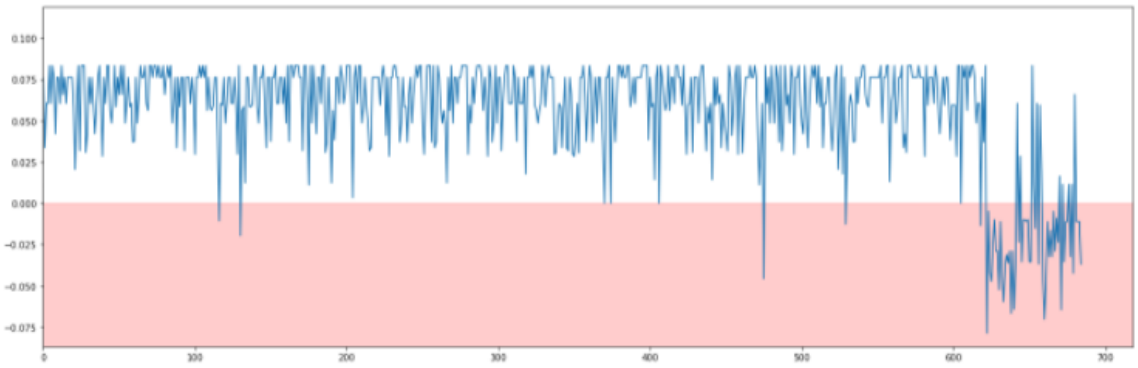
In [63]:
```
#Check the true negative
true_positive = df_test[(df_test['results'] == 0)]
true_positive.describe()
```

Out[63]:

	packet forward	packet received	announcement received	packet dropped	results	health	anomaly
count	622.000000	622.000000	622.000000	622.0	622.0	622.000000	622.000000
mean	0.259914	0.981103	0.048633	0.0	0.0	0.064224	0.008039
std	0.279496	0.005902	0.119320	0.0	0.0	0.020008	0.089369
min	0.000000	0.967213	0.000000	0.0	0.0	-0.045753	0.000000
25%	0.000000	0.983607	0.000000	0.0	0.0	0.056131	0.000000
50%	0.333333	0.983607	0.000000	0.0	0.0	0.076226	0.000000
75%	0.333333	0.983607	0.000000	0.0	0.0	0.076226	0.000000
max	1.000000	0.983607	0.750000	0.0	0.0	0.083373	1.000000

In [64]:
```
# Plot showing the outliers the model could find
fig, axes = plt.subplots(figsize=(22,7))
plt.plot(df_test['health'])
xmin, xmax = plt.xlim()
ymin, ymax = plt.ylim()
plt.xlim(xmin * 0, xmax * 1)
plt.ylim(ymin * 1, ymax * 1.3)
plt.axhspan(-0.18, 0.0, alpha=0.2, color ='red')
```

Out[64]: <matplotlib.patches.Polygon at 0x1756de85070>



Local Detection top Topology

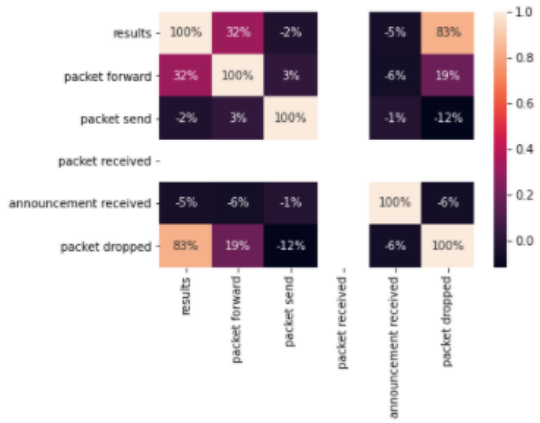
BlackHole_BN_evaluation.py

In [1]:
```
#import the liabries we need
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt #for charts
from sklearn.ensemble import IsolationForest
import seaborn as sns
import pickle
```

In [2]:
```
#Take the 80% of benign and malicious txt and concatenate them. Then take 20% of each again and concatenate
from sklearn.model_selection import train_test_split
benign = np.loadtxt("data/benign_top.txt", dtype='int')
malicious = np.loadtxt("data/BlackHole/SF_BH_BN_top_malicious.txt", dtype='int')

X_benign = benign[:, 1:]
Y_benign = benign[:, 0]
X_malicious = malicious[:, 1:]
Y_malicious = malicious[:, 0]

X = np.vstack((X_benign, X_malicious))
y = np.hstack((Y_benign, Y_malicious))

X_trainB, X_testB, Y_trainB, Y_testB = train_test_split(X_benign, Y_benign, test_size=0.2,random_state=0)
X_trainM, X_testM, Y_trainM, Y_testM = train_test_split(X_malicious, Y_malicious, test_size=0.2,random_state=0)
X_train = np.concatenate((X_trainB,X_trainM),axis=0)
X_test = np.concatenate((X_testB,X_testM), axis=0)
Y_train = np.concatenate((Y_trainB,Y_trainM), axis=0)
Y_test= np.concatenate((Y_testB,Y_testM), axis=0)

X_train = np.delete(X_train,2,1)
X_test = np.delete(X_test,2,1)
Y_test = Y_test.reshape(-1,1)
```

In [3]:
```
df = pd.DataFrame(np.concatenate((benign,malicious),axis=0), columns = ['results','packet forward','packet send',
                                                                         'packet received','announcement received',
                                                                        'packet dropped'])
sns.heatmap(df.corr(),annot = True, fmt= ".0%")
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x2dfe62b28b0>



In [4]:
```
df_malicious = pd.DataFrame(malicious, columns = ['results','packet forward','packet send',
                                                  'packet received','announcement received',
                                                  'packet dropped'])
```

In [5]:
```
df_benign = pd.DataFrame(benign, columns = ['results','packet forward','packet send',
                                            'packet received','announcement received',
                                            'packet dropped'])
```

In [6]:
```
df_malicious
```

Out[6]:

	results	packet forward	packet send	packet received	announcement received	packet dropped
0	1	5	59	0	0	1
1	1	2	60	0	0	4
2	1	2	59	0	0	3
3	1	6	60	0	0	0
4	1	2	59	0	0	3
...	...	...	...	...	...	...
305	1	1	60	0	0	2
306	1	3	60	0	0	0
307	1	2	60	0	0	1
308	1	0	60	0	0	3
309	1	1	60	0	0	2

310 rows × 6 columns

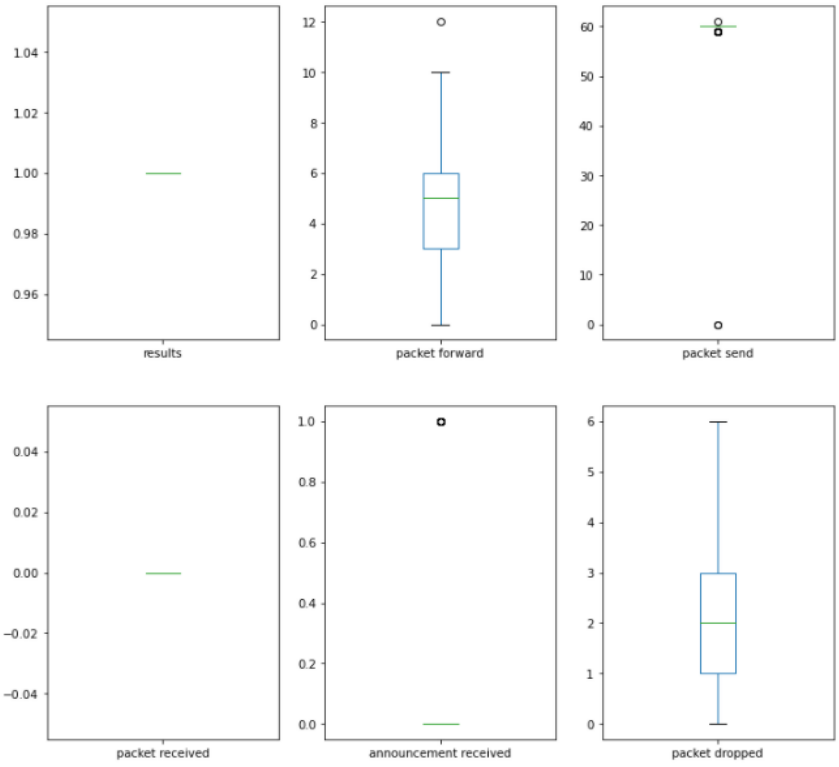
In [7]:
```
df_benign
```

Out[7]:

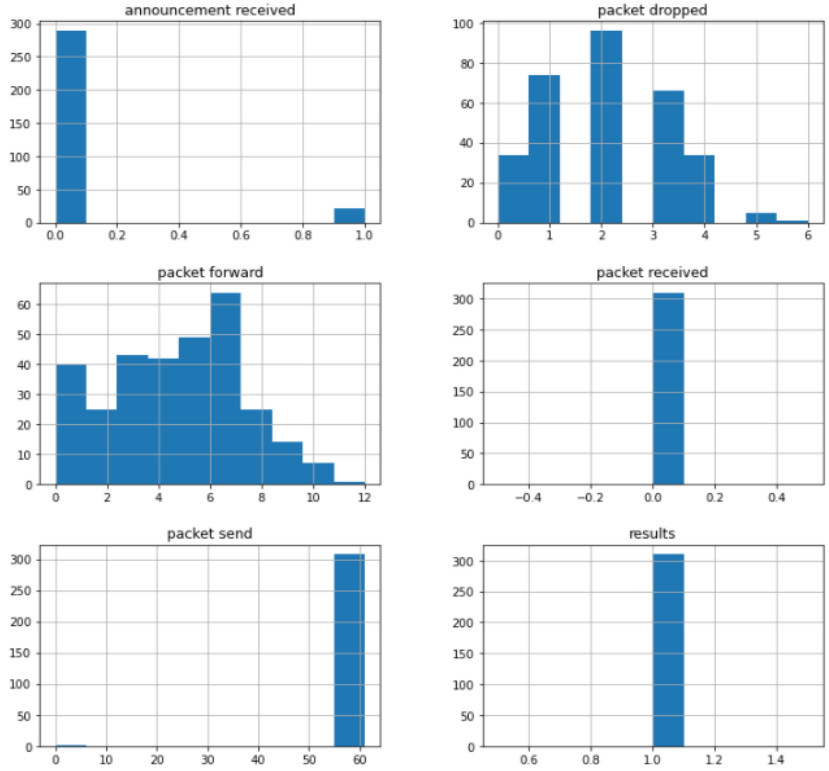
	results	packet forward	packet send	packet received	announcement received	packet dropped
0	0	5	59	0	0	0
1	0	5	59	0	1	0
2	0	0	59	0	1	0
3	0	2	59	0	1	0
4	0	0	59	0	0	0
...	...	...	...	...	...	...
3101	0	3	60	0	0	0
3102	0	6	59	0	1	0
3103	0	2	60	0	0	0
3104	0	2	59	0	0	0
3105	0	0	59	0	0	0

3106 rows × 6 columns

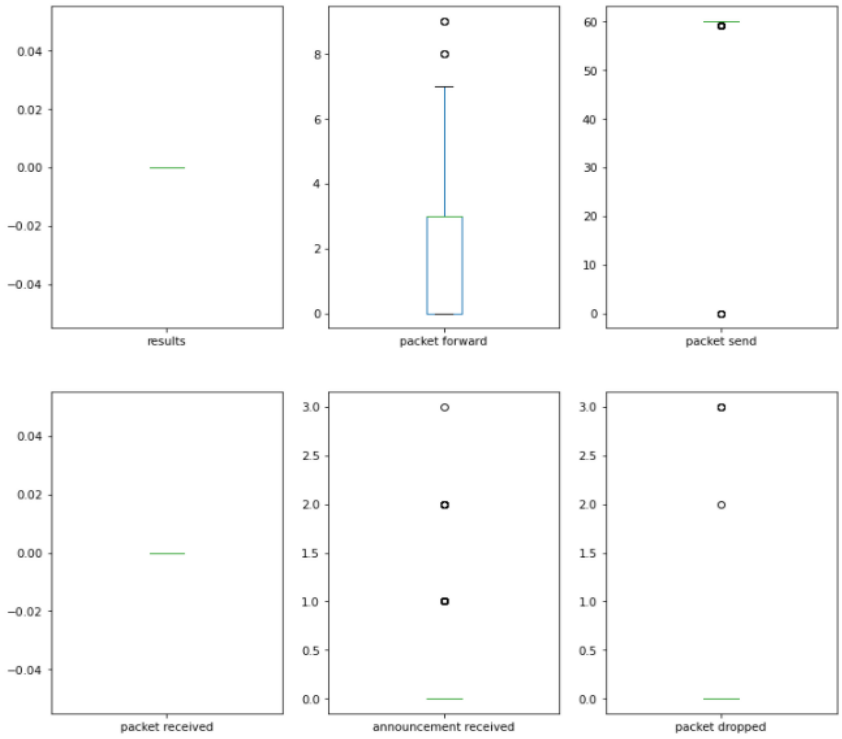
In [8]:
```
# scatter plot matrix
df_malicious.plot(kind='box', subplots=True,layout=(2,3), sharex = False, sharey = False, figsize=(12,12))
plt.show()
```



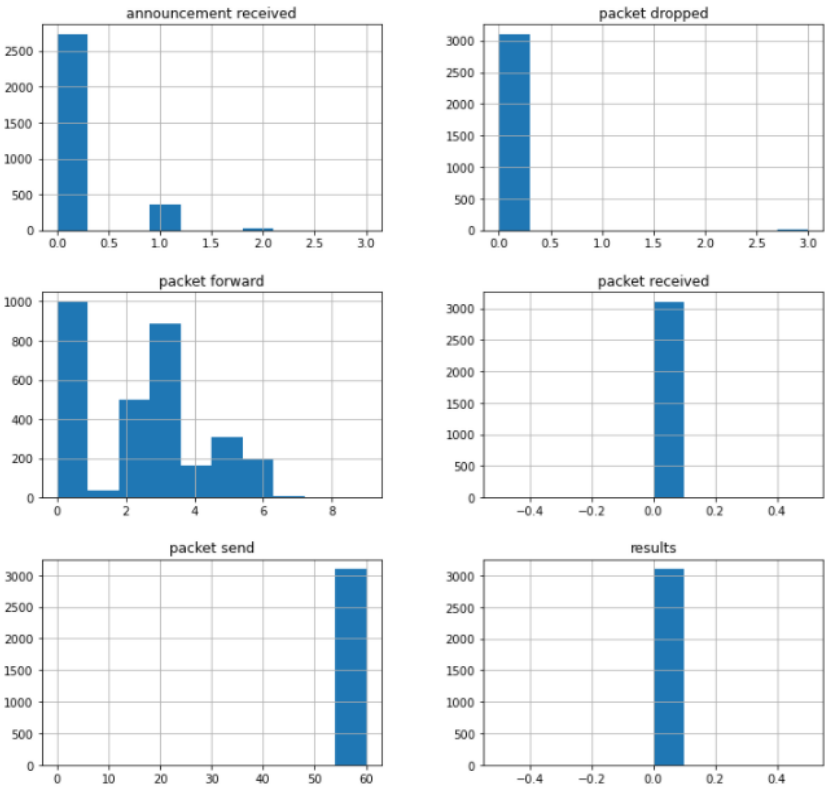
In [9]:
```
df_malicious.hist(figsize = (12,12))
plt.show()
```



In [10]:
```
# scatter plot matrix
df_benign.plot(kind='box', subplots=True,layout=(2,3), sharex = False, sharey = False, figsize=(12,12))
plt.show()
```



In [11]:
```
df_benign.hist(figsize = (12,12))
plt.show()
```



In [12]:
```
df
```

Out[12]:

	results	packet forward	packet send	packet received	announcement received	packet dropped
0	0	5	59	0	0	0
1	0	5	59	0	1	0
2	0	0	59	0	1	0
3	0	2	59	0	1	0
4	0	0	59	0	0	0
...	...	...	...	...	...	...
3411	1	1	60	0	0	2
3412	1	3	60	0	0	0
3413	1	2	60	0	0	1
3414	1	0	60	0	0	3
3415	1	1	60	0	0	2

3416 rows × 6 columns

In [13]:
```
print(df.isnull().sum())
```

```
results                 0
packet forward          0
packet send             0
packet received         0
announcement received   0
packet dropped          0
dtype: int64
```

In [14]:
```
print(df.min())
```

```
results                 0
packet forward          0
packet send             0
packet received         0
announcement received   0
packet dropped          0
dtype: int32
```

In [15]:
```
print(df.max())
```

```
results                  1
packet forward          12
packet send             61
packet received          0
announcement received    3
packet dropped           6
dtype: int32
```

In [16]:
```
print(df.mean())
```

```
results                  0.090749
packet forward           2.526054
packet send             59.665984
packet received          0.000000
announcement received    0.122365
packet dropped           0.188817
dtype: float64
```

In [17]:
```
print((df == 0).sum())
```

```
results                3106
packet forward         1015
packet send               7
packet received        3416
announcement received  3021
packet dropped         3135
dtype: int64
```

In [18]:
```
df[df['packet send'] == 0]
```

Out[18]:

	results	packet forward	packet send	packet received	announcement received	packet dropped
359	0	0	0	0	0	3
383	0	0	0	0	0	3
1844	0	0	0	0	0	3
2227	0	0	0	0	0	3
2251	0	0	0	0	0	2
3328	1	4	0	0	1	1
3329	1	5	0	0	0	0

In [19]:
```
X_train
```

Out[19]:
```
array([[ 3, 59,  0,  0],
       [ 0, 59,  0,  0],
       [ 3, 60,  0,  0],
       ...,
       [ 2, 59,  0,  2],
       [ 6, 60,  0,  1],
       [ 8, 60,  1,  2]])
```

In [20]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [21]:
```
X_train
```

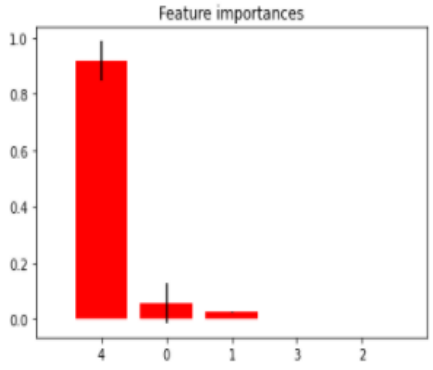
Out[21]:
```
array([[0.3       , 0.96721311, 0.        , 0.        ],
       [0.        , 0.96721311, 0.        , 0.        ],
       [0.3       , 0.98360656, 0.        , 0.        ],
       ...,
       [0.2       , 0.96721311, 0.        , 0.4       ],
       [0.6       , 0.98360656, 0.        , 0.2       ],
       [0.8       , 0.98360656, 0.5       , 0.4       ]])
```

In [22]:
```
loaded_model = pickle.load(open("blackHoleBN_iforest.sav", 'rb'))
```

In [23]:
```
y_pred = loaded_model.predict(X_test)
y_pred = np.where(y_pred==1,0,y_pred)
y_pred = np.where(y_pred==-1,1,y_pred)
```

In [27]:
```
# Feature Importance 1
# Use ensemble method: The goal of ensemble methods is to combine the
# predictions of several base estimators built with a given learning algorithm
# in order to improve generalizability / robustness over a single estimator.
# http://scikit-learn.org/stable/modules/ensemble.html
# Build an estimator (forest of trees) and compute the feature importances
# n_estimators = number of trees in forest

X = df.iloc[:, 1:].values
y = df.iloc[:, 0].values

estimator = ExtraTreesClassifier(n_estimators=100, max_features=5, random_state=0)
estimator.fit(X,y)
# Lets get the feature importances. Features with high importance score higher.
importances = estimator.feature_importances_
std = np.std([tree.feature_importances_ for tree in estimator.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")
for f in range(X.shape[1]):
    print("%d. feature %s (%f)" % (f + 1, df.drop(columns=['results']).columns[indices[f]], importances[indices[f]]))

# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
       color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), indices)
plt.xlim([-1, X.shape[1]])
plt.show()
```

```
Feature ranking:
1. feature packet dropped (0.916433)
2. feature packet forward (0.057472)
3. feature packet send (0.024277)
4. feature announcement received (0.001818)
5. feature packet received (0.000000)
```



In [24]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm = confusion_matrix(Y_test,y_pred)
ac = accuracy_score(Y_test,y_pred)
cr = classification_report(Y_test, y_pred)
print("classification report\n", cr)
print("confusion matrix\n", cm)
print("\naccuracy score\n", ac)

#LOAD TRAINED MODEL FROM MIDDLE AND WE HAVE BETTER RESULTS
```

```
classification report
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       622
           1       0.89      0.81      0.85        62

    accuracy                           0.97       684
   macro avg       0.94      0.90      0.92       684
weighted avg       0.97      0.97      0.97       684

confusion matrix
 [[616   6]
 [ 12  50]]

accuracy score
 0.9736842105263158
```

In [17]:
```
from sklearn.ensemble import ExtraTreesClassifier
estimator = ExtraTreesClassifier(n_estimators=100, max_features=5, random_state=0)
estimator.fit(X, y)
importances = estimator.feature_importances_
importances
```

Out[17]:
```
array([0.05747192, 0.02427662, 0.        , 0.00181806, 0.91643339])
```

In [20]:
```
#hyperparameter tuning
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import make_scorer, f1_score

Y_train = np.where(Y_train==1,-1,Y_train)
Y_train = np.where(Y_train==0,1,Y_train)
Y_test = np.where(Y_test==1,-1,Y_test)
Y_test = np.where(Y_test==0,1,Y_test)

f1sc = make_scorer(f1_score, average='micro')


param_grid = {'contamination': [0.06, 0.07, 0.08, 0.09],
              'max_samples': [100,150,200,250],
              'n_estimators': [100,150,200,250],
              'max_features': [1,2,3]
             }

iforest = IsolationForest(random_state=0)
clf = GridSearchCV(estimator=iforest, param_grid=param_grid, scoring=f1sc, cv=5,
                   return_train_score=False)

clf.fit(X_train, Y_train)
```

In [21]:
```
df_param = pd.DataFrame(clf.cv_results_)
df_param.head(7)
```

Out[21]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_contamination	param_max_features	param_max_samples	param_n_estimators	
0	0.136497	0.002218	0.028450	0.003285	0.06	1	100	100	{'contan 'max_f
1	0.205637	0.003580	0.042660	0.002825	0.06	1	100	150	{'contan 'max_f
2	0.279509	0.005590	0.053876	0.003997	0.06	1	100	200	{'contan 'max_f
3	0.345343	0.004476	0.069062	0.000671	0.06	1	100	250	{'contan 'max_f
4	0.139647	0.003869	0.027043	0.002678	0.06	1	150	100	{'contan 'max_f
5	0.208623	0.002844	0.040097	0.002945	0.06	1	150	150	{'contan 'max_f
6	0.277809	0.004754	0.054495	0.001811	0.06	1	150	200	{'contan 'max_f

In [22]:
```
df_param[['param_contamination','param_max_features', 'param_max_samples', 'param_n_estimators', 'mean_test_score']]
```

Out[22]:

	param_contamination	param_max_features	param_max_samples	param_n_estimators	mean_test_score
0	0.06	1	100	100	0.035163
1	0.06	1	100	150	0.041391
2	0.06	1	100	200	0.039558
3	0.06	1	100	250	0.041390
4	0.06	1	150	100	0.042489
...	...	...	...	...	...
187	0.09	3	200	250	0.067750
188	0.09	3	250	100	0.071046
189	0.09	3	250	150	0.067022
190	0.09	3	250	200	0.069949
191	0.09	3	250	250	0.070318

192 rows × 5 columns

In [23]:
```
clf.best_score_
```

Out[23]: 0.07104619938257964

In [24]:
```
clf.best_params_
```

Out[24]:
```
{'contamination': 0.09,
 'max_features': 3,
 'max_samples': 250,
 'n_estimators': 100}
```

In [76]:
```
df_param.loc[(df_param['param_contamination'] == 0.09) & (df_param['param_max_samples'] == 250) &
             (df_param['param_n_estimators'] == 100) & (df_param['param_max_features'] == 3)]
```

Out[76]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_contamination	param_max_features	param_max_samples	param_n_estimators	
188	0.176986	0.011944	0.036803	0.004036	0.09	3	250	100	{'cor 'ma

In [77]:
```
df_param.loc[(df_param['param_contamination'] == 0.08) & (df_param['param_max_samples'] == 250) &
             (df_param['param_n_estimators'] == 150) & (df_param['param_max_features'] == 1)]
```

Out[77]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_contamination	param_max_features	param_max_samples	param_n_estimators	
109	0.223896	0.00513	0.039818	0.006257	0.08	1	250	150	{'cor 'ma

In [25]:
```
forest = IsolationForest(contamination=0.09,n_estimators=150,random_state=0,max_features=1, max_samples=250)
forest.fit(X_train)
```

Out[25]:
```
IsolationForest(contamination=0.09, max_features=1, max_samples=250,
                n_estimators=150, random_state=0)
```

In [26]:
```
Y_test = np.where(Y_test==1,0,Y_test)
Y_test = np.where(Y_test==-1,1,Y_test)
```

In [26]:
```
y_pred = forest.predict(X_test)
y_pred = np.where(y_pred==1,0,y_pred)
y_pred = np.where(y_pred==-1,1,y_pred)
```

In [27]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm = confusion_matrix(Y_test,y_pred)
ac = accuracy_score(Y_test,y_pred)
cr = classification_report(Y_test, y_pred)
print("classification report\n", cr)
print("confusion matrix\n", cm)
print("\naccuracy score\n", ac)
```

```
classification report
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       622
           1       0.93      0.89      0.91        62

    accuracy                           0.98       684
   macro avg       0.96      0.94      0.95       684
weighted avg       0.98      0.98      0.98       684

confusion matrix
 [[618   4]
 [  7  55]]

accuracy score
 0.9839181286549707
```

In [28]:
```
#create a dataframe for test datas
df_test = pd.DataFrame(np.concatenate((X_test,Y_test),axis=1), columns = ['packet forward','packet send',
                                                                          'announcement received',
                                                                               'packet dropped', 'results'])
df_test.describe()
```

Out[28]:

	packet forward	packet send	announcement received	packet dropped	results
count	684.000000	684.000000	684.000000	684.000000	684.000000
mean	0.243713	0.978454	0.058942	0.036842	0.090643
std	0.208329	0.038091	0.175524	0.140859	0.287311
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.983607	0.000000	0.000000	0.000000
50%	0.300000	0.983607	0.000000	0.000000	0.000000
75%	0.300000	0.983607	0.000000	0.000000	0.000000
max	1.200000	0.983607	1.500000	1.200000	1.000000

In [29]:
```
df_test['health'] = forest.decision_function(X_test)
df_test['anomaly'] = y_pred
df_test
```

Out[29]:

	packet forward	packet send	announcement received	packet dropped	results	health	anomaly
0	0.3	0.967213	0.0	0.0	0.0	0.048208	0
1	0.5	0.983607	0.0	0.0	0.0	0.053722	0
2	0.0	0.983607	0.0	0.0	0.0	0.056287	0
3	0.3	0.983607	0.0	0.0	0.0	0.066789	0
4	0.3	0.967213	0.0	0.0	0.0	0.048208	0
...	...	...	...	...	...	...	...
679	0.4	0.983607	0.0	0.2	1.0	-0.042928	1
680	0.2	0.983607	0.0	0.8	1.0	-0.041709	1
681	0.6	0.983607	0.0	0.2	1.0	-0.042405	1
682	0.0	0.967213	0.0	0.6	1.0	-0.042020	1
683	0.7	0.983607	0.0	0.2	1.0	-0.079197	1

684 rows × 7 columns

In [30]:
```
#Check the false positives
false_positive = df_test[(df_test['anomaly'] == 0) & (df_test['results'] == 1)]
false_positive.describe()
#As we can see we can set a threshold on packets forward and packets dropped to identify better the predictions
```

Out[30]:

	packet forward	packet send	announcement received	packet dropped	results	health	anomaly
count	7.0	7.000000	7.000000	7.000000	7.0	7.000000	7.0
mean	0.4	0.976581	0.142857	0.114286	1.0	0.023642	0.0
std	0.1	0.008763	0.243975	0.195180	0.0	0.017221	0.0
min	0.3	0.967213	0.000000	0.000000	1.0	0.000000	0.0
25%	0.3	0.967213	0.000000	0.000000	1.0	0.011784	0.0
50%	0.4	0.983607	0.000000	0.000000	1.0	0.029372	0.0
75%	0.5	0.983607	0.250000	0.200000	1.0	0.034599	0.0
max	0.5	0.983607	0.500000	0.400000	1.0	0.043355	0.0

```
true_positive = df_test[(df_test['anomaly'] == 0) & (df_test['results'] == 0)]
true_positive.describe()
#As we can see we can set a threshold on packets forward and packets dropped to identify better the predictions
```

Out[31]:

	packet forward	packet send	announcement received	packet dropped	results	health	anomaly
count	618.000000	618.000000	618.000000	618.0	618.0	618.000000	618.0
mean	0.224272	0.979886	0.058252	0.0	0.0	0.049801	0.0
std	0.192551	0.006885	0.170341	0.0	0.0	0.013994	0.0
min	0.000000	0.967213	0.000000	0.0	0.0	0.000099	0.0
25%	0.000000	0.983607	0.000000	0.0	0.0	0.043129	0.0
50%	0.200000	0.983607	0.000000	0.0	0.0	0.056287	0.0
75%	0.300000	0.983607	0.000000	0.0	0.0	0.058503	0.0
max	0.700000	0.983607	1.500000	0.0	0.0	0.066789	0.0

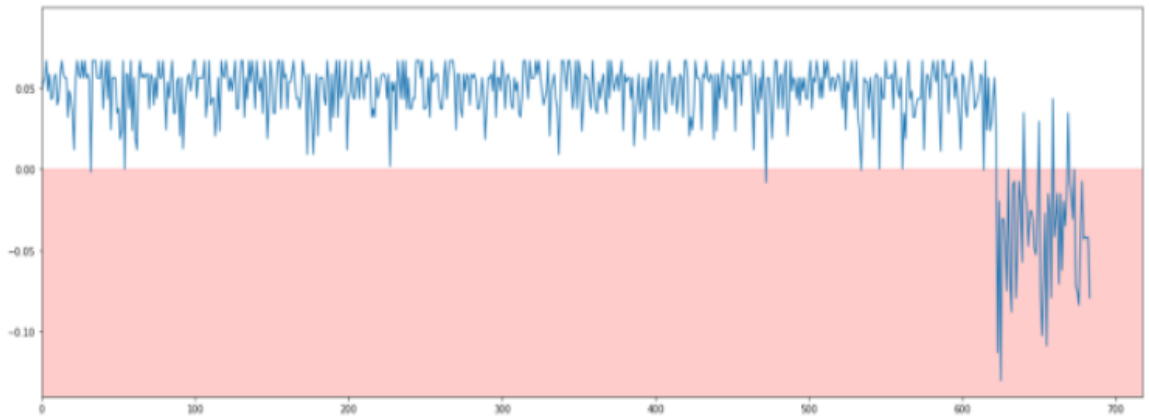
In [32]:
```
#Check the true negatives
true_negatives = df_test[(df_test['anomaly'] == 1) & (df_test['results'] == 1)]
true_negatives.describe()
#As we can see we can set a threshold on packets forward and packets dropped to identify better the predictions
```

Out[32]:

	packet forward	packet send	announcement received	packet dropped	results	health	anomaly
count	55.000000	55.000000	55.000000	55.000000	55.0	55.000000	55.0
mean	0.436364	0.963040	0.027273	0.443636	1.0	-0.045818	1.0
std	0.265528	0.132402	0.114592	0.248524	0.0	0.030197	0.0
min	0.000000	0.000000	0.000000	0.000000	1.0	-0.130173	1.0
25%	0.300000	0.983607	0.000000	0.200000	1.0	-0.068726	1.0
50%	0.400000	0.983607	0.000000	0.400000	1.0	-0.041709	1.0
75%	0.600000	0.983607	0.000000	0.600000	1.0	-0.020598	1.0
max	1.200000	0.983607	0.500000	1.200000	1.0	-0.007601	1.0

In [33]:
```
fig, axes = plt.subplots(figsize=(22,7))
plt.plot(df_test['health'])
xmin, xmax = plt.xlim()
ymin, ymax = plt.ylim()
plt.xlim(xmin * 0, xmax * 1)
plt.ylim(ymin * 1, ymax * 1.3)
plt.axhspan(-0.18, 0.0, alpha=0.2, color ='red')
```

Out[33]: <matplotlib.patches.Polygon at 0x2dfe7b67ac0>



Παράρτημα Β

Σε αυτό το παράρτημα θα δούμε τον κώδικα για τον άλλο αλγόριθμο τον SOM. Θα παραθέσω ακριβώς το ίδιο σενάριο με το παράρτημα Α για τους ίδιους λόγους. Άρα σε αυτό το παράρτημα θα δούμε το blackHole_BN για local detection σε middle και top τοπολογίες.

Local Detection middle Topology

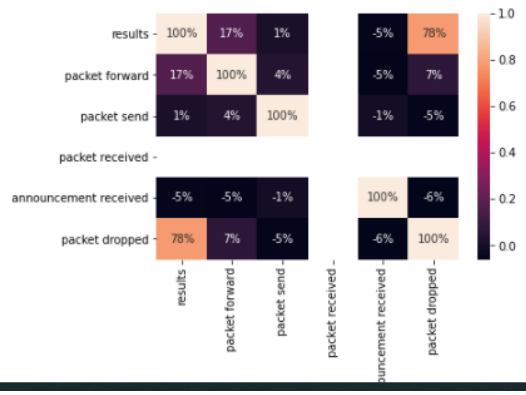
BlackHole_BN_evaluation.py

In [1]:
```
#import the liabries we need
from minisom import MiniSom
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt #for charts
import seaborn as sns
import time
import pickle
import shutil
```

In [2]:
```
#Take the 80% of benign and malicious txt and concatenate them. Then take 20% of each again and concatenate
from sklearn.model_selection import train_test_split
benign = np.loadtxt("data/benign_middle.txt", dtype='int')
malicious = np.loadtxt("data/BlackHole/SF_BH_BN_middle_malicious.txt", dtype='int')

X_benign = benign[:, 1:]
Y_benign = benign[:, 0]
X_malicious = malicious[:, 1:]
Y_malicious = malicious[:, 0]
X_trainB, X_testB, Y_trainB, Y_testB = train_test_split(X_benign, Y_benign, test_size=0.2,random_state=0)
X_trainM, X_testM, Y_trainM, Y_testM = train_test_split(X_malicious, Y_malicious, test_size=0.2,random_state=0)
X_train = np.concatenate((X_trainB,X_trainM),axis=0)
X_test = np.concatenate((X_testB,X_testM), axis=0)
Y_train = np.concatenate((Y_trainB,Y_trainM), axis=0)
Y_test= np.concatenate((Y_testB,Y_testM), axis=0)
```

In [3]:
```
df = pd.DataFrame(np.concatenate((benign,malicious),axis=0), columns = ['results','packet forward','packet send',
                                                                         'packet received','announcement received',
                                                                        'packet dropped'])
sns.heatmap(df.corr(),annot = True, fmt= ".0%")
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x19872d72e80>



In [4]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [5]:
```
X_train
```

Out[5]:
```
array([[0.66666667, 0.98360656, 0.        , 0.        , 0.        ],
       [0.33333333, 0.98360656, 0.        , 0.        , 0.        ],
       [0.        , 0.98360656, 0.        , 0.        , 0.        ],
       ...,
       [0.88888889, 0.98360656, 0.        , 0.        , 0.16666667],
       [0.55555556, 0.98360656, 0.        , 0.        , 0.33333333],
       [0.55555556, 0.98360656, 0.        , 0.        , 0.        ]])
```

In [6]:
```
'''
Here's naive classification function that classifies a sample in data using the label assigned to the associated winning neuron.
A label c is associated to a neuron if the majority of samples mapped in that neuron have label c. The function will assign
the most common label in the dataset in case that a sample is mapped to a neuron for which no class is assigned.
'''
def classify(som,datas):
    """Classifies each sample in data in one of the classes definited
    using the method labels_map.
    Returns a list of the same length of data where the i-th element
    is the class assigned to data[i].
    """
    winmap = som.labels_map(X_train, Y_train)
    default_class = np.sum(list(winmap.values())).most_common()[0][0]
    result = []
    for d in datas:
        win_position = som.winner(d)
        if win_position in winmap:
            result.append(winmap[win_position].most_common()[0][0])
        else:
            result.append(default_class)
    return result
```

In [7]:
```
#set hyperparameters
som_grid_rows = 20
som_grid_columns = 20
iterations = 150000
sigma = 1
learning_rate = 0.5
```

In [8]:
```
#initialization
som = MiniSom(x=som_grid_rows,
              y=som_grid_columns,
              input_len=X_train.shape[1],
              sigma=sigma,
              learning_rate=learning_rate)
som.random_weights_init(X_train)
```

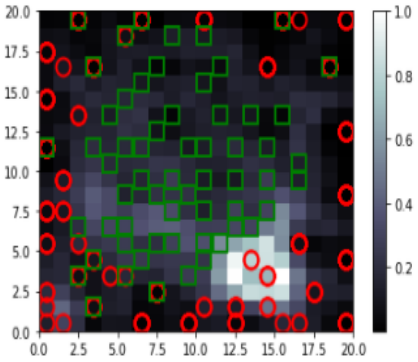
In [9]:
```
#training
start_time = time.time()
som.train(X_train,iterations,random_order=True)
elapsed_time = time.time() - start_time
print(elapsed_time, " seconds")
```

```
12.141347885131836  seconds
```

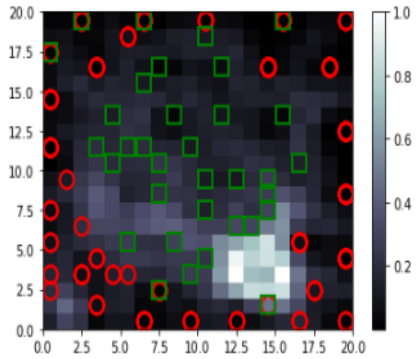
In [25]:
```
#save the trained model to be used for top topology
filename = 'blackHoleBN_SOM.sav'
pickle.dump(som, open(filename, 'wb'))

#move the file for top topology
original = "blackHoleBN_SOM.sav"
target =  r"C:\Users\steve\OneDrive\jupyter\Essay\SOM\FIT_DATA\localDetection\top"
shutil.move(original,target)
```

Out[25]:
```
'C:\\Users\\steve\\OneDrive\\jupyter\\Essay\\SOM\\FIT_DATA\\localDetection\\top\\blackHoleBN_SOM.sav'
```

In [10]:
```
#benign with blackholes attacks
from pylab import plot,axis,show,pcolor,colorbar,bone
winning_list = []
bone()
pcolor(som.distance_map().T) #distance map as background
colorbar()

#use different colors and markers for each label
markers = ['o','s','D']
colors = ['r','g','b']
for cnt,xx in enumerate(X_train):
    w = som.winner(xx) #getting the winner
    winning_list.append(w)
    #palce a marker on the winning position for the sample xx
    plot(w[0]+.5,w[1]+.5,markers[Y_train[cnt]],markerfacecolor='None',
        markeredgecolor=colors[Y_train[cnt]],markersize=12,markeredgewidth=2)
axis([0,som._weights.shape[0],0,som._weights.shape[1]])
show()
```



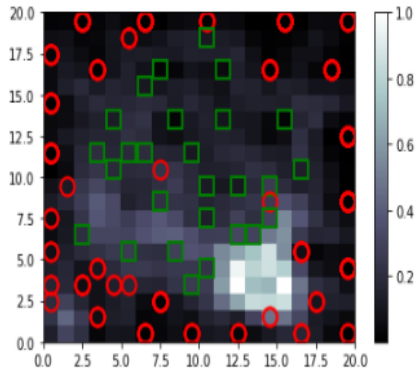
```
from pylab import plot,axis,show,pcolor,colorbar,bone
winning_list = []
bone()
pcolor(som.distance_map().T) #distance map as background
colorbar()

#use different colors and markers for each label
markers = ['o','s','D']
colors = ['r','g','b']
for cnt,xx in enumerate(X_test):
    w = som.winner(xx) #getting the winner
    winning_list.append(w)
    #palce a marker on the winning position for the sample xx
    plot(w[0]+.5,w[1]+.5,markers[Y_test[cnt]],markerfacecolor='None',
        markeredgecolor=colors[Y_test[cnt]],markersize=12,markeredgewidth=2)
axis([0,som._weights.shape[0],0,som._weights.shape[1]])
show()
```



In [12]:
```
from sklearn.metrics import classification_report,accuracy_score, confusion_matrix
classification = classify(som,X_test)
print(classification_report(Y_test, classification))
print(accuracy_score(Y_test, classification))
print(confusion_matrix(Y_test, classification))

#predicted 49 malicious and 636 benign
#actual 63 malicious and 622 benign
```

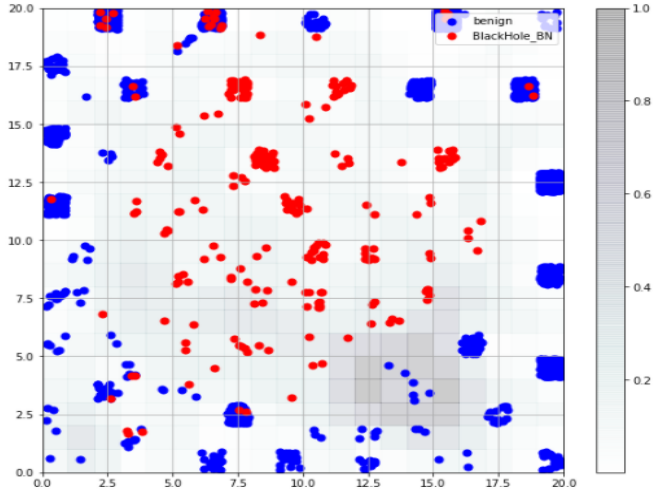
```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       622
           1       0.98      0.86      0.92        63

    accuracy                           0.99       685
   macro avg       0.98      0.93      0.95       685
weighted avg       0.99      0.99      0.98       685

0.9854014598540146
[[621   1]
 [  9  54]]
```

In [13]:
```
#benign with blackholes attacks
from pylab import plot,axis,show,pcolor,colorbar,bone
winning_list = []
bone()
pcolor(som.distance_map().T) #distance map as background
colorbar()

#use different colors and markers for each label
markers = ['o','s','D']
colors = ['r','g','b']
for cnt,xx in enumerate(X_test):
    w = som.winner(xx) #getting the winner
    winning_list.append(w)
    #palce a marker on the winning position for the sample xx
    plot(w[0]+.5,w[1]+.5,markers[classification[cnt]],markerfacecolor='None',
        markeredgecolor=colors[classification[cnt]],markersize=12,markeredgewidth=2)
axis([0,som._weights.shape[0],0,som._weights.shape[1]])
show()
```

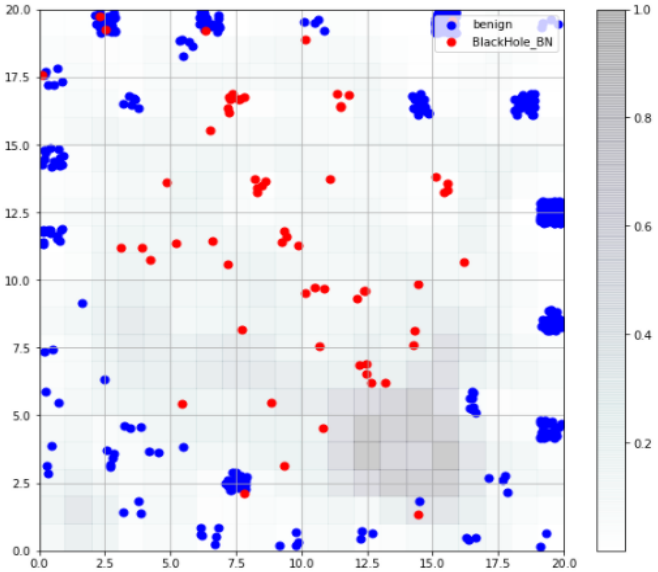


In [14]:
```
w_x, w_y = zip(*[som.winner(d) for d in X_train])
w_x = np.array(w_x)
w_y = np.array(w_y)
label_names = ['benign','BlackHole_BN']
plt.figure(figsize=(10, 9))
plt.pcolor(som.distance_map().T, cmap='bone_r', alpha=.2)
plt.colorbar()

for c in np.unique(Y_train):
    idx_target = Y_train==c
    plt.scatter(w_x[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                w_y[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                s=50, c=colors[c-1], label=label_names[c])
plt.legend(loc='upper right')
plt.grid()
plt.show()
```

In [14]:
```
w_x, w_y = zip(*[som.winner(d) for d in X_train])
w_x = np.array(w_x)
w_y = np.array(w_y)
label_names = ['benign','BlackHole_BN']
plt.figure(figsize=(10, 9))
plt.pcolor(som.distance_map().T, cmap='bone_r', alpha=.2)
plt.colorbar()

for c in np.unique(Y_train):
    idx_target = Y_train==c
    plt.scatter(w_x[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                w_y[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                s=50, c=colors[c-1], label=label_names[c])
plt.legend(loc='upper right')
plt.grid()
plt.show()
```



In [15]:
```
w_x, w_y = zip(*[som.winner(d) for d in X_test])
w_x = np.array(w_x)
w_y = np.array(w_y)
label_names = ['benign','BlackHole_BN']
plt.figure(figsize=(10, 9))
plt.pcolor(som.distance_map().T, cmap='bone_r', alpha=.2)
plt.colorbar()

for c in np.unique(Y_test):
    idx_target = Y_test==c
    plt.scatter(w_x[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                w_y[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                s=50, c=colors[c-1], label=label_names[c])
plt.legend(loc='upper right')
plt.grid()
plt.show()
```



In [16]:
```
#create a dataframe for test datas
Y_test = Y_test.reshape(-1,1)
df_test = pd.DataFrame(np.concatenate((X_test,Y_test),axis=1), columns = ['packet forward','packet send', 'packet received',
                                                                          'announcement received',
                                                                               'packet dropped', 'results'])
df_test.describe()
```

Out[16]:

	packet forward	packet send	packet received	announcement received	packet dropped	results
count	685.000000	685.000000	685.0	685.000000	685.000000	685.000000
mean	0.272019	0.981237	0.0	0.046715	0.029197	0.091971
std	0.277141	0.005836	0.0	0.117105	0.119728	0.289196
min	0.000000	0.967213	0.0	0.000000	0.000000	0.000000
25%	0.000000	0.983607	0.0	0.000000	0.000000	0.000000
50%	0.333333	0.983607	0.0	0.000000	0.000000	0.000000
75%	0.333333	0.983607	0.0	0.000000	0.000000	0.000000
max	1.000000	1.000000	0.0	0.750000	1.000000	1.000000

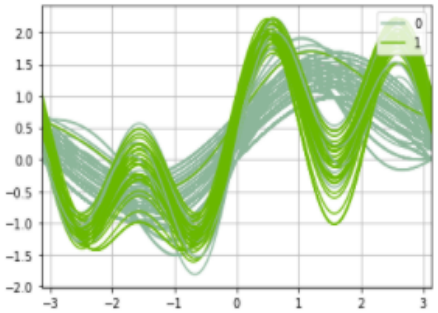
In [17]:
```
df_test['anomaly'] = classification
df_test
```

Out[17]:

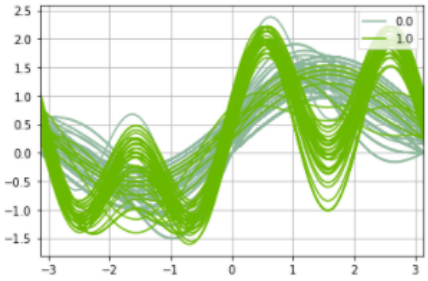
	packet forward	packet send	packet received	announcement received	packet dropped	results	anomaly
0	0.000000	0.983607	0.0	0.00	0.000000	0.0	0
1	1.000000	0.983607	0.0	0.00	0.000000	0.0	0
2	0.666667	0.983607	0.0	0.00	0.000000	0.0	0
3	0.666667	0.983607	0.0	0.00	0.000000	0.0	0
4	0.333333	0.983607	0.0	0.00	0.000000	0.0	0
...	...	...	...	...	...	...	...
680	0.333333	0.983607	0.0	0.25	0.000000	1.0	0
681	0.555556	0.983607	0.0	0.00	0.166667	1.0	1
682	0.555556	0.983607	0.0	0.00	0.166667	1.0	1
683	0.555556	0.983607	0.0	0.00	0.166667	1.0	1
684	0.888889	0.983607	0.0	0.00	0.166667	1.0	1

685 rows × 7 columns

In [18]:
```
import pandas.plotting as pdplt
pdplt.andrews_curves(df_test, 'anomaly')
plt.show()
```



In [20]:
```
import pandas.plotting as pdplt
pdplt.andrews_curves(df_test, 'results')
plt.show()
```



In [21]:
```
#Check the false positives
false_negative = df_test[(df_test['anomaly'] == 1) & (df_test['results'] == 0)]
false_negative
#As we can see we can set a threshold on packets forward and packets dropped to identify better the predictions
```

Out[21]:

	packet forward	packet send	packet received	announcement received	packet dropped	results	anomaly
618	0.888889	0.967213	0.0	0.25	0.0	0.0	1

In [22]:
```
#Check the false negatives
false_positives = df_test[(df_test['anomaly'] == 0) & (df_test['results'] == 1)]
false_positives
#As we can see we can set a threshold on packets forward and packets dropped to identify better the predictions
```

Out[22]:

	packet forward	packet send	packet received	announcement received	packet dropped	results	anomaly
633	0.111111	0.983607	0.0	0.00	0.500000	1.0	0
642	0.666667	0.983607	0.0	0.00	0.000000	1.0	0
644	0.777778	0.983607	0.0	0.00	0.000000	1.0	0
652	0.333333	0.983607	0.0	0.00	0.000000	1.0	0
655	0.666667	0.983607	0.0	0.00	0.000000	1.0	0
657	0.555556	0.983607	0.0	0.00	0.000000	1.0	0
661	0.444444	0.983607	0.0	0.00	0.500000	1.0	0
671	0.222222	0.983607	0.0	0.50	0.166667	1.0	0
680	0.333333	0.983607	0.0	0.25	0.000000	1.0	0

In [23]:
```
#Check the true negatives
true_negative = df_test[(df_test['results'] == 1)]
true_negative
#As we can see we can set a threshold on packets forward and packets dropped to identify better the predictions
```

Out[23]:

	packet forward	packet send	packet received	announcement received	packet dropped	results	anomaly
622	0.555556	0.967213	0.0	0.00	0.666667	1.0	1
623	0.333333	0.983607	0.0	0.00	0.500000	1.0	1
624	0.111111	0.983607	0.0	0.00	0.333333	1.0	1
625	0.444444	1.000000	0.0	0.00	0.166667	1.0	1
626	0.444444	0.983607	0.0	0.00	0.166667	1.0	1
...	...	...	...	...	...	...	...
680	0.333333	0.983607	0.0	0.25	0.000000	1.0	0
681	0.555556	0.983607	0.0	0.00	0.166667	1.0	1
682	0.555556	0.983607	0.0	0.00	0.166667	1.0	1
683	0.555556	0.983607	0.0	0.00	0.166667	1.0	1
684	0.888889	0.983607	0.0	0.00	0.166667	1.0	1

63 rows × 7 columns

In [ ]:
```
```

Local Detection top Topology

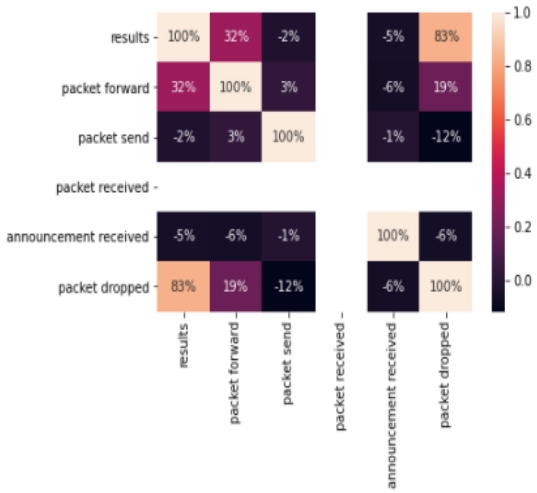
BlackHole_BN_evaluation.py

In [1]:
```
#import the liabries we need
from minisom import MiniSom
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt #for charts
import seaborn as sns
import time
import pickle
```

In [2]:
```
#Take the 80% of benign and malicious txt and concatenate them. Then take 20% of each again and concatenate
from sklearn.model_selection import train_test_split
benign = np.loadtxt("data/benign_top.txt", dtype='int')
malicious = np.loadtxt("data/BlackHole/SF_BH_BN_top_malicious.txt", dtype='int')

X_benign = benign[:, 1:]
Y_benign = benign[:, 0]
X_malicious = malicious[:, 1:]
Y_malicious = malicious[:, 0]
X_trainB, X_testB, Y_trainB, Y_testB = train_test_split(X_benign, Y_benign, test_size=0.2,random_state=0)
X_trainM, X_testM, Y_trainM, Y_testM = train_test_split(X_malicious, Y_malicious, test_size=0.2,random_state=0)
X_train = np.concatenate((X_trainB,X_trainM),axis=0)
X_test = np.concatenate((X_testB,X_testM), axis=0)
Y_train = np.concatenate((Y_trainB,Y_trainM), axis=0)
Y_test= np.concatenate((Y_testB,Y_testM), axis=0)
```

In [3]:
```
df = pd.DataFrame(np.concatenate((benign,malicious),axis=0), columns = ['results','packet forward','packet send',
                                                                         'packet received','announcement received',
                                                                        'packet dropped'])

sns.heatmap(df.corr(),annot = True, fmt= ".0%")
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x242d808fdf0>



In [4]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [5]:
```
X_train
```

Out[5]:
```
array([[0.3       , 0.96721311, 0.        , 0.        , 0.        ],
       [0.        , 0.96721311, 0.        , 0.        , 0.        ],
       [0.3       , 0.98360656, 0.        , 0.        , 0.        ],
       ...,
       [0.2       , 0.96721311, 0.        , 0.        , 0.4       ],
       [0.6       , 0.98360656, 0.        , 0.        , 0.2       ],
       [0.8       , 0.98360656, 0.        , 0.5       , 0.4       ]])
```

In [6]:
```
#Load the middle trained model
loaded_model = pickle.load(open("blackHoleBN_SOM.sav", 'rb'))
```

In [7]:
```
'''
Here's naive classification function that classifies a sample in data using the label assigned to the associated winning neuron.
A label c is associated to a neuron if the majority of samples mapped in that neuron have label c. The function will assign
the most common label in the dataset in case that a sample is mapped to a neuron for which no class is assigned.
'''
def classify(som,datas):
    """Classifies each sample in data in one of the classes definited
    using the method labels_map.
    Returns a list of the same length of data where the i-th element
    is the class assigned to data[i].
    """
    winmap = som.labels_map(X_train, Y_train)
    default_class = np.sum(list(winmap.values())).most_common()[0][0]
    result = []
    for d in datas:
        win_position = som.winner(d)
        if win_position in winmap:
            result.append(winmap[win_position].most_common()[0][0])
        else:
            result.append(default_class)
    return result
```

In [8]:
```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
classification = classify(loaded_model,X_test)
print(classification_report(Y_test, classification))
print(accuracy_score(Y_test, classification))
print(confusion_matrix(Y_test, classification))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       622
           1       1.00      0.89      0.94        62

    accuracy                           0.99       684
   macro avg       0.99      0.94      0.97       684
weighted avg       0.99      0.99      0.99       684

0.9897660818713451
[[622   0]
 [  7  55]]
```

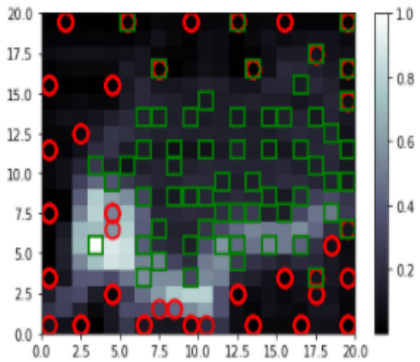
In [9]:
```
#set hyperparameters
som_grid_rows = 20
som_grid_columns = 20
iterations = 150000
sigma = 1
learning_rate = 0.5
```

In [10]:
```
#initialization
som = MiniSom(x=som_grid_rows,
              y=som_grid_columns,
              input_len=X_train.shape[1],
              sigma=sigma,
              learning_rate=learning_rate)
som.random_weights_init(X_train)
```

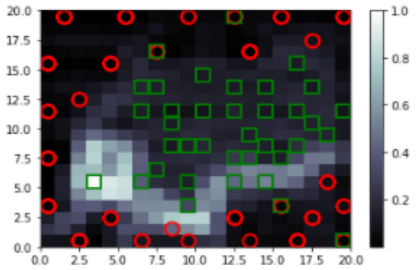
In [11]:
```
#training
start_time = time.time()
som.train(X_train,iterations,random_order=True)
elapsed_time = time.time() - start_time
print(elapsed_time, " seconds")
```

```
12.163004875183105  seconds
```

In [12]:
```
#benign with blackholes attacks
from pylab import plot,axis,show,pcolor,colorbar,bone
winning_list = []
bone()
pcolor(som.distance_map().T) #distance map as background
colorbar()

#use different colors and markers for each label
markers = ['o','s','D']
colors = ['r','g','b']
for cnt,xx in enumerate(X_train):
    w = som.winner(xx) #getting the winner
    winning_list.append(w)
    #palce a marker on the winning position for the sample xx
    plot(w[0]+.5,w[1]+.5,markers[Y_train[cnt]],markerfacecolor='None',
        markeredgecolor=colors[Y_train[cnt]],markersize=12,markeredgewidth=2)
axis([0,som._weights.shape[0],0,som._weights.shape[1]])
show()
```



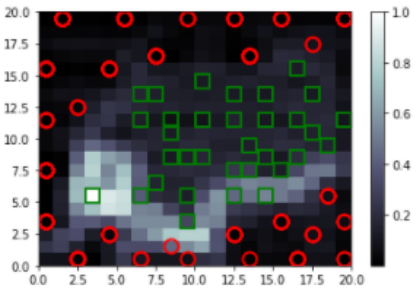
In [13]:
```
#benign with blackholes attacks
from pylab import plot,axis,show,pcolor,colorbar,bone
winning_list = []
bone()
pcolor(som.distance_map().T) #distance map as background
colorbar()

#use different colors and markers for each label
markers = ['o','s','D']
colors = ['r','g','b']
for cnt,xx in enumerate(X_test):
    w = som.winner(xx) #getting the winner
    winning_list.append(w)
    #palce a marker on the winning position for the sample xx
    plot(w[0]+.5,w[1]+.5,markers[Y_test[cnt]],markerfacecolor='None',
        markeredgecolor=colors[Y_test[cnt]],markersize=12,markeredgewidth=2)
axis([0,som._weights.shape[0],0,som._weights.shape[1]])
show()
```



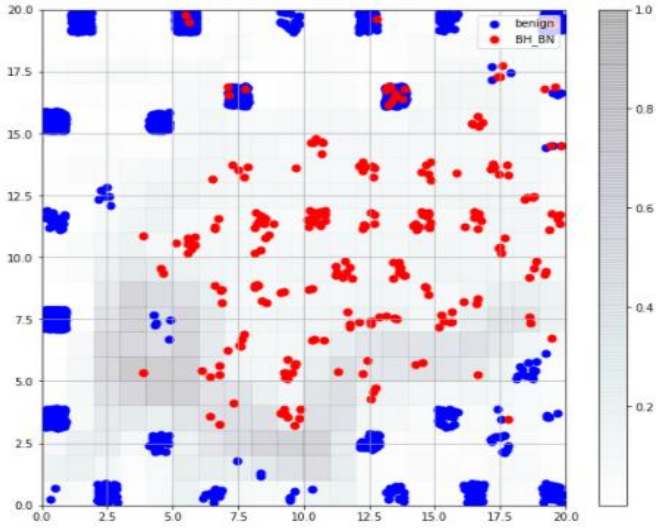
In [14]:
```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
classification = classify(som,X_test)
print(classification_report(Y_test, classification))
print(accuracy_score(Y_test, classification))
print(confusion_matrix(Y_test, classification))

#predicted 53 malicious and 631 benign
#actual malicious 65 and 620 benign
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       622
           1       1.00      0.92      0.96        62

    accuracy                           0.99       684
   macro avg       1.00      0.96      0.98       684
weighted avg       0.99      0.99      0.99       684

0.9926900584795322
[[622   0]
 [  5  57]]
```

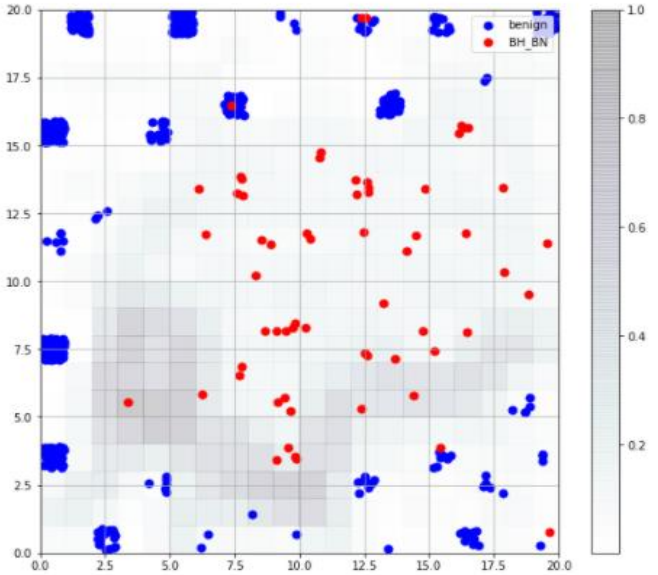
In [15]:
```
#benign with blackholes attacks
from pylab import plot,axis,show,pcolor,colorbar,bone
winning_list = []
bone()
pcolor(som.distance_map().T) #distance map as background
colorbar()

#use different colors and markers for each label
markers = ['o','s','D']
colors = ['r','g','b']
for cnt,xx in enumerate(X_test):
    w = som.winner(xx) #getting the winner
    winning_list.append(w)
    #palce a marker on the winning position for the sample xx
    plot(w[0]+.5,w[1]+.5,markers[classification[cnt]],markerfacecolor='None',
        markeredgecolor=colors[classification[cnt]],markersize=12,markeredgewidth=2)
axis([0,som._weights.shape[0],0,som._weights.shape[1]])
show()
```



In [16]:
```
w_x, w_y = zip(*[som.winner(d) for d in X_train])
w_x = np.array(w_x)
w_y = np.array(w_y)
label_names = ['benign','BH_BN']
plt.figure(figsize=(10, 9))
plt.pcolor(som.distance_map().T, cmap='bone_r', alpha=.2)
plt.colorbar()

for c in np.unique(Y_train):
    idx_target = Y_train==c
    plt.scatter(w_x[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                w_y[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                s=50, c=colors[c-1], label=label_names[c])
plt.legend(loc='upper right')
plt.grid()
plt.show()
```



In [17]:
```
w_x, w_y = zip(*[som.winner(d) for d in X_test])
w_x = np.array(w_x)
w_y = np.array(w_y)
label_names = ['benign','BH_BN']
plt.figure(figsize=(10, 9))
plt.pcolor(som.distance_map().T, cmap='bone_r', alpha=.2)
plt.colorbar()

for c in np.unique(Y_test):
    idx_target = Y_test==c
    plt.scatter(w_x[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                w_y[idx_target]+.5+(np.random.rand(np.sum(idx_target))-.5)*.8,
                s=50, c=colors[c-1], label=label_names[c])
plt.legend(loc='upper right')
plt.grid()
plt.show()
```



In [18]:
```
#create a dataframe for test datas
Y_test = Y_test.reshape(-1,1)
df_test = pd.DataFrame(np.concatenate((X_test,Y_test),axis=1), columns = ['packet forward','packet send', 'packet received',
                                                                          'announcement received',
                                                                               'packet dropped', 'results'])
df_test.describe()
```

Out[18]:

	packet forward	packet send	packet received	announcement received	packet dropped	results
count	684.000000	684.000000	684.0	684.000000	684.000000	684.000000
mean	0.243713	0.978454	0.0	0.059942	0.036842	0.090643
std	0.208329	0.038091	0.0	0.175524	0.140859	0.287311
min	0.000000	0.000000	0.0	0.000000	0.000000	0.000000
25%	0.000000	0.983607	0.0	0.000000	0.000000	0.000000
50%	0.300000	0.983607	0.0	0.000000	0.000000	0.000000
75%	0.300000	0.983607	0.0	0.000000	0.000000	0.000000
max	1.200000	0.983607	0.0	1.500000	1.200000	1.000000

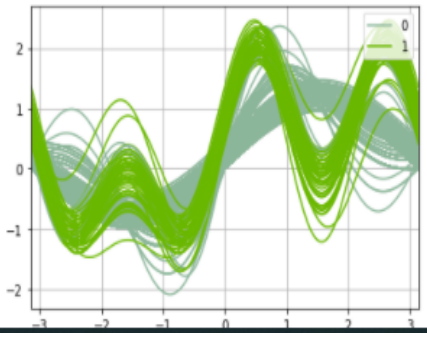
In [19]:
```
df_test['anomaly'] = classification
df_test
```

Out[19]:

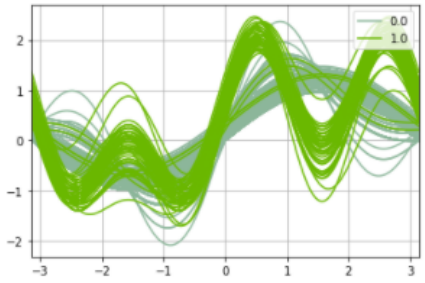
	packet forward	packet send	packet received	announcement received	packet dropped	results	anomaly
0	0.3	0.967213	0.0	0.0	0.0	0.0	0
1	0.5	0.983607	0.0	0.0	0.0	0.0	0
2	0.0	0.983607	0.0	0.0	0.0	0.0	0
3	0.3	0.983607	0.0	0.0	0.0	0.0	0
4	0.3	0.967213	0.0	0.0	0.0	0.0	0
...	...	...	...	...	...	...	...
679	0.4	0.983607	0.0	0.0	0.2	1.0	1
680	0.2	0.983607	0.0	0.0	0.8	1.0	1
681	0.6	0.983607	0.0	0.0	0.2	1.0	1
682	0.0	0.967213	0.0	0.0	0.6	1.0	1
683	0.7	0.983607	0.0	0.0	0.2	1.0	1

684 rows × 7 columns

In [20]:
```
import pandas.plotting as pdplt
pdplt.andrews_curves(df_test, 'anomaly')
plt.show()
```



In [21]:
```
import pandas.plotting as pdplt
pdplt.andrews_curves(df_test, 'results')
plt.show()
```



In [21]:
```
#Check the false positives
false_negative = df_test[(df_test['anomaly'] == 1) & (df_test['results'] == 0)]
false_negative
#As we can see we can set a threshold on packets forward and packets dropped to identify better the predictions
```

Out[21]:

	packet forward	packet send	packet received	announcement received	packet dropped	results	anomaly
618	0.888889	0.967213	0.0	0.25	0.0	0.0	1

In [22]:
```
#Check the false negatives
false_positives = df_test[(df_test['anomaly'] == 0) & (df_test['results'] == 1)]
false_positives
#As we can see we can set a threshold on packets forward and packets dropped to identify better the predictions
```

Out[22]:

	packet forward	packet send	packet received	announcement received	packet dropped	results	anomaly
633	0.111111	0.983607	0.0	0.00	0.500000	1.0	0
642	0.666667	0.983607	0.0	0.00	0.000000	1.0	0
644	0.777778	0.983607	0.0	0.00	0.000000	1.0	0
652	0.333333	0.983607	0.0	0.00	0.000000	1.0	0
655	0.666667	0.983607	0.0	0.00	0.000000	1.0	0
657	0.555556	0.983607	0.0	0.00	0.000000	1.0	0
661	0.444444	0.983607	0.0	0.00	0.500000	1.0	0
671	0.222222	0.983607	0.0	0.50	0.166667	1.0	0
680	0.333333	0.983607	0.0	0.25	0.000000	1.0	0

In [23]:
```
#Check the true negatives
true_negative = df_test[(df_test['results'] == 1)]
true_negative
#As we can see we can set a threshold on packets forward and packets dropped to identify better the predictions
```

Out[23]:

	packet forward	packet send	packet received	announcement received	packet dropped	results	anomaly
622	0.555556	0.967213	0.0	0.00	0.666667	1.0	1
623	0.333333	0.983607	0.0	0.00	0.500000	1.0	1
624	0.111111	0.983607	0.0	0.00	0.333333	1.0	1
625	0.444444	1.000000	0.0	0.00	0.166667	1.0	1
626	0.444444	0.983607	0.0	0.00	0.166667	1.0	1
...	...	...	...	...	...	...	...
680	0.333333	0.983607	0.0	0.25	0.000000	1.0	0
681	0.555556	0.983607	0.0	0.00	0.166667	1.0	1
682	0.555556	0.983607	0.0	0.00	0.166667	1.0	1
683	0.555556	0.983607	0.0	0.00	0.166667	1.0	1
684	0.888889	0.983607	0.0	0.00	0.166667	1.0	1

63 rows × 7 columns