



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

ΑΝΙΧΝΕΥΣΗ ΕΠΙΘΕΣΕΩΝ ΣΕ ΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ ΜΕ ΑΛΓΟΡΙΘΜΟΥΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Πλακούρα Ευαγγελία, 02103

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ:

Επιβλέπων

Πλαγιανάκος Βασίλειος

ΛΑΜΙΑ, 2023-2024

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον αξιότιμο κύριο Πλαγιανάκο Βασίλειο, Κοσμήτορα της Σχολής Θετικών Επιστημών του Πανεπιστημίου Θεσσαλίας για την καθοδήγησή του στην εκπόνηση της εργασίας και την παρότρυνσή του για πειραματισμό στο επιστημονικό μου ενδιαφέρον.

Επίσης, θα ήθελα να απευθύνω τις ευχαριστίες μου στον κύριο Αναγνώστου Παναγιώτη, διδάκτορα της σχολής Θετικών Επιστημών, ο οποίος αποτέλεσε σημαντική βοήθεια στην όλη προσπάθεια.

Τέλος, ευχαριστώ την οικογένεια και τους ανθρώπους που με στήριξαν για την επίτευξη του στόχου μου.

Η προσπάθειά μου αφιερώνεται στη γιαγιά μου, Ελένη.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ.....	5
ABSTRACT.....	6

ΓΕΝΙΚΟ ΜΕΡΟΣ

1 ΑΡΧΕΣ ΑΣΦΑΛΕΙΑΣ ΔΙΚΤΥΩΝ.....	7
2.1 ΦΟΡΕΙΣ ΕΠΙΘΕΣΗΣ.....	8
2.2 ΕΠΙΦΑΝΕΙΕΣ ΕΠΙΘΕΣΗΣ.....	8
3 ΚΑΤΗΓΟΡΙΕΣ ΕΠΙΘΕΣΕΩΝ ΣΕ ΔΙΚΤΥΑ.....	9
4.1 ΕΥΠΑΘΕΙΕΣ TCP/IP.....	11
4.2 ΕΥΠΑΘΕΙΕΣ ΠΡΩΤΟΚΟΛΛΟΥ TCP.....	12
4.3 ΕΥΠΑΘΕΙΕΣ ΠΡΩΤΟΚΟΛΛΟΥ IP.....	13
4.4 ΕΥΠΑΘΕΙΕΣ ΠΡΩΤΟΚΟΛΛΟΥ ICMP.....	13
4.5 ΕΥΠΑΘΕΙΕΣ ΠΡΩΤΟΚΟΛΛΟΥ UDP.....	14
5.1 ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ.....	15
5.2 ΕΠΙΒΛΕΠΟΜΕΝΗ ΚΑΙ ΜΗ ΕΠΙΒΛΕΠΟΜΕΝΗ ΜΑΘΗΣΗ.....	15
6 ΑΛΓΟΡΙΘΜΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΣΤΗΝ ΑΝΙΧΝΕΥΣΗ ΕΠΙΘΕΣΕΩΝ.....	17

ΕΙΔΙΚΟ ΜΕΡΟΣ

1 ΕΙΣΑΓΩΓΗ.....	19
2 ΣΚΟΠΟΣ.....	20
2.1 ΣΥΛΛΟΓΗ ΔΕΔΟΜΕΝΩΝ.....	20
3.2 ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ.....	23
4 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ ΑΝΙΧΝΕΥΣΗΣ.....	31
5.1 ΕΚΠΑΙΔΕΥΣΗ ΑΛΓΟΡΙΘΜΩΝ.....	33
5.2 ΠΕΡΙΓΡΑΦΗ ΠΕΙΡΑΜΑΤΩΝ.....	34
5.3 ΠΑΡΑΜΕΤΡΟΙ ΕΚΠΑΙΔΕΥΣΗΣ.....	35
6 ΑΠΟΤΕΛΕΣΜΑΤΑ	41
6.1 ΠΑΡΑΤΗΡΗΣΗΣ ΠΡΟΕΠΕΞΕΡΓΑΣΙΑΣ ΔΕΔΟΜΕΝΩΝ.....	41
6.2 ΣΥΓΚΡΙΤΙΚΗ ΑΝΑΛΥΣΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΜΕΤΑΞΥ ΑΛΓΟΡΙΘΜΩΝ.....	46
6.3 ΣΥΜΠΕΡΑΣΜΑΤΑ ΣΥΓΚΡΙΣΗΣ ΑΝΑ ΖΕΥΓΗ ΚΑΙ ΑΝΑ ΤΡΙΑΔΑ.....	46
6.4 ΕΡΜΗΝΕΥΣΙΜΟΤΗΤΑ.....	48
6.5 ΑΠΟΔΟΣΗ ΜΟΝΤΕΛΟΥ ΤΩΝ ΔΥΟ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ.....	48
6.6 ΣΥΓΚΡΙΣΗ ΤΩΝ ΔΥΟ ΜΟΝΤΕΛΩΝ ΕΚΠΑΙΔΕΥΣΗΣ.....	49
7 ΣΥΜΠΕΡΑΣΜΑ.....	51
8 ΣΥΖΗΤΗΣΗ.....	51
9 ΚΩΔΙΚΕΣ ΥΛΟΠΟΙΗΣΗΣ.....	53
9.1 Κώδικες Data Analysis.....	53
9.2 Κώδικες Εξαγωγής Κορυφαίων Χαρακτηριστικών.....	61
9.3 Κώδικας μοντέλου πρόβλεψης δύο χαρακτηριστικών.....	74
9.4 Κώδικας μέτρησης χρόνου απόκρισης του μοντέλου χρήσης όλων των χαρακτηριστικών και του μοντέλου χρήσης των δύο κορυφαίων χαρακτηριστικών.....	77
9.5 Κώδικες Παραμέτρων.....	83
10 ΒΙΒΛΙΟΓΡΑΦΙΑ-ΑΝΑΦΟΡΕΣ.....	102

ΠΕΡΙΛΗΨΗ

Εισαγωγή: Η ανίχνευση επιθέσεων σε δίκτυα υπολογιστών μέσω αλγορίθμων μηχανικής μάθησης αποτελεί μια σύγχρονη προσέγγιση ανάλυσης μεγάλων συνόλων δεδομένων για τον εντοπισμό ύποπτων δραστηριοτήτων και αποκλίσεων από τις πρότυπες ροές δεδομένων.

Σκοπός: Πραγματοποιείται εκπαίδευση τεσσάρων αλγορίθμων μηχανικής μάθησης με στόχο την εξαγωγή των κορυφαίων χαρακτηριστικών των ροών δεδομένων που είναι κοινά στους περισσότερους από αυτούς τους αλγορίθμους. Στη συνέχεια της εργασίας, προτείνεται ένα νέο μοντέλο που χρησιμοποιεί τους ίδιους αλγορίθμους, ωστόσο η εκπαίδευσή του περιορίζεται στα κορυφαία κοινά χαρακτηριστικά που είχαν εξαχθεί στο πρώτο μέρος. Ο στόχος αυτής της προσέγγισης είναι η επίτευξη μιας καλής απόδοσης και η παράλληλη μείωση του αριθμού των χαρακτηριστικών που απαιτούνται για την εκπαίδευση.

Μεθοδολογία: Για τη διεξαγωγή της εργασίας, χρησιμοποιήθηκε ένα dataset 41 χαρακτηριστικών που περιγράφουν ροές του δικτύου οι οποίες χαρακτηρίζονται είτε ως κανονικές, είτε ως ανωμαλίες. Σύμφωνα με τα 27 από αυτά χαρακτηριστικά, εκπαιδεύτηκαν οι τέσσερις αλγόριθμοι μηχανικής μάθησης, οι Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier και Extra Trees Classifier. Έγινε χρήση της μεθόδου ‘feature_importances’ της βιβλιοθήκης scikit-learn της Python ώστε να υποδειχτεί η σημασία κάθε χαρακτηριστικού και της μεθόδου ‘.intersection()’ ώστε να αναδειχθούν τα κορυφαία κοινά χαρακτηριστικά των αλγορίθμων. Στο δεύτερο κομμάτι της εργασίας, εκπαιδεύτηκαν οι ίδιοι αλγόριθμοι, αυτή τη φορά μόνο με τα τρία χαρακτηριστικά που υποδείχθηκαν από τα αποτελέσματα του πρώτου μοντέλου και υπολογίστηκε το ποσοστό ακρίβειας των προβλέψεων το f1 score, καθώς και η χρονική διαφορά μεταξύ των δύο μοντέλων.

Αποτελέσματα: Οι συγκρίσεις των χαρακτηριστικών ανέδειξαν τα χαρακτηριστικά ‘src_bytes’, ‘dst_bytes’ και ‘dst_host_srv_count’ ως τα πιο κοινά εμφανιζόμενα. Η εκπαίδευση των αλγορίθμων με αυτά χαρακτηριστικά έδωσε ποσοστό ακρίβειας και f1 score από 96 - 98%, ανάλογα τον αλγόριθμο, γεγονός που ενσωματώνει το μοντέλο αποτελεσματικά μέσα στα πλαίσια των αποδεκτά χρησιμοποιούμενων εφαρμογών ασφαλείας δικτύων. Επιπλέον, μετρήθηκε πράγματι διαφορά στο χρόνο (σε δευτερόλεπτα).

Συμπέρασμα: Η χρήση αλγορίθμων μηχανικής μάθησης, συμβάλει στην εξαγωγή σημαντικών πληροφοριών για τον ρόλο συγκεκριμένων παραμέτρων στις επιθέσεις δικτύων. Λαμβάνοντάς τες υπόψη, μπορούν να προταθούν διάφορα συστήματα ασφαλείας δικτύων με χαμηλότερη

πολυπλοκότητα, γρηγορότερη σε πραγματικό χρόνο απόκριση και πιο στοχευμένη αντιμετώπιση.

Λέξεις – Κλειδιά: αλγόριθμοι, δεδομένα, μηχανική μάθηση, κορυφαία χαρακτηριστικά, ασφάλεια, επιθέσεις, ροές δικτύων

ABSTRACT

Introduction: Detecting attacks on networks through machine learning algorithms is a modern approach to analyzing large data sets to identify suspicious activities and deviations from standard data flows.

Purpose: Four machine learning algorithms are trained with the aim of extracting the top features of data streams that are common across most of these algorithms. Subsequently, a new model is proposed, which utilizes the same algorithms but limits its training to the top features previously extracted. This approach aims to achieve good performance while significantly reducing the number of features required for training.

Methods: A dataset with 41 features describing network flows, which are either normal or anomalies, was used. The algorithms Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier, and Extra Trees Classifier were trained, and the 'feature_importances' method from the scikit-learn library in Python was used to indicate the importance of each feature, as well as the .intersection() method in order to highlight the top common features among the algorithms. The same algorithms were then trained using only the three features identified by the results of the first model, and the accuracy of the predictions was calculated, as well as the time difference between the two models.

Results: The feature comparisons highlighted 'src_bytes', 'dst_bytes' and 'dst_host_srv_count' as the most commonly encountered features. Training the algorithms with those features yielded an accuracy and F1 score rate from 96% to 98%, depending on the algorithm, which indicates that the implementation is effective within the acceptable standards for network security applications.. Additionally, there was indeed a difference in time measured in seconds.

Conclusion: Using machine learning algorithms, valuable insights can be gained into the role of specific parameters in network attacks. Considering their significance, network security

systems can be proposed with lower complexity, faster real-time response, and more targeted mitigation.

Key – Words: Algorithms, data, machine learning, top features, security, attacks, network flows

ΓΕΝΙΚΟ ΜΕΡΟΣ

1 ΑΡΧΕΣ ΑΣΦΑΛΕΙΑΣ ΔΙΚΤΥΩΝ

Ορίζουμε την ασφάλεια δικτύων ως {την ικανότητα μιας επιχείρησης ή ενός οργανισμού να προστατεύει τις πληροφορίες του από τυχόν αλλοιώσεις και καταστροφές, καθώς και από μη εξουσιοδοτημένη χρήση των πόρων του}[7]. Η έννοια αυτή περιλαμβάνει συμπληρωματικά από μεριάς δικτύου, την ικανότητα της επιτυχημένης αναγνώρισης και υψηλής αξιοπιστίας αντιμετώπισης των κακόβουλων ενεργειών και διατήρησης του απορρήτου του νόμιμου χρήστη του δικτύου. Οι βασικές λοιπόν αρχές που διέπουν την ασφάλεια των δικτύων υπολογιστών είναι τρεις: η διαθεσιμότητα, η εμπιστευτικότητα και η ακεραιότητα.

Η διαθεσιμότητα συνδέεται με την διασφάλιση της ελεύθερης ροής των πληροφοριών στους εξουσιοδοτημένους χρήστες. Για τη διατήρησή της χρησιμοποιούνται πρακτικές όπως προστασία από επιθέσεις τύπου DDoS, Disaster Recovery και εξασφάλιση εφεδρικών πόρων ώστε να υπάρχει συνεχής λειτουργία του συστήματος.

Η εμπιστευτικότητα σχετίζεται με την εξουσιοδότηση της πληροφορίας και πως αυτή παρέχεται στο σωστό κάθε φορά χρήστη. Για την επίτευξή της χρησιμοποιούνται τεχνικές όπως η κρυπτογράφηση, η διαχείριση πρόσβασης σε πληροφορίες και πόρους και η αυθεντικοποίηση της ταυτότητας των χρηστών πριν αποκτήσουν πρόσβαση στα δεδομένα.

Τέλος, η ακεραιότητα εξασφαλίζει την ακρίβεια και την αξιοπιστία των διαμοιραζόμενων πληροφοριών. Η προστασία της ακεραιότητας διασφαλίζεται υπό τις εξής μεθόδους: hash functions με τις οποίες δημιουργούνται μοναδικοί κωδικοί ελέγχου δεδομένων που ελέγχουν τυχόν αλλαγές ή παραβιάσεις, ψηφιακές υπογραφές για την επιβεβαίωση της γνησιότητας των μηνυμάτων και εγγράφων και τον έλεγχο σφαλμάτων με τεχνικές όπως checksums, parity bits ώστε να ανιχνευτούν και διορθωθούν τα σφάλματα στα δεδομένα.

2.1 ΦΟΡΕΙΣ ΕΠΙΘΕΣΗΣ

Ένας φορέας επίθεσης είναι η διαδρομή μέσω της οποίας ένας επιτιθέμενος εκτελεί μια κυβερνοεπίθεση. Οι κύριες απειλές για την ασφάλεια περιλαμβάνουν:

- Αναγνώριση: Συλλογή πληροφοριών για μελλοντικές επιθέσεις.
- Γνωστές Ευπάθειες: Εκμετάλλευση αδυναμιών σε υλικό ή λογισμικό.
- SQL Injection: Χειραγώγηση ερωτημάτων βάσεων δεδομένων μέσω μη ασφαλούς εισόδου.
- Phishing: Απάτη μέσω παραπλανητικών emails που κλέβουν διαπιστευτήρια.
- Προηγμένη Μόνιμη Απειλή (APT): Μακροχρόνια, κρυφή παρακολούθηση και υποκλοπή δεδομένων.
- Κακόβουλο Λογισμικό: Ιοί, σκουλήκια και άλλες κακόβουλες εφαρμογές.
- Αδύναμος Έλεγχος Ταυτότητας: Ασθενείς μηχανισμοί επαλήθευσης χρήστη.

2.2 ΕΠΙΦΑΝΕΙΕΣ ΕΠΙΘΕΣΗΣ

Η επιφάνεια επίθεσης αναφέρεται στο σύνολο όλων των τρωτών σημείων ή ευπαθειών σε μια υπολογιστική συσκευή ή δίκτυο που είναι διαθέσιμα και προσβάσιμα στους επιτιθέμενους. Περιλαμβάνει τις αδυναμίες που μπορούν να εκμεταλλευτούν οι επιτιθέμενοι για να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση ή να προκαλέσουν ζημιά στο σύστημα. Η μείωση της επιφάνειας επίθεσης επιτυγχάνεται με τη βελτίωση της ασφάλειας και την ελαχιστοποίηση των δυνατών σημείων εισόδου για κακόβουλες ενέργειες. Οι επιφάνειες επίθεσης ενός οργανισμού μπορούν να καταταγούν σε τέσσερις βασικές κατηγορίες:

- Επιφάνεια Επίθεσης Δικτύου: Περιλαμβάνει ευπάθειες που σχετίζονται με θύρες, πρωτόκολλα, και συσκευές δικτύου, όπως δρομολογητές και εκτυπωτές. Αυτές οι ευπάθειες μπορούν να εκμεταλλευτούν ακατάλληλα κρυπτογραφημένα πρωτόκολλα ή ανοιχτές θύρες.
- Επιφάνεια Επίθεσης Λογισμικού: Αναφέρεται στις αδυναμίες που ενδέχεται να προκύψουν από μη ενημερωμένο ή ευάλωτο λογισμικό, εφαρμογές και υπηρεσίες. Οι ευπάθειες στην περίπτωση αυτή, περιλαμβάνουν γνωστά τρωτά σημεία και αδυναμίες σε εκτελέσιμα αρχεία ή ιστοσελίδες.

- Φυσική Επιφάνεια Επίθεσης: Αφορά τα φυσικά σημεία εισόδου που είναι προσβάσιμα από επιτιθέμενους που βρίσκονται στην ίδια τοποθεσία με τον στόχο. Περιλαμβάνει απειλές από εσωτερικούς χρήστες, φυσικές διαρρήξεις και κλοπές υλικού.
- Επιφάνεια Επίθεσης Κοινωνικής Μηχανικής: Χρησιμοποιεί ψυχολογικές τεχνικές για να εξαπατήσει χρήστες ώστε να αποκαλύψουν ευαίσθητες πληροφορίες. Περιλαμβάνει επιθέσεις phishing, ψεύτικες κλήσεις και δούρειους ίππους που χρησιμοποιούν κοινωνικές παγίδες.

3 ΚΑΤΗΓΟΡΙΕΣ ΕΠΙΘΕΣΕΩΝ ΣΕ ΔΙΚΤΥΑ

Το κακόβουλο λογισμικό έχει κατηγοριοποιηθεί σύμφωνα με τα χαρακτηριστικά που διέπουν τον τύπο της επίθεσης. Οι παραλλαγές των επιθέσεων υποστηρίζονται από διαφορετικά εργαλεία που εξυπηρετούν τον σκοπό αυτό. Οι κύριες κατηγορίες επιθέσεων περιλαμβάνουν:

1. Άρνηση Υπηρεσίας (Denial of Service – DoS και Distributed Denial of Service – DDoS):
 - ο Οι επιθέσεις DoS αποσκοπούν στην εξάντληση των πόρων του συστήματος, καθιστώντας το ανίκανο να εξυπηρετήσει τους εξουσιοδοτημένους χρήστες.
 - ο Οι επιθέσεις DDoS έχοντας τον ίδιο σκοπό, εκτελούνται από πολλαπλά καταναμεμημένα συστήματα, καθιστώντας την αντιμετώπισή τους δυσκολότερη.
2. Επιθέσεις τύπου Man-in-the-Middle (MitM):
 - ο Ο επιτιθέμενος παρεμβάλλεται στη δυαδική επικοινωνία μεταξύ κανονικών χρηστών, παρέχοντας τη δυνατότητα παρακολούθησης ή τροποποίησης των πληροφοριών χωρίς να γίνεται αντιληπτό από τους συμμετέχοντες.
3. Επιθέσεις Δικτύου (Network Attacks):
 - ο Packet Sniffing και IP Spoofing: Αυτές οι μέθοδοι χρησιμοποιούνται για την απόσπαση ευαίσθητων πληροφοριών μέσω μη εξουσιοδοτημένης πρόσβασης.
4. Επιθέσεις σε Συστήματα και Υπηρεσίες (System and Service Attacks):
 - ο Μέσω τεχνικών Exploits ή Buffer Overflow, οι επιτιθέμενοι εκμεταλλεύονται τις ευπάθειες του συστήματος.
5. Επιθέσεις με Social Engineering:
 - ο Χρησιμοποιώντας Phishing ή Pretexting, δηλαδή την επινόηση ψεύτικων σεναρίων, αποσπώνται απόρρητες πληροφορίες από τους χρήστες.

6. Επιθέσεις Ανίχνευσης (Reconnaissance Attacks):

- ο Μέθοδοι όπως Scanning και Fingerprinting χρησιμοποιούνται για την ανίχνευση και συλλογή πληροφοριών, με σκοπό την ανεύρεση αδυναμιών στο σύστημα.

7. Επιθέσεις Εισβολής (Intrusion Attacks):

- ο Brute Force Attacks: απόσπαση κωδικών με δοκιμή όλων των πιθανών συνδυασμών και SQL Injection: συλλογή πληροφοριών με στόχο την ανεύρεση αδυναμιών, κυριαρχούν σε αυτή την κατηγορία.

8. Λογική Βόμβα:

- ο Ενεργοποιείται όταν πληρούνται συγκεκριμένες προϋποθέσεις.

9. Αποστολή Ανεπιθύμητης Αλληλογραφίας (Spam):

- ο Περιλαμβάνει την αποστολή μεγάλου όγκου ανεπιθύμητων μηνυμάτων.

10. Botnet:

- ο Ένα δίκτυο υπολογιστών μολυσμένο και απομακρυσμένα ελεγχόμενο, το οποίο χρησιμοποιείται για συντονισμένες επιθέσεις ή άλλες κακόβουλες δραστηριότητες.

Τα δύο πιο γνωστά εργαλεία επιθέσεων είναι το Sniffer τα Εργαλεία Κατάχρησης (Abuse Tools). Το Sniffer παρέχει συνεχή παρακολούθηση και καταγραφή της κυκλοφορίας του δικτύου. Τα Abuse Tools αποτελεί ένα γενικό όρο των εργαλείων εκμετάλλευσης αδυναμιών και διεξαγωγής επιθέσεων, όπως εργαλεία για σπάσιμο κωδικών πρόσβασης, sniffers και εργαλεία για επιθέσεις τύπου DDoS.

Το κακόβουλο λογισμικό περιέχει και άλλες μεθόδους όπως ιούς (viruses), σκουλήκια (worms), δούρειοι ίπποι (trojan horses), ransomware, spyware, adware, rootkits, keylogger και backdoor. Αναλυτικότερα:

1. Ιοί (viruses): Προγράμματα που προσκολλώνται κακόβουλα σε αρχεία και εξαπλώνονται κατά την εκτέλεσή τους.
2. Σκουλήκια (worms): Αυτοαναπαραγόμενα προγράμματα που εξαπλώνονται στο δίκτυο χωρίς να απαιτείται η εκτέλεσή τους από τον χρήστη.
3. Δούρειοι Ίπποι (trojan horses): Προγράμματα που εμφανίζονται νόμιμα, αλλά περιέχουν κακόβουλο κώδικα.

4. Ransomware: Λογισμικό που κρυπτογραφεί τα αρχεία του χρήστη και απαιτεί χρηματικό ποσό για την αποκρυπτογράφησή τους.
5. Spyware: Λογισμικό παρακολούθησης που συλλέγει τις δραστηριότητες του χρήστη χωρίς τη συγκατάθεσή του.
6. Adware: Λογισμικό που εμφανίζει ανεπιθύμητες διαφημίσεις, συχνά συνδεδεμένο με το spyware.
7. Rootkits: Λογισμικό που στοχεύει στην απόκρυψη πληροφοριών και την απόκτηση δικαιωμάτων διαχειριστή από τον επιτιθέμενο.
8. Keylogger: Λογισμικό που καταγράφει πληκτρολογήσεις για τη συλλογή ευαίσθητων προσωπικών δεδομένων.
9. Backdoor: Λογισμικό που επιτρέπει την παράκαμψη των κανονικών διαδικασιών ελέγχου, προσφέροντας στον επιτιθέμενο πρόσβαση στο σύστημα.

4.1 ΕΥΠΑΘΕΙΕΣ TCP/IP

Το αρχικά σχεδιασμένο δίκτυο ήταν απλό και δεν προοριζόταν για εκτενή και πολύπλοκη χρήση, όπως χαρακτηρίζεται σήμερα. Το βασικό πρόβλημα έγκειται στις εγγενείς αδυναμίες στον σχεδιασμό του που είναι δύσκολο να εξαλειφθούν. Πολλά από τα πρώιμα πρωτόκολλα της σουίτας TCP/IP θεωρούνται σήμερα ανασφαλή και ευάλωτα σε ποικίλες μορφές επιθέσεων, όπως η υποκλοπή κωδικών πρόσβασης και η άρνηση παροχής υπηρεσιών (DoS). Για παράδειγμα, το TCP/IP περιλαμβάνει τα εργαλεία r-utilities του Berkeley, όπως το rlogin (απομακρυσμένη σύνδεση), το rcp (απομακρυσμένη αντιγραφή) και το rsh (απομακρυσμένη εκτέλεση εντολών). Αυτά τα εργαλεία σχεδιάστηκαν αρχικά για πρόσβαση σε UNIX μηχανές χωρίς τη χρήση κωδικού πρόσβασης. Παρόλο που προσφέρουν κάποια πλεονεκτήματα, η χρήση τους μπορεί να καταστήσει το σύστημα εξαιρετικά ανασφαλές. Η εγγενής τους φύση στον σχεδιασμό σημαίνει ότι παρουσιάζουν καθολικές ευπάθειες, οι οποίες επηρεάζουν όχι μόνο το Διαδίκτυο αλλά και τα περισσότερα τοπικά δίκτυα. Το γεγονός ότι το TCP over IP είναι πρωτόκολλο χαμηλού επιπέδου συνεπάγεται ότι όλα τα πρωτόκολλα υψηλότερου επιπέδου, όπως το HTTP, το Telnet και το SMTP, είναι ευάλωτα μέσω κληρονομικότητας. Για παράδειγμα, η επίθεση του σκουληκιού Morris του 1988 αποτέλεσε μια σημαντική προειδοποίηση για την ασφάλεια του πρώιμου Διαδικτύου, το οποίο είχε αρχικά σχεδιαστεί σε μια εποχή πριν την ευρεία χρήση των προσωπικών υπολογιστών και των έξυπνων συσκευών.

Το σκουλήκι αυτό ήταν ένα από τα πρώτα κακόβουλα λογισμικά που εξαπλώθηκε μέσω του Διαδικτύου, προμηνύοντας μελλοντικές επιθέσεις όπως το Melissa και το Code Red. Παρότι ο αρχικός σκοπός του ήταν απλώς να μετρηθεί το μέγεθος του ARPANET, το σκουλήκι εκμεταλλεύτηκε γνωστές ευπάθειες σε προγράμματα όπως το sendmail και το finger, προκαλώντας σοβαρά προβλήματα απόδοσης σε μολυσμένα συστήματα. Παρότι το πρωτόκολλο TCP είναι προσανατολισμένο στη σύνδεση, μπορούν να αξιοποιηθούν τα τρωτά του σημεία, με βάση επιθέσεις όπως TCP SYN flooding, κατάχρηση συνόδου TCP ή επίθεση επαναφοράς TCP.

4.2 ΕΥΠΑΘΕΙΕΣ ΠΡΩΤΟΚΟΛΛΟΥ TCP

Τα τμήματα του πρωτοκόλλου TCP (Transmission Control Protocol) βρίσκονται ενσωματωμένα σε πακέτα IP (Internet Protocol), με την επικεφαλίδα TCP να ακολουθεί αμέσως μετά την επικεφαλίδα IP, παρέχοντας συγκεκριμένες πληροφορίες για το TCP. Το πρωτόκολλο TCP είναι σχεδιασμένο για αξιόπιστη επικοινωνία και περιλαμβάνει επιβεβαιώσεις για τη διασφάλιση της παράδοσης δεδομένων, καθώς και μηχανισμούς ελέγχου ροής και κατάστασης για την αποφυγή συμφόρησης. Οι διαδικασίες αυτές συνεισφέρουν στην αξιοπιστία του, με αντίτιμο τον μεγαλύτερο φόρτο σε σύγκριση με το UDP (User Datagram Protocol). Για την καθιέρωση σύνδεσης TCP, απαιτείται η διαδικασία της τριπλής χειραψίας (three-way handshake), ενώ για την ομαλή τερματισμό της σύνδεσης χρησιμοποιείται η τετραπλή χειραψία (four-way handshake). Παρά την αξιοπιστία του, το πρωτόκολλο TCP παρουσιάζει ευπάθειες που μπορούν να αξιοποιηθούν σε διάφορους τύπους επιθέσεων. Χαρακτηριστική επίθεση αποτελεί το TCP SYN Flooding που είναι μια μορφή DoS κατά την οποία εξαντλούνται οι πόροι του διακομιστή μέσω ατελών αιτημάτων σύνδεσης. Η κατάχρηση συνόδου TCP, δηλαδή η αθέμιτη παρακολούθηση και έλεγχος ενεργών συνόδων TCP για απόσπαση πληροφοριών ή χειραγώγηση, και οι επιθέσεις επαναφοράς TCP, δηλαδή τον αυθαίρετο τερματισμό συνόδων TCP μέσω αποστολής πακέτων επαναφοράς, αποτελούν τα τρωτά σημεία της ασφάλειας του δικτύου.

4.3 ΕΥΠΑΘΕΙΕΣ ΠΡΩΤΟΚΟΛΛΟΥ IP

Το Πρωτόκολλο Internet (IP) είναι ένα πρωτόκολλο χωρίς σύνδεση που χρησιμοποιείται κυρίως για τη δρομολόγηση πληροφοριών στο Διαδίκτυο. Σκοπός του είναι η βέλτιστη παράδοση των δεδομένων στον προορισμό τους χωρίς εγγυήσεις αξιοπιστίας. Για την παροχή υπευθυνότητας και αξιοπιστίας, το IP βασίζεται στα πρωτόκολλα ανώτερου επιπέδου της σουίτας TCP/IP. Τα πρωτόκολλα αυτά χρησιμοποιούν τη διεύθυνση προέλευσης των πακέτων για την αναγνώριση του αποστολέα και για την επικοινωνία μαζί του μέσω της αποστολής απαντήσεων σε αυτή τη διεύθυνση πηγής. Ωστόσο, επειδή το IP δεν επαληθεύει την αυθεντικότητα της διεύθυνσης προέλευσης στα πακέτα, είναι δυνατόν να παραποιηθεί η διεύθυνση αυτή, καθιστώντας το πακέτο να φαίνεται ότι προέρχεται από μια άλλη, παραποιημένη διεύθυνση. Εκτός από την παραποίηση σε επίπεδο IP, το spoofing μπορεί να εφαρμοστεί και σε άλλα επίπεδα, όπως το ARP spoofing, το οποίο συνδέει μια ψευδή διεύθυνση MAC με τη διεύθυνση IP ενός νόμιμου κεντρικού υπολογιστή για να εκτρέψει την κυκλοφορία. Το πρωτόκολλο SMTP είναι επίσης ευάλωτο στην παραποίηση διεύθυνσης αποστολέα, επιτρέποντας την αποστολή email με ψευδείς διευθύνσεις προέλευσης. Ως αποτέλεσμα, η εμπιστοσύνη σε οποιοδήποτε πακέτο στο διαδίκτυο δεν μπορεί να θεωρείται δεδομένη, και η αξιοπιστία πρέπει να επιτυγχάνεται μέσω εξελιγμένων μηχανισμών ασφαλείας που επιτρέπουν την επιβολή πολιτικών ασφάλειας και την ταξινόμηση της κίνησης. Οι ευπάθειες που βασίζονται στις διευθύνσεις IP περιλαμβάνουν την επίθεση Man-in-the-Middle, Πειρατεία συνόδου (Session Hijacking), όπου αποκτάται παράνομα πρόσβαση σε ενεργές συνδέσεις επιτρέποντας στον επιτιθέμενο να υποδυθεί έναν νόμιμο χρήστη, η παραποίηση διεύθυνσης IP (IP Spoofing), DDoS, επίθεση smurf, όπου το χακαρισμένο μηχάνημα κατακλύζεται από εισερχόμενες απαντήσεις ping, και επιθέσεις εξάντλησης πόρων, ώστε η μνήμη και το εύρος ζώνης του συστήματος να καταστείτε αδύναμο ή μη αποκρινόμενο.

4.4 ΕΥΠΑΘΕΙΕΣ ΠΡΩΤΟΚΟΛΛΟΥ ICMP

Το Πρωτόκολλο Μηνυμάτων Ελέγχου Διαδικτύου (ICMP) είναι ένα πρωτόκολλο χωρίς σύνδεση που λειτουργεί στο επίπεδο δικτύου, χωρίς τη χρήση αριθμών θύρας, σε αντίθεση με τα πρωτόκολλα TCP και UDP. Ο κύριος στόχος του ICMP δεν είναι η μεταφορά δεδομένων, αλλά η μεταφορά διαγνωστικών μηνυμάτων και η αναφορά σφαλμάτων, διασφαλίζοντας έτσι

την αποδοτική λειτουργία των δικτυακών συνδέσεων. Συγκεκριμένα, το ICMP χρησιμοποιείται για να ενημερώνει τον αποστολέα όταν υπάρχουν ζητήματα κατά την παράδοση δεδομένων, όπως αδυναμία πρόσβασης σε διαδρομές, κεντρικούς υπολογιστές, ή θύρες. Τυπικές χρήσεις του ICMP περιλαμβάνουν τη χρήση εντολών όπως το ping (ICMP echo request) και το traceroute (ICMP TTL expired in transit), οι οποίες είναι βασικά εργαλεία για τους διαχειριστές δικτύων για τη διάγνωση προβλημάτων συνδεσιμότητας. Η πλειονότητα της κίνησης ICMP προέρχεται από δρομολογητές, τείχη προστασίας και τελικά σημεία, με σκοπό την παρακολούθηση και διάγνωση των δικτυακών προβλημάτων. Παρά το γεγονός ότι το ICMP αποτελεί σημαντικό εργαλείο για τη διάγνωση δικτύου, ορισμένες φορές επιλέγεται να μπλοκαριστεί ώστε να αποτραπούν οι επιτιθέμενοι να συλλέξουν πληροφορίες για το εσωτερικό του δικτύου. Η κίνηση ICMP είναι ιδιαίτερα σημαντική, είτε αυτή παράγεται από χρήστες, είτε από δικτυακές συσκευές που αναφέρουν σφάλματα. Στην πρώτη περίπτωση, είναι χρήσιμο να εντοπιστούν ενδεχόμενες προσπάθειες σαρώσεων δικτύου από χρήστες εντός του δικτύου, ενώ στη δεύτερη περίπτωση, τα μηνύματα ICMP από τις συσκευές δικτύου μπορούν να βοηθήσουν στον εντοπισμό και την αντιμετώπιση προβλημάτων ή διακοπών στη λειτουργία του δικτύου. Οι ευπάθειες του ICMP περιλαμβάνουν την αναγνώριση δικτύων και λειτουργικών συστημάτων, επιθέσεις ανακατεύθυνσης και σάρωσης δικτύου, δημιουργία κρυφών σηράγγων για παράκαμψη τειχών προστασίας, και επιθέσεις άρνησης υπηρεσιών (DoS) όπως οι επιθέσεις ICMP flood και smurf.

4.5 ΕΥΠΑΘΕΙΕΣ ΠΡΩΤΟΚΟΛΛΟΥ UDP

Το UDP (User Datagram Protocol) είναι ένα πρωτόκολλο επιπέδου μεταφοράς χωρίς σύνδεση που λειτουργεί ως διεπαφή μεταξύ του IP και των διεργασιών ανώτερου επιπέδου. Σε αντίθεση με το TCP, το UDP δεν ενσωματώνει μηχανισμούς αξιοπιστίας, έλεγχο ροής ή ανάκτηση σφαλμάτων, επιτρέποντάς του να λειτουργεί με λιγότερη πολυπλοκότητα και χαμηλότερη επιβάρυνση στο δίκτυο. Οι επικεφαλίδες των τμημάτων UDP περιλαμβάνουν μόνο τους αριθμούς θύρας πηγής και προορισμού, ένα άθροισμα ελέγχου (checksum), και το μήκος του τμήματος. Το UDP είναι κατάλληλο για εφαρμογές όπου οι μηχανισμοί αξιοπιστίας του TCP δεν είναι απαραίτητοι, καθώς μπορεί να υποστηρίζονται από το πρωτόκολλο ανώτερου επιπέδου. Χρησιμοποιείται ευρέως σε πρωτόκολλα όπως τα NFS, SNMP, DNS, TFTP και σε

εφαρμογές που απαιτούν την επεξεργασία σε πραγματικό χρόνο, όπως διαδικτυακά παιχνίδια, streaming πολυμέσων, και VoIP. Το UDP, αν και απλό και αποδοτικό, παρουσιάζει αρκετές ευπάθειες ασφαλείας. Η έλλειψη μηχανισμών αξιοπιστίας και η δυνατότητα επαναυπολογισμού του αθροίσματος ελέγχου καθιστούν το UDP ευάλωτο σε επιθέσεις υποκλοπής και τροποποίησης δεδομένων. Επιπλέον, δεν περιλαμβάνονται διαδικασίες επαλήθευσης της πηγής των πακέτων, επιτρέποντας έτσι την πλαστογράφηση (spoofing) και τη δημιουργία ψευδών πακέτων. Επιθέσεις όπως το UDP flooding και το SQL Slammer worm εκμεταλλεύονται αυτές τις ευπάθειες για την πρόκληση εξάντλησης πόρων και DoS. Ειδικά λογισμικά μπορούν να χρησιμοποιηθούν για την εκτέλεση τέτοιων επιθέσεων, ενώ ο κώδικας εκμετάλλευσης παραμένει διαθέσιμος, επιδεινώνοντας τις επιπτώσεις.

5.1 ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Η έννοια της μηχανικής μάθησης βασίζεται στην εκπαίδευση των υπολογιστών ώστε να μαθαίνουν από τα διαθέσιμα δεδομένα και στην δυνατότητα να βρίσκονται σε θέση στην οποία δύναται να προβλέψουν σχετικά με τα δεδομένα αυτά. {Με άλλα λόγια, αντί να γράφει ο προγραμματιστής σαφείς κανόνες για τον τρόπο επίλυσης ενός συγκεκριμένου προβλήματος, δίνει εντολή στον υπολογιστή να μάθει από παραδείγματα}[1].

5.2 ΕΠΙΒΛΕΠΟΜΕΝΗ ΚΑΙ ΜΗ ΕΠΙΒΛΕΠΟΜΕΝΗ ΜΑΘΗΣΗ

Δύο σημαντικές προσεγγίσεις τα πλαίσια της τεχνητής νοημοσύνης και της μηχανικής μάθησης, είναι η επιβλεπόμενη (supervised) και μη επιβλεπόμενη (unsupervised) μάθηση. Η κυριότερη διαφορά μεταξύ των δύο, είναι ότι η πρώτη εκμεταλλεύεται σημασμένα δεδομένα (με ετικέτα - labeled) για να πραγματοποιήσει προβλέψεις, ενώ στη δεύτερη δεν υπάρχουν βοηθητικές ετικέτες. Το κάθε είδος μάθησης υπερτερεί του άλλου ανάλογα με τις ανάγκες του προβλήματος.

Στην εποπτευόμενη μάθηση, τα σύνολα δεδομένων που χρησιμοποιούνται, είναι σχεδιασμένα ώστε να εκπαιδεύουν τους αλγορίθμους μηχανικής μάθησης χρησιμοποιώντας τις δεδομένες ετικέτες (είσοδος), στοχεύοντας τη σωστή τελική απόκριση (έξοδος). Έτσι, το μοντέλο μπορεί

να μαθαίνει και να αυξήσει την ακρίβειά του κατά την πάροδο του χρόνου. Τα προβλήματα που λύνει η εποπτευόμενη μάθηση είναι δύο κατηγοριών: προβλήματα ταξινόμησης και προβλήματα παλινδρόμησης. Τα προβλήματα ταξινόμησης αφορούν τον διαχωρισμό των δεδομένων σε κατηγορίες ή κλάσεις. Ο στόχος είναι η εκπαίδευση του μοντέλου για την πρόβλεψη της κλάσης στην οποία ανήκει το νέο δείγμα δεδομένων. Τα προβλήματα αυτά είναι συνήθως δυαδικά (binary classification) ή πολυκατηγορικά (multiclass classification). Μερικά παραδείγματα είναι η ανίχνευση επιθέσεων σε δίκτυα, η ταξινόμηση ανεπιθύμητων μηνυμάτων, αναγνώριση χειρόγραφων αριθμητικών ψηφίων. Σε αντίθεση με την ταξινόμηση, τα προβλήματα της παλινδρόμησης ασχολούνται με συνεχείς τιμές και όχι κατηγορίες. Τα μοντέλα που δημιουργούνται στοχεύουν στην πρόβλεψη των αριθμητικών τιμών βασισμένα με νέα δείγματα δεδομένων. Μερικά παραδείγματα είναι η πρόβλεψη τιμών κατοικιών βάσει διάφορων χαρακτηριστικών, η πρόβλεψη της θερμοκρασίας ενός χώρου βάσει δεδομένων αισθητήρων, η εκτίμηση της ζήτησης προϊόντων βάσει του ιστορικού πωλήσεων. Σημαντικός διαχωρισμός ανάμεσα στα προβλήματα αποτελούν οι αλγόριθμοι και οι τεχνικές που χρησιμοποιούνται. Αλγόριθμοι όπως οι Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM) και K-Nearest Neighbors (KNN), είναι χαρακτηριστικοί της ταξινόμησης. Ενώ, αλγόριθμοι όπως οι Linear Regression, Polynomial Regression, Support Vector Regression (SVR) και Ridge Regression, είναι αντίστοιχα χαρακτηριστικοί της παλινδρόμησης.

Στη μη επιβλεπόμενη μάθηση, χρησιμοποιούνται αλγόριθμοι μηχανικής μάθησης με σκοπό την ανακάλυψη μοτίβων στο σύνολο δεδομένων χωρίς την παρέμβαση του ανθρώπου. Τα μαθησιακά μοντέλα της μη επιβλεπόμενης μάθησης ανήκουν σε τρεις κατηγορίες λύσης προβλημάτων: την ομαδοποίηση, τη συσχέτιση και τη μείωση διαστάσεων.

Τα προβλήματα ομαδοποίησης αναφέρονται στο διαχωρισμό ενός συνόλου δεδομένων σε ομάδες (clusters), ώστε κάθε ομάδα να έχει δεδομένα πιο όμοια μεταξύ τους συγκριτικά με δεδομένα από άλλες ομάδες. Μερικά παραδείγματα χρήσης της τεχνικής αυτής είναι η ομαδοποίηση πελατολογίου βάσει των αγορών τους, η τμηματοποίηση των ανεπτυγμένων στρατηγικών marketing, η αναγνώριση δομών και μοτίβων σε βιολογικά δεδομένα. Κοινές τεχνικές όπως K-means Clustering, Hierarchical Clustering και DBSCAN (Density – Based Spatial Clustering of Applications with Noise), χαρακτηρίζουν την αρχή της ομαδοποίησης.

Όσον αφορά τα προβλήματα συσχέτισης, εντοπίζονται οι κανόνες συσχέτισης ώστε να περιγραφεί η σύνδεση μεταξύ των δεδομένων. Εφαρμόζονται κυρίως στην εξόρυξη δεδομένων ιατρικών αρχείων με στόχο την ανεύρεση συσχετίσεων μεταξύ συμπτωμάτων και ασθενειών, συσχετίσεις μεταξύ προϊόντων και μαζικών αγορών. Οι συνήθεις τεχνικές που χρησιμοποιούνται είναι οι αλγόριθμοι Apriori Algorithm, Eclat Algorithm, FP-Growth (Frequent Pattern Growth). Κύρια αιτία των προβλημάτων που στοχεύουν στη μείωση διαστάσεων είναι η μείωση του αριθμού των τυχαίων μεταβλητών υπό εξέταση. Η λύση υπολογίζεται στην απλοποίηση των δεδομένων καθώς διατηρείται όσο το δυνατόν περισσότερη από την αρχική πληροφορία. Η ευκολότερη προεπεξεργασία των δεδομένων, η μείωση θορύβου και η αποφυγή της curse of dimensionality είναι τα κυριότερα προτερήματα της αντιμετώπισης του προβλήματος. Κάποια παραδείγματα χρήσης της μεθόδου είναι η μείωση την χαρακτηριστικών ενός συνόλου δεδομένων για την αύξηση της αποδοτικότητας των αλγορίθμων μηχανικής μάθησης, η οπτικοποίηση δεδομένων σε 2D ή 3D χώρους, η προεπεξεργασία δεδομένων. Αντίστοιχα, οι τεχνικές που επιστρατεύονται τις μεθόδους μείωσης της διάστασης είναι οι Principal Component Analysis (PCA), t-Distributed Stochastic Embedding (t-SNE), Linear Discriminant Analysis (LDA), Autoencoders (σε νευρωνικά δίκτυα).

Εκτός από τα δύο βασικά αυτά είδη μηχανικής μάθησης, υπάρχουν και λοιπές κατηγορίες. Η μάθηση με ενίσχυση περιλαμβάνεται στην επιβλεπόμενη μάθηση, ωστόσο δεν υποστηρίζει άμεση ανατροφοδότηση σχετικά με τα αποτελέσματα που προκύπτουν από κάθε επιλογή ξεχωριστά. Στη σχετικά επιβλεπόμενη μάθηση χαρακτηρίζονται ορισμένα εκ των διαθέσιμων δεδομένων.

6 ΑΛΓΟΡΙΘΜΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΣΤΗΝ ΑΝΙΧΝΕΥΣΗ ΕΠΙΘΕΣΕΩΝ

Η ανίχνευση επιθέσεων σε δίκτυα μέσω των εφαρμογών της μηχανικής μάθησης, είναι ένας σημαντικά εξελισσόμενος τομέας καθώς οι επιθέσεις από κακόβουλα λογισμικά συνεχώς αυξάνουν και γίνονται ολοένα και πιο πολύπλοκες. Καθώς οι κίνδυνοι που διατρέχονται είναι πολλαπλοί, έχουν σχεδιαστεί συστήματα ειδικά για την αντιμετώπιση των διαφόρων μορφών επίθεσης, όπως και την πρόληψη επιθέσεων εμποδίζοντας τη μόλυνση από κακόβουλο

λογισμικό και άλλες δικλείδες επίθεσης. Τα συστήματα ανίχνευσης εισβολής (IDS) αποτελούν την προστασία αυτή της πρόληψης των επιθέσεων και παίζουν κρίσιμο ρόλο στην προστασία των δικτύων από εξωτερικές απειλές. Στόχος τους λοιπόν είναι να λειτουργούν ως αμυντικός μηχανισμός, προσφέροντας προστασία έναντι επιθέσεων που στοχεύουν υπολογιστικά συστήματα στο Διαδίκτυο. Σε αντίθεση με τα συμβατικά τείχη προστασίας, τα IDS είναι ικανά να εντοπίζουν διάφορους τύπους κακόβουλης δραστηριότητας και κατάχρησης συστημάτων, μια δυνατότητα που δεν παρέχουν τα παραδοσιακά τείχη προστασίας. Η ανίχνευση εισβολών βασίζεται στην υπόθεση ότι οι επιτιθέμενοι παρουσιάζουν διαφορετικές συμπεριφορές σε σχέση με τους νόμιμους χρήστες. Γενικά, τα IDS διακρίνονται σε δύο βασικές κατηγορίες: την ανίχνευση ανωμαλιών και την ανίχνευση κακής χρήσης, που συχνά ονομάζεται και ανίχνευση βασισμένη σε υπογραφές. Η ανίχνευση ανωμαλιών προσπαθεί να εντοπίσει πιθανές επιθέσεις, παρακολουθώντας αποκλίσεις από τα συνηθισμένα πρότυπα κανονικής συμπεριφοράς – ροής των δεδομένων. Αντίστοιχα, η ανίχνευση κακής χρήσης βασίζεται στην αναγνώριση γνωστών μοτίβων επιθέσεων ή ευπαθειών των συστημάτων για να εντοπίσει τις εισβολές. Για την ανίχνευση των εισβολών, εφαρμόζονται είτε μεμονωμένες τεχνικές, όπως τα νευρωνικά δίκτυα, οι γενετικοί αλγόριθμοι και οι μηχανές υποστήριξης διανυσμάτων, είτε άλλες προσεγγίσεις που συνδυάζουν διαφορετικές τεχνικές, εφαρμόζοντας υβριδικές ή ενισχυτικές μεθόδους. Αυτές οι τεχνικές χρησιμοποιούνται κυρίως ως ταξινομητές, με σκοπό να διακρίνουν ή να αναγνωρίζουν εάν η εισερχόμενη διαδικτυακή κίνηση είναι νόμιμη ή αποτελεί απόπειρα εισβολής. Έχουν δοκιμαστεί και προταθεί πολλές μέθοδοι και μοντέλα, ωστόσο δε χρησιμοποιούνται όλες με την ίδια συχνότητα και δεν αποδίδουν όλες το ίδιο. Εστιάζοντας στους ευρέως αποδεκτούς αλγορίθμους στην ανίχνευση επιθέσεων σε δίκτυα, παρατηρούνται τόσο αλγόριθμοι επιβλεπόμενης όσο και μη επιβλεπόμενης μάθησης. Στην πρώτη κατηγορία ανήκει ο Support Vector Machines (SVM) όπου η ικανότητά του να ταξινομεί την κυκλοφορία του δικτύου, δείχνει αν είναι κακόβουλη ή όχι. Ο Random Forest και γενικότερα τα δένδρα αποφάσεων χρησιμοποιούνται ευρέως για την ταξινόμηση και ανίχνευση επιθέσεων. Ο Naïve Bayes βασιζόμενος στην θεωρία των πιθανοτήτων, ανιχνεύει απλά και γρήγορα επιθέσεις ιδιαίτερα όταν υπάρχει πληθώρα δεδομένων. Την κατηγορία κλείνουν τα νευρωνικά δίκτυα (neural networks), τα οποία, ιδίως τα βαθιά νευρωνικά δίκτυα, χρησιμοποιούνται καταλλήλως σε πολυσύνθετα συστήματα ανίχνευσης επιθέσεων. Στη δεύτερη κατηγορία, αυτή της μη επιβλεπόμενης μάθησης, οι δύο πιο συχνά παρατηρούμενοι αλγόριθμοι είναι ο k-Means Clustering και ο Isolation Forest. Ο k-means ομαδοποιώντας τα δεδομένα με βάση την

ομοιότητα, χρησιμοποιείται στην ανίχνευση ανωμαλιών δικτύου, ενώ ο isolation forest είναι σχεδιασμένος για την ανίχνευση ανωμαλιών και αποτελεσματικός στην απομόνωση κακόβουλων δραστηριοτήτων σε μεγάλα σύνολα δεδομένων. Μέθοδοι εξίσου γνωστές αλλά λιγότερο εφαρμοσμένες για ανίχνευση επιθέσεων αποτελούν οι KNN (K-Nearest Neighbors), Bayesian Networks και Meta – Heuristic Algorithms. Τέλος, εξίσου χρήσιμες τεχνικές στον τομέα της ανίχνευσης επιθέσεων σε δίκτυα, έχουν αποδειχθεί στην πράξη οι Self-Organizing map (SOM) και οι γενετικοί αλγόριθμοι (συνήθως) για την μη επιβλεπόμενη μάθηση, οι ταξινομητές συνόλων (Ensemble classifiers) στην επιβλεπόμενη μάθηση, ενώ η ασαφής λογική, οι υβριδικοί ταξινομητές και ορισμένες περιπτώσεις γενετικών αλγορίθμων, έχουν εφαρμογές και στις δύο περιπτώσεις μηχανικής μάθησης.

ΕΙΔΙΚΟ ΜΕΡΟΣ

1 ΕΙΣΑΓΩΓΗ

Οι επιθέσεις από κακόβουλο λογισμικό ολοένα και αυξάνονται. Καθημερινά, οι χρήστες κακόβουλου λογισμικού βρίσκουν νέους τρόπους να εισβάλουν στα δεδομένα των νόμιμων χρηστών, διαταράσσοντας τη λειτουργία του δικτύου και του συστήματος είτε αποκτώντας πρόσβαση σε ευαίσθητα δεδομένα είτε διακόπτοντας τη λειτουργία του υπολογιστή, χρησιμοποιώντας ποικίλες νέες μεθόδους. Οι εταιρίες και τα προγράμματα αντιμετώπισης του κακόβουλου λογισμικού βρίσκονται πάντα ένα βήμα πίσω συγκριτικά με τον θύτη, αντιμετωπίζοντας τις καινούριες απειλές. Ο θύτης από την άλλη, τροποποιώντας τις ήδη υπάρχουσες κακόβουλες επιθέσεις ή επινοώντας καινούριες, βρίσκει τρόπους να παραβιάσει τις δικλίδες ασφαλείας του συστήματος και να έχει πρόσβαση σε αυτό και τα δεδομένα του. Το κακόβουλο λογισμικό απειλεί συνεχώς τη λειτουργία του δικτύου και των συστημάτων εκμεταλλευόμενο επακόλουθα και τις τεχνολογικές εξελίξεις, αποτελώντας ίσως τον μεγαλύτερο από τους ψηφιακούς κινδύνους.

Οι έρευνες που ασχολούνται με το κακόβουλο λογισμικό, ιδιαίτερα την ανίχνευση του μέσω αλγορίθμων μηχανικής μάθησης, υστερούν σε απόδοση ή ακρίβεια ανίχνευσης. Η ακρίβεια ανίχνευσης των επιθέσεων μπορεί να είναι εξαιρετικά υψηλή κάτω από βέλτιστες συνθήκες.

Ωστόσο, η πραγματική απόδοση των τεχνικών που χρησιμοποιούνται εξαρτάται από ποικίλους παράγοντες, όπως η ποιότητα και το μέγεθος του dataset που χρησιμοποιείται, η επιλογή των αλγορίθμων, οι τεχνικές προεπεξεργασίας των δεδομένων, και η συντήρηση του μοντέλου. Σημαντικό επίσης ρόλο καταλαμβάνει το πραγματικό περιβάλλον ανίχνευσης έναντι του εργαστηριακού, όπως και η δυνατότητα ή μη της ανίχνευσης αγνώστων (zero-based) επιθέσεων. Σύμφωνα με έρευνες, το ποσοστό ακρίβειας ανίχνευσης επιθέσεων κυμαίνεται από περίπου 90% έως 99%.

2 ΣΚΟΠΟΣ

Η παρούσα πτυχιακή εργασία στοχεύει στην εκπαίδευση και εφαρμογή αλγορίθμων μηχανικής μάθησης σε δεδομένα από ροές δικτύων υπολογιστών για την ανίχνευση επιθέσεων σε αυτά. Σκοπός είναι να εξαχθούν τα κορυφαία κοινά χαρακτηριστικά των επιθέσεων που προκύπτουν από την πλειοψηφία των αλγορίθμων. Ακολουθεί επιπλέον, η δημιουργία ενός νέου μοντέλου με τους ίδιους αλγορίθμους που χρησιμοποιήθηκαν αρχικά, με τη διαφορά ότι η εκπαίδευσή του γίνεται μόνο με τα νέα χαρακτηριστικά που βρέθηκαν, ώστε να ελεγχθούν τα αποτελέσματα του αρχικού πειράματος αλλά και να προσεγγιστεί μια επιπλέον λύση στην ασφάλεια κακόβουλων επιθέσεων.

3.1 ΣΥΛΛΟΓΗ ΔΕΔΟΜΕΝΩΝ

Για την παρούσα εργασία χρησιμοποιήθηκε το Kaggle ως πηγή εύρεσης δεδομένων και συγκεκριμένα το dataset “Network Intrusion Detection”. Το υπό επεξεργασία dataset αποτελείται από ποικίλες προσομοιωμένες εισβολές σε περιβάλλον στρατιωτικού δικτύου. Δημιουργήθηκε μια προσομοίωση ενός κοινού LAN της πολεμικής αεροπορίας των ΗΠΑ το οποίο δέχθηκε εισβολές σε πολλές από τις συνδέσεις του. Η κάθε σύνδεση, το μέγεθος της οποίας είναι περίπου 100 byte κάθε φορά, σημαίνεται είτε ως ‘normal’ είτε ως ‘anomaly’ αναγνωρίζοντας κάποιο τύπο επίθεσης. Οι συνδέσεις (TCP/IP) ορίζονται από περίπου 41 ποιοτικά (3) και ποσοτικά (38) χαρακτηριστικά και μια μεταβλητή κλάση, αυτή του

χαρακτηρισμού της ροής δεδομένων. Οι ροές αυτές ορίζονται από δεδομένα του πραγματικού κόσμου, δηλαδή κανονικά δεδομένα ή δεδομένα επίθεσης. Ο συνολικός αριθμός των εγγραφών (σειρές) είναι 25.192 που ταξινομούνται στα 42 χαρακτηριστικά (στήλες). Τα χαρακτηριστικά μπορεί να χωριστούν και σε επιπλέον κατηγορίες, όπως χαρακτηριστικά σύνδεσης, χαρακτηριστικά σφαλμάτων και επιθέσεων, χαρακτηριστικά εξυπηρετητών και δικτυακής κίνησης και χαρακτηριστικά υπολογισμού ρυθμού.

Αναλυτικότερα, στα χαρακτηριστικά σύνδεσης αναφέρονται τα εξής:

- ο 'Duration': Η διάρκεια της σύνδεσης σε δευτερόλεπτα.
- ο 'Protocol_type': Ο τύπος πρωτοκόλλου που χρησιμοποιείται στη σύνδεση (π.χ., TCP, UDP, ICMP).
- ο 'Service': Η υπηρεσία που χρησιμοποιείται στη σύνδεση (π.χ., http, ftp, smtp).
- ο 'Flag': Η κατάσταση της σύνδεσης (π.χ., SF για επιτυχή σύνδεση, S0 για σύνδεση που ξεκίνησε αλλά δεν ολοκληρώθηκε).
- ο 'Src_bytes': Ο αριθμός των bytes που αποστέλλονται από την πηγή.
- ο 'Dst_bytes': Ο αριθμός των bytes που λαμβάνονται από τον προορισμό.
- ο 'Land': Δυναμική μεταβλητή που υποδεικνύει αν η πηγή και ο προορισμός είναι τα ίδια (1 αν είναι ίδια, 0 αν είναι διαφορετικά).

Τα χαρακτηριστικά σφαλμάτων και επιθέσεων περιλαμβάνουν:

1. 'Wrong_fragment': Ο αριθμός των λανθασμένων θραυσμάτων πακέτων.
2. 'Urgent': Ο αριθμός των επειγόντων πακέτων που αποστέλλονται.
3. 'Hot': Ο αριθμός των "καυτών" δεικτών σύνδεσης, δηλαδή συνδέσεις που σχετίζονται με κακόβουλες ενέργειες, όπως λογαριασμοί που παραβιάστηκαν.
4. 'Num_failed_logins': Ο αριθμός των αποτυχημένων προσπαθειών σύνδεσης.
5. 'Logged_in': Δυναμική μεταβλητή που δείχνει αν ο χρήστης έχει συνδεθεί επιτυχώς (1 αν ναι, 0 αν όχι).
6. 'Num_compromised': Ο αριθμός των παραβιασμένων δεικτών σύνδεσης.

Στη συνέχεια, τα χαρακτηριστικά εξυπηρετητών και δικτυακής κίνησης περιλαμβάνουν:

1. 'Count': Ο αριθμός των συνδέσεων που δημιουργήθηκαν από την ίδια πηγή κατά τη διάρκεια των τελευταίων 2 δευτερολέπτων.

2. 'Srv_count': Ο αριθμός των συνδέσεων στον ίδιο προορισμό υπηρεσίας.
3. 'Serror_rate': Το ποσοστό των συνδέσεων που έχουν καταλήξει με σφάλματα προέλευσης.
4. 'Srv_serror_rate': Το ποσοστό των συνδέσεων στον ίδιο προορισμό υπηρεσίας με σφάλματα προέλευσης.
5. 'Rerror_rate': Το ποσοστό των συνδέσεων που καταλήγουν με σφάλματα προορισμού.
6. 'Srv_rerror_rate': Το ποσοστό των συνδέσεων στον ίδιο προορισμό υπηρεσίας με σφάλματα προορισμού.

Τέλος, τα χαρακτηριστικά υπολογισμού ρυθμού περιλαμβάνουν:

1. 'Same_srv_rate': Το ποσοστό των συνδέσεων που κατέληξαν στην ίδια υπηρεσία.
2. 'Diff_srv_rate': Το ποσοστό των συνδέσεων που κατέληξαν σε διαφορετικές υπηρεσίες.
3. 'Srv_diff_host_rate': Το ποσοστό των συνδέσεων που εξυπηρετούνται από διαφορετικούς εξυπηρετητές.
4. 'Dst_host_count': Ο αριθμός των συνδέσεων σε συγκεκριμένο προορισμό κατά τη διάρκεια του παραθύρου.
5. 'Dst_host_srv_count': Ο αριθμός των συνδέσεων σε συγκεκριμένο προορισμό υπηρεσίας.
6. 'Dst_host_same_srv_rate': Το ποσοστό των συνδέσεων στον ίδιο προορισμό υπηρεσίας.
7. 'Dst_host_diff_srv_rate': Το ποσοστό των συνδέσεων σε διαφορετικές υπηρεσίες στον ίδιο προορισμό.
8. 'Dst_host_same_src_port_rate': Το ποσοστό των συνδέσεων με την ίδια θύρα πηγής.
9. 'Dst_host_srv_diff_host_rate': Το ποσοστό των συνδέσεων σε διαφορετικούς εξυπηρετητές στην ίδια υπηρεσία.
10. 'Dst_host_serror_rate': Το ποσοστό των συνδέσεων στον προορισμό με σφάλματα προέλευσης.
11. 'Dst_host_srv_serror_rate': Το ποσοστό των συνδέσεων στον ίδιο προορισμό υπηρεσίας με σφάλματα προέλευσης.
12. 'Dst_host_rerror_rate': Το ποσοστό των συνδέσεων στον προορισμό με σφάλματα προορισμού.

13. 'Dst_host_srv_error_rate': Το ποσοστό των συνδέσεων στον ίδιο προορισμό υπηρεσίας με σφάλματα προορισμού.

Στα δεδομένα περιλαμβάνονται επίσης στατιστικά στοιχεία, όπως κατηγορίες πρωτοκόλλου (TCP, UDP, ICMP), υπηρεσίες (http, ftp, smtp ή private), και μπορούν να εξαχθούν συμπεράσματα για τη διακύμανση της πληροφορίας (π.χ., src_bytes, dst_bytes).

3.2 ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

Στην αρχική ανάλυση των δεδομένων του dataset, εξετάζοντας την ποιότητα και τις σχέσεις των δεδομένων, παρατηρήθηκε σε διαγράμματα που πραγματοποιήθηκαν (scatter plot, histograms, box plots και pair plots), μια τιμή που απείχε περίπου μια τάξη μεγέθους από τις υπόλοιπες. Για τον λόγο αυτό, για την ανίχνευση και αφαίρεση των ακραίων τιμών στο dataset, χρησιμοποιήθηκε η μέθοδος του κριτηρίου IQR (Interquartile Range). Σύμφωνα με αυτή, μπορούν να προσδιοριστούν οι τιμές που ξεφεύγουν από την τυπική κατανομή των δεδομένων. Στη συνέχεια, αφού εντοπιστούν οι ακραίες τιμές, αφαιρούνται από το πλαίσιο δεδομένων πριν την οπτικοποίηση.

Για την καλύτερη κατανόηση των δεδομένων και την αποφυγή μεροληψίας (bias) που μπορεί να προκύψει από αλληλεξαρτήσεις μεταξύ των χαρακτηριστικών, πραγματοποιήθηκε περαιτέρω ανάλυση της μεταξύ τους συσχέτισης. Μέσω της χρήσης ενός heatmap, εντοπίστηκαν τα χαρακτηριστικά που παρουσίαζαν υψηλή συσχέτιση, δηλαδή αυτά που ο συντελεστής συσχέτισής τους ήταν μεγαλύτερος από 70% (με τιμές άνω του +0.7 για θετική συσχέτιση και κάτω του -0.7 για αρνητική συσχέτιση). Στη συνέχεια, τα ζεύγη αυτών των χαρακτηριστικών συγκρίθηκαν με βάση την πρακτική τους σημασία στο dataset και επιλέχθηκαν αυτά που κατά την ανθρώπινη κρίση φάνηκαν τα πιο σημαντικά για το υπό εξέταση πρόβλημα. Τα υπόλοιπα αφαιρέθηκαν από το σύνολο εκπαίδευσης των αλγορίθμων. Πιο συγκεκριμένα:

1. num_compromised και num_root (1.00):

- ο Το num_compromised αναφέρεται σε πόσα συστήματα έχουν παραβιαστεί.

- Το num_root αναφέρεται σε πόσες root-level ενέργειες έχουν πραγματοποιηθεί.

Και οι δύο μεταβλητές είναι σημαντικές για ανάλυση ασφαλείας. Ωστόσο, το num_root είναι πιο εξειδικευμένο, ενώ το num_compromised μπορεί να δώσει μια πιο ευρεία εικόνα της επίθεσης. Οπότε επιλέγεται το num_compromised.

2. error_rate και srv_error_rate (0.99):

- Το error_rate αναφέρεται στο ποσοστό αποτυχημένων συνδέσεων (SYN errors) από το σύστημα.
- Το srv_error_rate αφορά αποτυχημένες συνδέσεις για συγκεκριμένες υπηρεσίες.

Το error_rate είναι πιο γενικό και καλύπτει όλα τα connections, ενώ το srv_error_rate αφορά πιο συγκεκριμένα τις υπηρεσίες. Για την ανάλυση πιο γενικής συμπεριφοράς, επιλέγεται το error_rate.

3. rerror_rate και srv_rerror_rate (0.99):

- Ομοίως, το ίδιο σκεπτικό όπως παραπάνω. Οπότε επιλέγεται το rerror_rate, ως πιο γενικό.

4. dst_host_error_rate και dst_host_srv_error_rate (0.98):

- Και οι δύο μεταβλητές αναφέρονται σε αποτυχημένες συνδέσεις με τον προορισμό, ωστόσο το dst_host_error_rate καλύπτει όλες τις υπηρεσίες, ενώ το dst_host_srv_error_rate καλύπτει συγκεκριμένες. Επιλέγεται το dst_host_error_rate, ως πιο γενικό.

5. flag_S0 και srv_error_rate (0.98):

- Το flag_S0 αναφέρεται σε SYN packets που δεν έχουν ολοκληρωθεί.
- Το srv_error_rate αναφέρεται σε αποτυχημένες συνδέσεις.

Το srv_error_rate φαίνεται πιο σχετικό με το πρόβλημα.

6. dst_host_rerror_rate και srv_rerror_rate (0.92):

- Το dst_host_rerror_rate αφορά αποτυχημένες συνδέσεις στον προορισμό.

- Το `srv_error_rate` αφορά αποτυχημένες συνδέσεις σε συγκεκριμένες υπηρεσίες.

Το `dst_host_error_rate` επιλέγεται ως πιο γενικό και χρήσιμο.

7. `dst_host_srv_count` και `dst_host_same_srv_rate` (0.90):

- Το `dst_host_srv_count` αφορά τον αριθμό των συνδέσεων σε έναν συγκεκριμένο προορισμό.

- Το `dst_host_same_srv_rate` αφορά το ποσοστό συνδέσεων στον ίδιο προορισμό.

Το `dst_host_srv_count` είναι πιο ακριβές και γενικό.

8. `srv_error_rate` και `flag_REJ` (0.84):

- Το `srv_error_rate` αφορά αποτυχημένες συνδέσεις σε συγκεκριμένες υπηρεσίες.

- Το `flag_REJ` σημαίνει ότι οι συνδέσεις απορρίφθηκαν.

Το `flag_REJ` είναι πιο σαφές γιατί εκφράζει απορρίψεις συνδέσεων.

9. `hot` και `is_guest_login` (0.83):

- Το `hot` αναφέρεται σε επιθετικές δραστηριότητες (π.χ., δημιουργία κελύφους).

- Ενώ το `is_guest_login` δείχνει αν η σύνδεση είναι με `guest` λογαριασμό.

Το `hot` είναι πιο σχετικό με δραστηριότητες που σχετίζονται με επιθέσεις.

10. `flag_SF` και `same_srv_rate` (0.83):

- Το `flag_SF` υποδεικνύει επιτυχείς συνδέσεις.

- Το `same_srv_rate` αφορά το ποσοστό συνδέσεων στον ίδιο διακομιστή.

Το `flag_SF` είναι πιο σαφές για την ανάλυση της επιτυχίας ή αποτυχίας συνδέσεων.

11. `service_ftp` και `is_guest_login` (0.81):

- Το `service_ftp` αναφέρεται στη χρήση της υπηρεσίας FTP.

- Το `is_guest_login` δείχνει αν έγινε σύνδεση με `guest` λογαριασμό.

Η `is_guest_login` είναι πιο γενική και χρήσιμη για τον εντοπισμό μη ασφαλών συνδέσεων.

12. `dst_host_same_srv_rate` και `same_srv_rate` (0.79):

- ο Και οι δύο μεταβλητές αναφέρονται στο ποσοστό συνδέσεων στον ίδιο διακομιστή, αλλά το `same_srv_rate` είναι πιο γενικό.

13. `num_compromised` και `su_attempted` (0.78):

- ο Το `num_compromised` αναφέρεται στον αριθμό παραβιασμένων συστημάτων.
- ο Το `su_attempted` αναφέρεται σε απόπειρες χρήσης της εντολής `su`.

Το `num_compromised` δίνει μια πιο συνολική εικόνα της παραβίασης.

14. `service_domain_u` και `protocol_type_udp` (0.76):

- ο Το `service_domain_u` αφορά τη χρήση της υπηρεσίας `domain_u`.
- ο Το `protocol_type_udp` αναφέρεται στο πρωτόκολλο UDP.

Το `protocol_type_udp` είναι πιο σχετικό για τη δικτυακή ανάλυση, καθώς το πρωτόκολλο είναι θεμελιώδες.

15. `flag_SF` και `class_normal` (0.76):

- ο Το `flag_SF` υποδεικνύει επιτυχείς συνδέσεις.
- ο Το `class_normal` αφορά την κατηγορία κανονικής λειτουργίας.

Το `class_normal` φαίνεται πιο σχετικό με τον στόχο ταξινόμησης

16. `service_http` και `dst_host_srv_count` (0.74):

- ο Το `service_http` αφορά τη χρήση της υπηρεσίας HTTP.
- ο Το `dst_host_srv_count` αφορά τον αριθμό συνδέσεων σε έναν προορισμό.

Το `dst_host_srv_count` είναι πιο γενικό για την ανάλυση συνδέσεων.

17. `dst_host_error_rate` και `flag_SF` (-0.74):

- ο Το `dst_host_error_rate` αφορά αποτυχημένες συνδέσεις στον προορισμό.
- ο Το `flag_SF` υποδεικνύει επιτυχείς συνδέσεις.

Το `dst_host_error_rate` είναι πιο σχετικό για την ανάλυση αποτυχημένων συνδέσεων.

18. `protocol_type_tcp` και `protocol_type_udp` (-0.77):

- ο Και οι δύο μεταβλητές αφορούν διαφορετικά πρωτόκολλα.

Επιλέγεται το `protocol_type_tcp`, καθώς είναι το πιο κοινό πρωτόκολλο δικτύου για πολλές υπηρεσίες.

19. `dst_host_srv_error_rate` και `srv_error_rate` (0.98):

- ο Το `dst_host_srv_error_rate` αναφέρεται σε αποτυχημένες συνδέσεις στον προορισμό για συγκεκριμένες υπηρεσίες.
- ο Το `srv_error_rate` αφορά γενικά αποτυχημένες συνδέσεις σε όλες τις υπηρεσίες.

Το `srv_error_rate` επιλέγεται ως πιο γενικό.

20. `dst_host_srv_error_rate` και `flag_S0` (0.98):

- ο Το `dst_host_srv_error_rate` καλύπτει αποτυχημένες συνδέσεις για συγκεκριμένες υπηρεσίες στον προορισμό.
- ο Το `flag_S0` δείχνει συνδέσεις που δεν ολοκληρώθηκαν (SYN flag).

Το `flag_S0` είναι πιο σημαντικό για την ανίχνευση επιθέσεων τύπου SYN flood.

21. `dst_host_srv_error_rate` και `error_rate` (0.98):

- ο Το `dst_host_srv_error_rate` αφορά συγκεκριμένες υπηρεσίες.
- ο Το `error_rate` καλύπτει αποτυχημένες συνδέσεις γενικά σε όλες τις υπηρεσίες.

Το `error_rate` είναι πιο γενικό και χρήσιμο για την ανάλυση επιθέσεων.

22. `flag_S0` και `error_rate` (0.98):

- ο Το `flag_S0` δείχνει συνδέσεις που δεν ολοκληρώθηκαν.
- ο Το `error_rate` αφορά το ποσοστό αποτυχημένων συνδέσεων.

Και οι δύο μεταβλητές είναι σημαντικές, αλλά το `flag_S0` είναι πιο συγκεκριμένο για επιθέσεις SYN flood.

23. `dst_host_error_rate` και `flag_REJ` (0.81):

- ο Το `dst_host_error_rate` αναφέρεται σε αποτυχημένες συνδέσεις στον προορισμό.

- ο To flag_REJ σημαίνει ότι οι συνδέσεις απορρίφθηκαν.

To flag_REJ είναι πιο άμεσο και σαφές για αποτυχημένες συνδέσεις.

24. dst_host_same_srv_rate και dst_host_srv_count (0.70):

- ο To dst_host_same_srv_rate αφορά το ποσοστό συνδέσεων στον ίδιο διακομιστή.
- ο To dst_host_srv_count δείχνει τον αριθμό των συνδέσεων σε έναν προορισμό.

To dst_host_srv_count επιλέγεται ως πιο γενικό.

25. dst_host_error_rate και flag_SF (-0.74):

- ο To dst_host_error_rate αφορά αποτυχημένες συνδέσεις στον προορισμό.
- ο To flag_SF δείχνει επιτυχείς συνδέσεις.

To dst_host_error_rate επιλέγεται ως πιο σχετικό.

26. dst_host_srv_error_rate και flag_SF (-0.75):

- ο To dst_host_srv_error_rate αφορά αποτυχημένες συνδέσεις για συγκεκριμένες υπηρεσίες.
- ο To flag_SF δείχνει επιτυχείς συνδέσεις.

To dst_host_srv_error_rate είναι πιο σημαντικό για την ανάλυση αποτυχιών.

27. flag_SF και flag_S0 (-0.75):

- ο To flag_SF δείχνει επιτυχείς συνδέσεις.
- ο To flag_S0 αναφέρεται σε ημιτελείς SYN συνδέσεις.

To flag_S0 είναι πιο κρίσιμο για ανίχνευση επιθέσεων τύπου SYN flood.

28. flag_SF και srv_error_rate (-0.76):

- ο To flag_SF αφορά επιτυχείς συνδέσεις.
- ο To srv_error_rate αναφέρεται σε αποτυχημένες συνδέσεις για υπηρεσίες.

To srv_error_rate είναι πιο σημαντικό για την ανίχνευση επιθέσεων.

29. same_srv_rate και srv_error_rate (-0.76):

- ο To same_srv_rate αφορά την επαναληψιμότητα συνδέσεων σε έναν διακομιστή.
- ο To srv_error_rate αφορά αποτυχημένες συνδέσεις.

To srv_error_rate είναι πιο χρήσιμο για την ανίχνευση σφαλμάτων.

30. error_rate και flag_SF (-0.76):

- ο To error_rate αναφέρεται σε αποτυχημένες συνδέσεις.
- ο To flag_SF αφορά επιτυχείς συνδέσεις.

To error_rate επιλέγεται ως πιο κρίσιμο.

31. same_srv_rate και dst_host_error_rate (-0.76):

- ο To same_srv_rate αφορά την επαναληψιμότητα συνδέσεων.
- ο To dst_host_error_rate αφορά αποτυχημένες συνδέσεις στον προορισμό.

To dst_host_error_rate είναι πιο σχετικό με την ανάλυση αποτυχιών συνδέσεων.

32. flag_S0 και same_srv_rate (-0.77):

- ο To flag_S0 αναφέρεται σε ημιτελείς συνδέσεις.
- ο To same_srv_rate αφορά την επαναληψιμότητα συνδέσεων.

To flag_S0 είναι πιο σημαντικό για την ανίχνευση επιθέσεων SYN flood.

33. dst_host_srv_error_rate και flag_REJ (0.83):

- ο To flag_REJ είναι πιο ξεκάθαρο για την ανίχνευση απορριφθέντων συνδέσεων, οπότε επιλέγεται.

34. dst_host_error_rate και error_rate (0.93):

- ο To error_rate είναι πιο γενικό και καλύπτει μεγαλύτερη γκάμα περιπτώσεων, οπότε το κρατάμε.

35. dst_host_error_rate και srv_error_rate (0.92):

- ο To dst_host_error_rate επιλέγεται ως πιο γενικό.

36. `dst_host_srv_error_rate` και `error_rate` (0.96):

- Το `error_rate` είναι πιο γενικό για τη συνολική ανάλυση.

37. `flag_REJ` και `error_rate` (0.83):

- Το `flag_REJ` είναι πιο σαφές για την ανίχνευση απορριφθέντων συνδέσεων.

38. `service_http` και `logged_in` (0.70):

- Το `service_http` αναφέρεται στη χρήση της υπηρεσίας HTTP.
- Το `logged_in` δείχνει αν ένας χρήστης είναι συνδεδεμένος.

Το `logged_in` είναι πιο γενικό και σχετικό με την ανίχνευση επιτυχημένων συνδέσεων.

39. `logged_in` και `service_ftp` (0.81):

- Το `logged_in` δείχνει αν ένας χρήστης είναι συνδεδεμένος.
- Το `service_ftp` αναφέρεται στη χρήση της υπηρεσίας FTP.

Το `logged_in` επιλέγεται ως πιο γενικό.

40. `su_attempted` και `num_root` (0.77):

- Το `su_attempted` αναφέρεται σε απόπειρες χρήσης της εντολής `su` για την απόκτηση δικαιωμάτων `root`.
- Το `num_root` μετρά πόσες `root` ενέργειες έχουν πραγματοποιηθεί.

Το `num_root` είναι πιο σαφές για ενέργειες που σχετίζονται με την απόκτηση δικαιωμάτων `root`, οπότε το επιλέγουμε.

41. `su_attempted` και `num_compromised` (0.78):

- Το `num_compromised` προσφέρει μια συνολική εικόνα των παραβιασμένων συστημάτων, οπότε και επιλέγεται.

42. `same_srv_rate` και `dst_host_srv_count` (0.70):

- Το `dst_host_srv_count` είναι πιο χρήσιμο για την παρακολούθηση του αριθμού συνδέσεων.

43. flag_SF και same_srv_rate (0.83):

- ο Το flag_SF επιλέγεται ως πιο σημαντικό για την ανάλυση της επιτυχίας των συνδέσεων.

Συγκρίνοντας, ελέγχοντας και διορθώνοντας τις επικαλύψεις, καταλήγουμε συνοψίζοντας στις εξής μεταβλητές που τελικά θα αφαιρεθούν: [num_root, srv_error_rate, srv_error_rate, dst_host_srv_error_rate, flag_S0, dst_host_same_srv_rate, is_guest_login, service_domain_u, flag_SF, service_http, protocol_type_udp, su_attempted, service_ftp, error_rate, dst_host_srv_error_rate, dst_host_error_rate, error_rate].

Για τις στήλες που περιείχαν αριθμητικά δεδομένα, υπολογίστηκε επιπλέον η τυπική απόκλιση. Τα αποτελέσματα αυτής της ανάλυσης, έδειξαν ότι δύο από τις στήλες, οι num_outbound_cmds και is_host_login, παρουσίαζαν μηδενική τυπική απόκλιση. Αυτό υποδηλώνει ότι δεν παρείχαν καμία πληροφορία, καθώς οι τιμές τους ήταν σταθερές σε όλο το σύνολο δεδομένων. Ως εκ τούτου, κρίθηκε ότι δεν προσφέρουν αξία στο μοντέλο και αφαιρέθηκαν από το σύνολο εκπαίδευσης.

Συνολικά, επιλέχθηκαν 27 χαρακτηριστικά και ετικέτες για την πρώτη φάση υλοποίησης της εκπαίδευσης του μοντέλου.

4 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ ΑΝΙΧΝΕΥΣΗΣ

Οι πλατφόρμες και τα εργαλεία που συντελούν στην διεκπεραίωση της εργασίας, σχετίζονται με τη γλώσσα προγραμματισμού Python και το οικοσύστημα βιβλιοθηκών της για τη μηχανική μάθηση και την επεξεργασία δεδομένων. Με μια πιο ενδελεχή ματιά, αρχικά ενσωματώθηκε η βιβλιοθήκη Pandas, η οποία αφορά την επεξεργασία και ανάλυση δεδομένων. Χρησιμοποιείται για τη φόρτωση, καθαρισμό, χειρισμό και ανάλυση δεδομένων, όπως στην περίπτωση μας, μορφής CSV. Επιπλέον, ενσωματώθηκε μια από τις πιο διαδεδομένες βιβλιοθήκες μηχανικής

μάθησης στην Python, η Scikit-learn. Η προεπεξεργασία δεδομένων, η δημιουργία των μοντέλων μηχανικής μάθησης, η αξιολόγηση της απόδοσης των μοντέλων αλλά και η υλοποίηση γνωστών αλγορίθμων, ανήκουν στις δυνατότητες που παρέχει η χρήση της βιβλιοθήκης. Τα σχετικά εργαλεία της βιβλιοθήκης αυτής που έχουν ενεργό ρόλο στην εργασία αυτή είναι δύο. Ο ‘OneHotEncoder’ που χρησιμοποιείται για την κωδικοποίηση κατηγορικών μεταβλητών σε δυαδικές (one-hot) μορφές και ο ‘SimpleImputer’ που χρησιμοποιείται για την αντιμετώπιση ελλειπόντων τιμών στα δεδομένα, αντικαθιστώντας τα με κάποια στρατηγική (π.χ., εδώ με τη μέση τιμή). Οι αλγόριθμοι που επιλέχθηκαν για την εκπαίδευση των μοντέλων έγινε με βάση την ικανότητα των αλγορίθμων να υποστηρίζουν την ιδιότητα “feature_importances” της Python.

Η ιδιότητα ‘feature_importances’ υποστηρίζεται κυρίως σε δέντρα απόφασης και σε σύνολα μεθόδων που βασίζονται σε αυτά. Παρέχει μια εκτίμηση της σημαντικότητας ή της συνεισφοράς κάθε χαρακτηριστικού στην τελική απόφαση του μοντέλου, επιστρέφοντας ένα πίνακα με αριθμούς που αντιστοιχούν στην σημασία κάθε χαρακτηριστικού. Τα χαρακτηριστικά με κανονικοποιημένους αριθμούς κοντά στο 1 είναι πιο κρίσιμα για την πρόβλεψη του μοντέλου, ενώ αυτά με πολύ μικρές τιμές (κοντά στο 0) μπορεί να είναι λιγότερο χρήσιμα ή ακόμη και άσχετα.

Η διαδικασία εύρεσης των κοινών σημαντικών χαρακτηριστικών από όλους τους αλγόριθμους, πραγματοποιείται μέσω της μεθόδου ‘.intersection()’. Στην Python, η intersection αναφέρεται στη διαδικασία εντοπισμού κοινών στοιχείων μεταξύ δύο ή περισσότερων συλλογών. Οι συλλογές αυτές μπορεί να είναι λίστες, σύνολα ή άλλες iterable δομές.

Ως απόρροια, οι επιλεχθέντες αλγόριθμοι αποτελούνται από τέσσερα δένδρα απόφασης. Ο ‘RandomTreeClassifier’ ο οποίος είναι ένας αλγόριθμος για την ταξινόμηση που χρησιμοποιεί πολλαπλά δένδρα απόφασης, ο ‘GradientBoostingClassifier’, αλγόριθμος που χρησιμοποιεί τη μέθοδο του gradient boosting για τη δημιουργία ενός ισχυρού μοντέλου από πολλαπλά αδύναμα μοντέλα. Ο τρίτος αλγόριθμος είναι ο ‘ExtraTreeClassifier’, μια παραλλαγή του Random Forest που επιτρέπει περισσότερη τυχαιότητα στην κατασκευή των δέντρων απόφασης και ο και τέταρτος αλγόριθμος είναι ο ‘DecisionTreeClassifier’, γνωστός βασικός αλγόριθμος δένδρου απόφασης για την ταξινόμηση.

Ως πλατφόρμα ανάπτυξης της εργασίας, χρησιμοποιείται το Jupiter Notebook. Είναι ιδιαίτερα γνωστό για τη διαδραστικότητά του αφού συνδυάζει την ανάπτυξη αναλύσεων δεδομένων και ζωντανό εκτελέσιμο κώδικα στο ίδιο περιβάλλον. Το περιβάλλον εκτέλεσης της εργασίας είναι το Google Colab.

5.1 ΕΚΠΑΙΔΕΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

Για το παρόν πείραμα, επιλέχθηκαν οι αλγόριθμοι Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier και Extra Trees Classifier. Αναλυτικότερα, στην πρώτη περίπτωση, ο αλγόριθμος στοχεύει σε ένα μοντέλο πρόβλεψης της τιμής της μεταβλητής - στόχου αρχίζοντας από το ριζικό κόμβο, με επιλογή ενός χαρακτηριστικού που λειτουργεί ως διαχωριστικό όριο στα δεδομένα. Με αναδρομική διαδικασία συμπληρώνονται όλοι οι κόμβοι με ομοιόμορφα κατανομημένες τιμές, μέχρι την πλήρωση των κριτηρίων τερματισμού του δένδρου. Στη δεύτερη περίπτωση, ο αλγόριθμος εκπαιδεύει πολλά δένδρα απόφασης σε τυχαία δείγματα από το σύνολο δεδομένων, με τη μέθοδο Bootstrap και λαμβάνει μια πρόβλεψη από το καθένα. Η τελική απόφαση, είναι η συνδυασμένη πρόβλεψη όλων των επιμέρους δένδρων. Στην επόμενη περίπτωση, με το Gradient Boosting ως τεχνική ταξινόμησης, η εκπαίδευση αρχίζει με την εκπαίδευση ενός «ασθενούς» δένδρου απόφασης και συνεχίζει με τη διόρθωση των επιμέρους σφαλμάτων στην εκπαίδευση του επόμενου δένδρου. Ως τελικό μοντέλο παρουσιάζεται ο προκύπτων συνδυασμός δένδρων. Ο τελευταίος επιλεγμένος αλγόριθμος, εκπαιδεύεται όμοια με τον Random Forest, ωστόσο διαφοροποιείται στο γεγονός ότι σε κάθε κόμβο, το διαχωριστικό χαρακτηριστικό επιλέγεται τυχαία και όχι σύμφωνα με τον καλύτερο πρακτικό διαχωρισμό, και η τιμή του διαχωρισμού είναι τυχαία.

Παρά τις διαφορές μεταξύ των αλγορίθμων, τα στάδια επεξεργασίας και εκπαίδευσής τους παραμένουν κοινά. Πραγματοποιείται αρχικά η προετοιμασία των δεδομένων. Τα δεδομένα διαχωρίζονται σε χαρακτηριστικά (X) και ετικέτες (y), με τα χαρακτηριστικά να αποτελούν τις εισόδους του μοντέλου και τις ετικέτες να αντιπροσωπεύουν την κατηγορία που πρέπει να προβλεφθεί. Για την ένταξη των αριθμητικών χαρακτηριστικών, απαιτείται η ένταξη της διαδικασίας αντιμετώπισης των απουσιαζουσών τιμών, όπου οι κενές τιμές αντικαθίστανται με

τον μέσο όρο των τιμών της στήλης, χρησιμοποιώντας τον μετασχηματιστή SimpleImputer. Παράλληλα, για τα κατηγορηματικά χαρακτηριστικά, χρησιμοποιείται η κωδικοποίηση σε δυαδικές μεταβλητές μέσω της τεχνικής One-Hot Encoding. Ακολουθεί η διαδικασία αφαίρεσης των «ανεπιθύμητων» χαρακτηριστικών με την μέθοδο `.drop()`. Η καθαυτού εκπαίδευση των μοντέλων πραγματοποιείται με τη χρήση της μεθόδου `.fit()`. Τα μοντέλα εκπαιδεύονται πάνω στο σύνολο δεδομένων των επεξεργασμένων πλέον αριθμητικών και κατηγορηματικών χαρακτηριστικών.

5.2 ΠΕΡΙΓΡΑΦΗ ΠΕΙΡΑΜΑΤΩΝ

Στόχος του πειράματος ήταν να διερευνηθεί αν τα χαρακτηριστικά των ροών δικτύου μπορούν να αξιοποιηθούν μέσω της εκπαίδευσης αλγορίθμων μηχανικής μάθησης, για να εντοπιστούν κορυφαία κοινά χαρακτηριστικά που έχουν καθοριστική σημασία στη διάκριση μεταξύ κακόβουλης επίθεσης και κανονικής δραστηριότητας στο δίκτυο. Αρχικά, εκπαιδεύτηκαν οι τέσσερις διαφορετικοί αλγόριθμοι και πραγματοποιήθηκε σύγκριση των χαρακτηριστικών που αναγνωρίζονταν ως τα πιο σημαντικά από καθέναν από αυτούς. Ως τελικό αποτέλεσμα, παίρνουμε τα κορυφαία χαρακτηριστικά που είναι κοινά για όλους τους αλγορίθμους. Το χαρακτηριστικό λοιπόν που αναδείχθηκε κοινώς ως το πιο σημαντικό, είναι το "dst_host_srv_count". Στη συνέχεια, οι αλγόριθμοι συνδυάστηκαν ανά ζεύγη και ανά τριάδες, όπου και υπήρξαν ενδιαφέροντα αποτελέσματα. Βρέθηκαν κάποια κοινά χαρακτηριστικά με μεγαλύτερη συχνότητα. Τα χαρακτηριστικά "src_bytes", "dst_bytes" και "dst_host_srv_count" αναδείχθηκαν ως τα πιο συχνά σημαντικά χαρακτηριστικά στις περισσότερες από αυτές τις περιπτώσεις.

Για να επεκταθούν και να ελεγχθούν τα αποτελέσματα των χαρακτηριστικών που αναδείχθηκαν από τα αρχικά μοντέλα, υλοποιήθηκε ένα επιπλέον μοντέλο. Σε αυτή την προσέγγιση, οι ίδιοι αλγόριθμοι μηχανικής μάθησης που χρησιμοποιήθηκαν αρχικά εκπαιδεύτηκαν ξανά, αυτή τη φορά χρησιμοποιώντας μόνο τα τρία χαρακτηριστικά που

αναγνωρίστηκαν ως τα πλέον σημαντικά: τα "src_bytes", "dst_bytes" και "dst_host_srv_count". Ο σκοπός αυτής της διαδικασίας ήταν διττός.

Πρώτον, εξετάστηκε αν αυτά τα χαρακτηριστικά είναι επαρκή για την ορθή πρόβλεψη κακόβουλων επιθέσεων στο δίκτυο, μειώνοντας ταυτόχρονα την πολυπλοκότητα του μοντέλου. Αυτός ο έλεγχος επιτρέπει την αξιολόγηση του κατά πόσο η χρήση μόνο τριών χαρακτηριστικών μπορεί να προσφέρει εξίσου αποτελεσματική διάκριση μεταξύ κανονικής και κακόβουλης δραστηριότητας, συγκριτικά με τα αποτελέσματα της αρχικής εκπαίδευσης που περιλάμβανε 27 χαρακτηριστικά.

Δεύτερον, εξετάστηκε ο χρόνος απόκρισης των αποτελεσμάτων, συγκρίνοντας την εκπαίδευση των μοντέλων των 27 χαρακτηριστικών έναντι της εκπαίδευσης των τριών κοινών χαρακτηριστικών. Αυτή η σύγκριση αξιολογεί την πιθανότητα ταχύτερης απόκρισης του συστήματος, διατηρώντας ταυτόχρονα ισορροπία ακρίβειας και ευαισθησίας στην ανίχνευση των επιθέσεων. Η μείωση της πολυπλοκότητας και του χρόνου απόκρισης είναι κρίσιμης σημασίας για εφαρμογές σε πραγματικό χρόνο, όπου η ταχύτητα και η αποδοτικότητα είναι απαραίτητες για την αποτελεσματική ανίχνευση και αντιμετώπιση των κακόβουλων επιθέσεων.

5.3 ΠΑΡΑΜΕΤΡΟΙ ΕΚΠΑΙΔΕΥΣΗΣ

Κάθε μοντέλο μηχανικής μάθησης, αξιοποιεί ένα σύνολο προεπιλεγμένων παραμέτρων στη βιβλιοθήκη scikit-learn. Αυτές οι παράμετροι ελέγχουν διάφορες πτυχές της εκπαίδευσης του μοντέλου, όπως το βάθος των δέντρων, τον αριθμό των εκτιμητών στα σύνολα, τους ρυθμούς μάθησης και άλλα. Είναι σημαντικό να κατανοηθεί η βαρύτητα των παραμέτρων, καθώς μερικές δύναται να επηρεάσουν τα αποτελέσματα των πειραμάτων μας, ενώ άλλες να παραμένουν με ουδέτερη βαρύτητα στις αλλαγές.

Οι default παράμετροι που χρησιμοποιούνται από κάθε αλγόριθμο στη βιβλιοθήκη scikit-learn, βρίσκονται αρχικοποιώντας το αντικείμενο του ταξινομητή και εκτυπώνοντας τις παραμέτρους του με τη μέθοδο .get_params(). Με τη μέθοδο αυτή εμφανίζονται όλες οι παράμετροι και οι προεπιλεγμένες τιμές τους. Οι προεπιλεγμένες παράμετροι έχουν ρυθμιστεί ώστε να θεωρούνται γενικά καλές για πολλά διαφορετικά προβλήματα, αλλά δεν είναι πάντα οι

βέλτιστες για όλα τα σύνολα δεδομένων. Οι τιμές τους, μπορεί να διαφέρουν ανάλογα με την έκδοση του scikit-learn που χρησιμοποιείται.

Προεπιλεγμένες παράμετροι κάθε αλγορίθμου:

Decision Tree Classifier:

- criterion='gini'
- max_depth=None
- min_samples_split=2
- min_samples_leaf=1
- min_weight_fraction_leaf=0.0
- max_features=None
- random_state=None
- max_leaf_nodes=None
- min_impurity_decrease=0.0
- class_weight=None
- ccp_alpha=0.0

Random Forest Classifier:

- n_estimators=100
- criterion='gini'
- max_depth=None
- min_samples_split=2
- min_samples_leaf=1
- min_weight_fraction_leaf=0.0
- max_features='sqrt'
- max_leaf_nodes=None
- bootstrap=True
- oob_score=False
- n_jobs=None
- random_state=None
- verbose=0
- warm_start=False

- class_weight=None
- ccp_alpha=0.0

Gradient Boosting Classifier:

- loss='deviance'
- learning_rate=0.1
- n_estimators=100
- subsample=1.0
- criterion='friedman_mse'
- max_depth=3
- min_samples_split=2
- min_samples_leaf=1
- min_weight_fraction_leaf=0.0
- max_features=None
- random_state=None
- max_leaf_nodes=None
- validation_fraction=None
- n_iter_no_change=None
- tol=0.0001
- ccp_alpha=0.0

Extra Trees Classifier:

- n_estimators=100
- criterion='gini'
- max_depth=None
- min_samples_split=2
- min_samples_leaf=1
- min_weight_fraction_leaf=0.0
- max_features='sqrt'
- max_leaf_nodes=None
- bootstrap=False
- oob_score=False

- `n_jobs=None`
- `random_state=None`
- `verbose=0`
- `warm_start=False`
- `class_weight=None`
- `ccp_alpha=0.0`

Εξετάστηκε λοιπόν η επίδραση διαφορετικών παραμέτρων στους αλγόριθμους ταξινόμησης ως προς τη βαρύτητα στη σημασία των χαρακτηριστικών. Κάθε αλγόριθμος δοκιμάστηκε με τυχαίες διαφορετικές ρυθμίσεις παραμέτρων για να κατανοηθεί η επιρροή τους στα αποτελέσματα της ταξινόμησης.

Αλγόριθμος	1ος Κύκλος Αλλαγών	2ος Κύκλος Αλλαγών	3ος Κύκλος Αλλαγών
Decision Tree Classifier	<code>max_depth=5</code>	<code>min_samples_split=5</code> , <code>min_samples_leaf=5</code>	<code>max_depth=7</code> , <code>criterion='entropy'</code>
Random Forest Classifier	<code>n_estimators=200</code> , <code>max_depth=10</code>	<code>min_samples_split=5</code> , <code>min_samples_leaf=5</code>	<code>n_estimators=150</code> , <code>min_samples_split=10</code>
Gradient Boosting Classifier	<code>learning_rate=0.05</code> , <code>n_estimators=200</code>	<code>subsample=0.8</code> , <code>max_features=0.5</code>	<code>learning_rate=0.1</code> , <code>n_estimators=100</code> , <code>max_depth=3</code>
Extra Trees Classifier	<code>n_estimators=200</code> , <code>max_features='log2'</code>	<code>min_samples_split=5</code> , <code>min_samples_leaf=5</code>	<code>n_estimators=100</code> , <code>max_features='sqrt'</code> , <code>min_samples_leaf=10</code>

Εικόνα 1: Αλλαγές Παραμέτρων

Οι κύκλοι πειραμάτων πραγματοποιήθηκαν με τυχαία επιλεγμένη τροποποίηση των παραμέτρων σε κάθε φάση. Τα αποτελέσματα από κάθε κύκλο συγκρίθηκαν με τα χαρακτηριστικά που είχαν εντοπιστεί χρησιμοποιώντας τις αρχικές, προεπιλεγμένες παραμέτρους. Αυτή η σύγκριση επέτρεψε την ανάλυση του αντίκτυπου που είχαν οι αλλαγές στις παραμέτρους στην ακρίβεια και την απόδοση του μοντέλου.

Αποτελέσματα αλλαγών:

Η διαφοράς μεταξύ των αρχικών και τροποποιημένων παραμέτρων έδειξε ότι στον πρώτο κύκλο αλλαγών, τα κοινά χαρακτηριστικά παρέμειναν τα ίδια, υποδεικνύοντας ότι οι συγκεκριμένες αλλαγές δεν είχαν σημαντική επίδραση στην κατάταξη των χαρακτηριστικών. Η σημασία δηλαδή των χαρακτηριστικών δεν επηρεάστηκε από τις αλλαγές των τιμών τους. Αντίθετα, στον δεύτερο και τρίτο κύκλο δοκιμών, τα κοινά χαρακτηριστικά άλλαξαν καθώς τροποποιήθηκαν οι παράμετροι. Επομένως, φαίνεται η άμεση επιρροή των παραμέτρων στη βαρύτητα με την οποία οι αλγόριθμοι αξιολογούν τη σημασία των χαρακτηριστικών. Αυτό σημαίνει ότι οι αλλαγές οδηγούν σε διαφορετικές εκτιμήσεις σχετικά με το ποια είναι τα κρίσιμα χαρακτηριστικά, επηρεάζοντας έτσι την ικανότητα των μοντέλων να αναγνωρίζουν κοινούς παράγοντες που επηρεάζουν την απόδοση. Τα χαρακτηριστικά που επηρεάζονται ενδέχεται να αντικατοπτρίζουν επίσης και τη διαφορετική στρατηγική της επιλογής χαρακτηριστικών από τους αλγορίθμους.

Έλεγχος κάθε αλγορίθμου ξεχωριστά:

Για να σχηματιστεί μια πιο σαφής εικόνα για την επίδραση των αλλαγών στις παραμέτρους των αλγορίθμων στην απόδοση του μοντέλου, πραγματοποιήθηκε μια ακόμη δοκιμασία κατά την οποία για κάθε αλγόριθμο ταξινόμησης ξεχωριστά εξετάστηκαν τα κορυφαία χαρακτηριστικά του, με τροποποιήσεις σε τρεις τυχαία επιλεγμένες παραμέτρους: τις `min_samples_split`, `min_samples_leaf` και `max_depth`. Τα αποτελέσματα της δοκιμασίας αυτής έδειξαν ότι το χαρακτηριστικό `dst_host_srv_count` εμφανίζεται αρκετά σταθερά σε όλους τους αλγόριθμους, ενώ τα χαρακτηριστικά `src_bytes` και `dst_bytes` επανεμφανίζονται συχνά, αν και λιγότερο σταθερά σε σχέση με το πρώτο.

```
Classifier: Decision Tree Classifier
Parameters: {'min_samples_split': 2, 'min_samples_leaf': 1}
Top Features: ['src_bytes', 'service_ecr_i', 'dst_host_srv_count', 'hot', 'dst_bytes']
```

```
Classifier: Decision Tree Classifier
Parameters: {'min_samples_split': 5, 'min_samples_leaf': 5}
Top Features: ['src_bytes', 'service_ecr_i', 'dst_host_srv_count', 'hot', 'dst_bytes']
```

```
Classifier: Decision Tree Classifier
Parameters: {'max_depth': None}
Top Features: ['src_bytes', 'service_ecr_i', 'dst_host_srv_count', 'hot', 'dst_bytes']
```

```
Classifier: Decision Tree Classifier
Parameters: {'max_depth': 7}
Top Features: ['src_bytes', 'service_ecr_i', 'dst_host_srv_count', 'hot', 'dst_bytes']
```

```
Classifier: Random Forest Classifier
Parameters: {'min_samples_split': 2, 'min_samples_leaf': 1}
Top Features: ['dst_bytes', 'src_bytes', 'dst_host_srv_count', 'dst_host_diff_srv_rate', 'same_srv_rate']
```

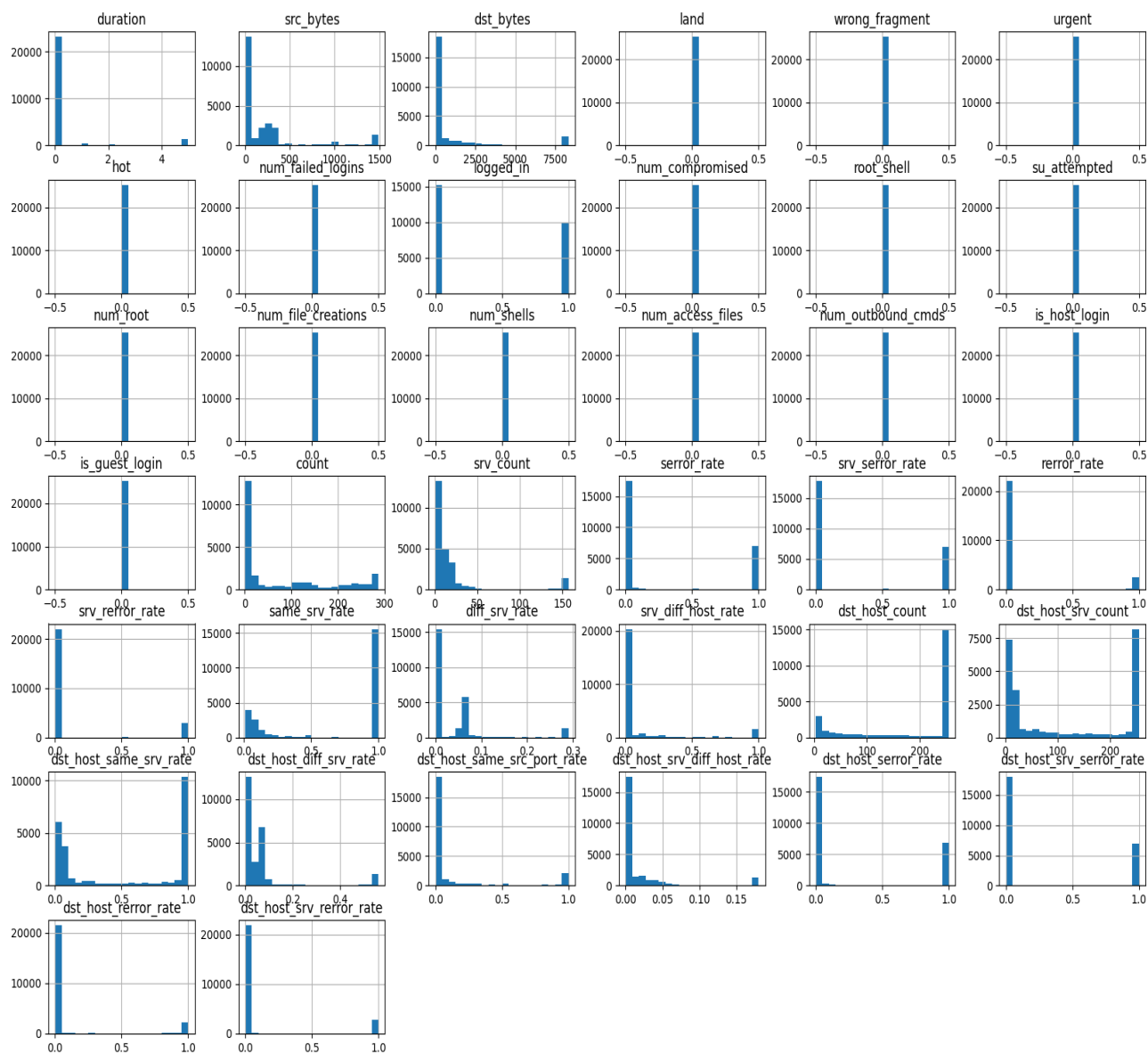
```
Classifier: Random Forest Classifier
Parameters: {'min_samples_split': 5, 'min_samples_leaf': 5}
Top Features: ['src_bytes', 'dst_bytes', 'dst_host_srv_count', 'diff_srv_rate', 'same_srv_rate']
```

Εικόνα 2: Δοκιμασία αλλαγών παραμέτρων

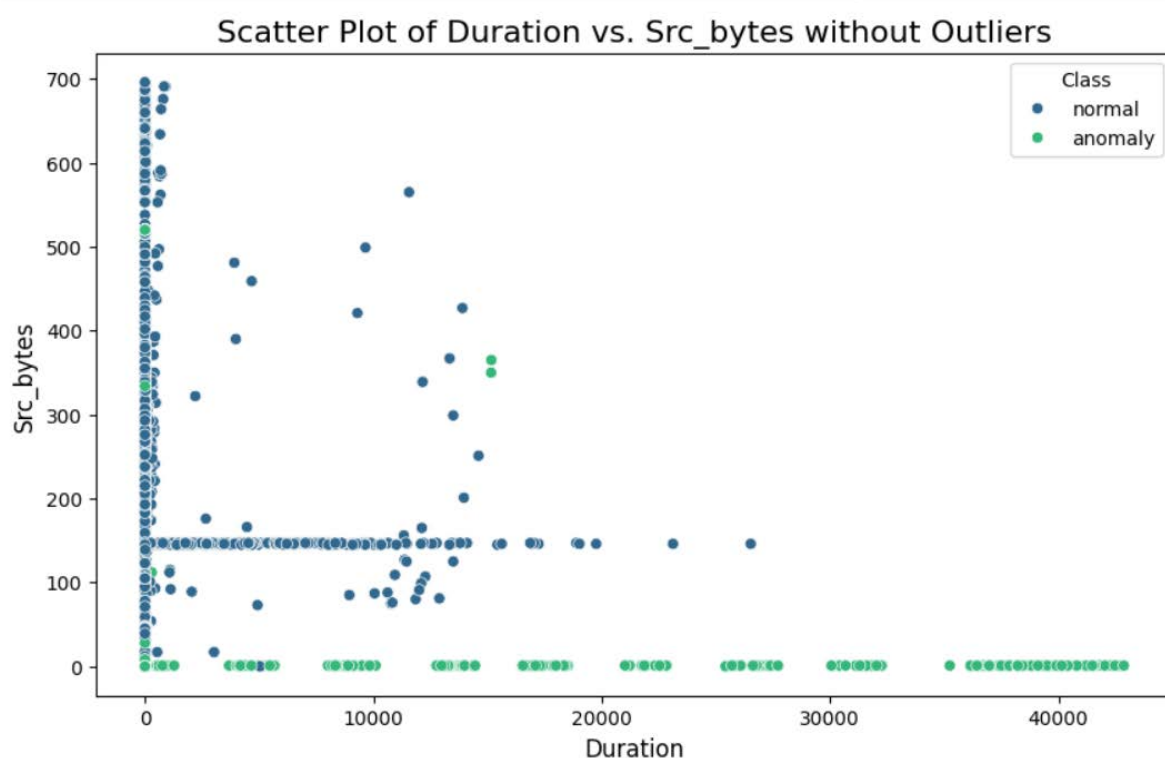
Ο Decision Tree Classifier και ο Gradient Boosting Classifier εμφάνισαν περιορισμένες αλλαγές στη λίστα των κορυφαίων χαρακτηριστικών, ανεξαρτήτως των παραμετροποιήσεων. Επομένως, είναι πιθανό τα μοντέλα αυτά να στηρίζονται σε συγκεκριμένα χαρακτηριστικά και οι μεταβολές των παραμέτρων να μην επηρεάζουν ουσιαστικά την αξιολόγηση της σημασίας αυτών των χαρακτηριστικών. Αντίθετα, οι αλγόριθμοι Random Forest και Extra Trees Classifier παρουσίασαν μεγαλύτερες διαφοροποιήσεις στη λίστα των κορυφαίων χαρακτηριστικών. Πιθανότατα αυτοί οι αλγόριθμοι να είναι πιο ευαίσθητοι στις παραμετροποιήσεις και να εξετάζουν ευρύτερο φάσμα χαρακτηριστικών όταν αλλάζουν οι παράμετροι. Οι διαφοροποιήσεις αυτές ενισχύουν την ευελιξία των συγκεκριμένων αλγορίθμων, επιτρέποντάς τους να αξιοποιούν εύρος δεδομένων ανάλογα με τις ρυθμίσεις.

6.1 ΠΑΡΑΤΗΡΗΣΗΣ ΠΡΟΕΠΕΞΕΡΓΑΣΙΑΣ ΔΕΔΟΜΕΝΩΝ

Histograms of Features (Outliers Clipped)



Εικόνα 3: Ιστογράμματα ανάλυσης δεδομένων

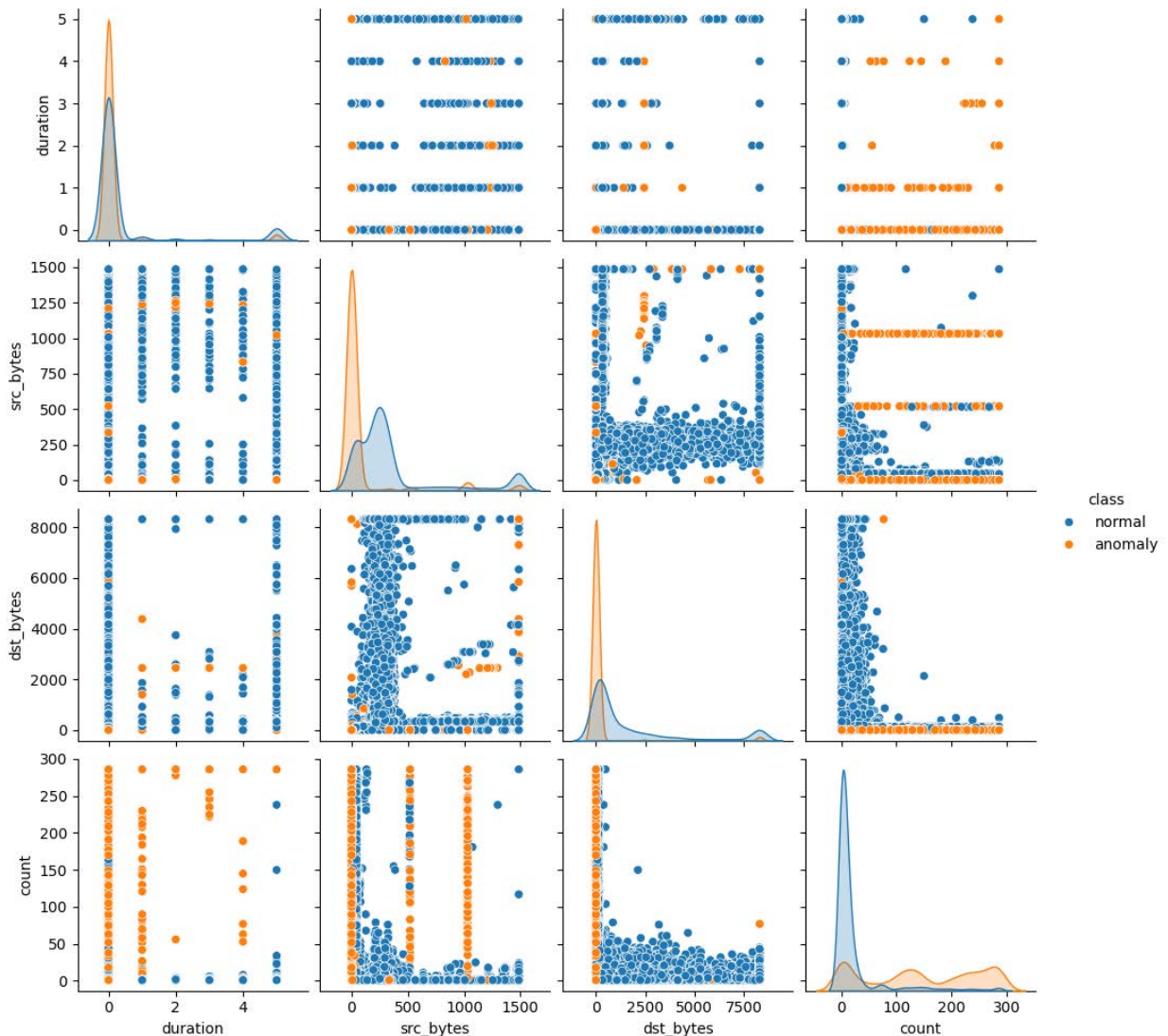


Εικόνα 4: Scatter plot ανάλυσης δεδομένων

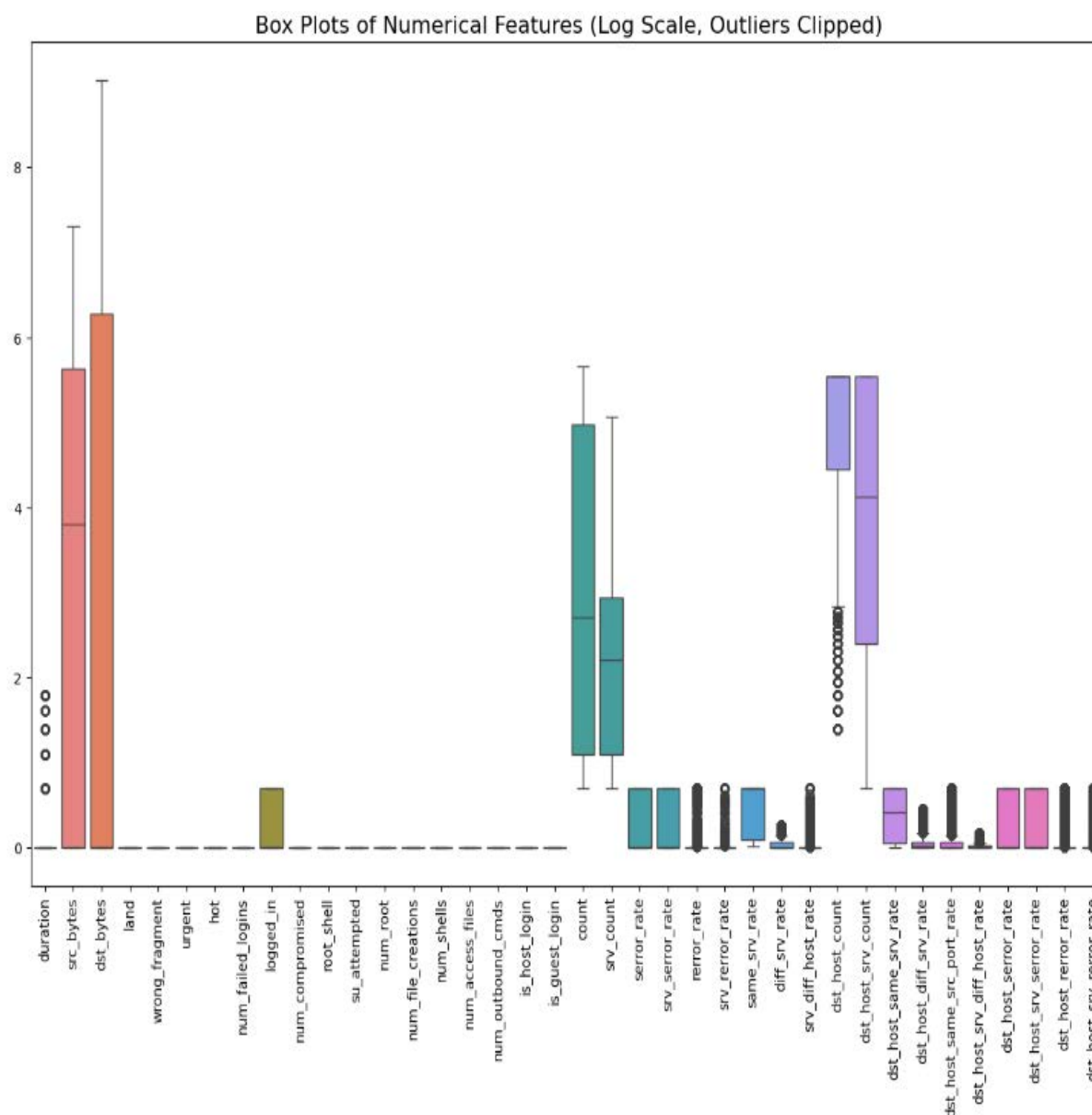
Η επιλογή για τη δημιουργία ενός scatter plot με τα χαρακτηριστικά 'src_bytes' στον άξονα x και 'duration' στον άξονα y βασίστηκε στη φύση και τη σημασία των δεδομένων αυτών. Πιο αναλυτικά:

- Χαρακτηριστικό 'src_bytes': Αυτό το χαρακτηριστικό αναφέρεται στον αριθμό των bytes που αποστέλλονται από την πηγή κατά τη διάρκεια μιας σύνδεσης. Η ανάλυσή του μπορεί να παρέχει σημαντικές πληροφορίες σχετικά με τον όγκο της κίνησης δεδομένων, καθώς και τη σημασία της συγκεκριμένης σύνδεσης στο πλαίσιο της κατηγοριοποίησης.
- Χαρακτηριστικό 'duration': Αυτό το χαρακτηριστικό αναφέρεται στη χρονική διάρκεια κάθε σύνδεσης ή επικοινωνίας. Η κατανόηση της διάρκειας είναι κρίσιμη για την ανάλυση της επιρροής του χρόνου στην κατηγοριοποίηση των δεδομένων.

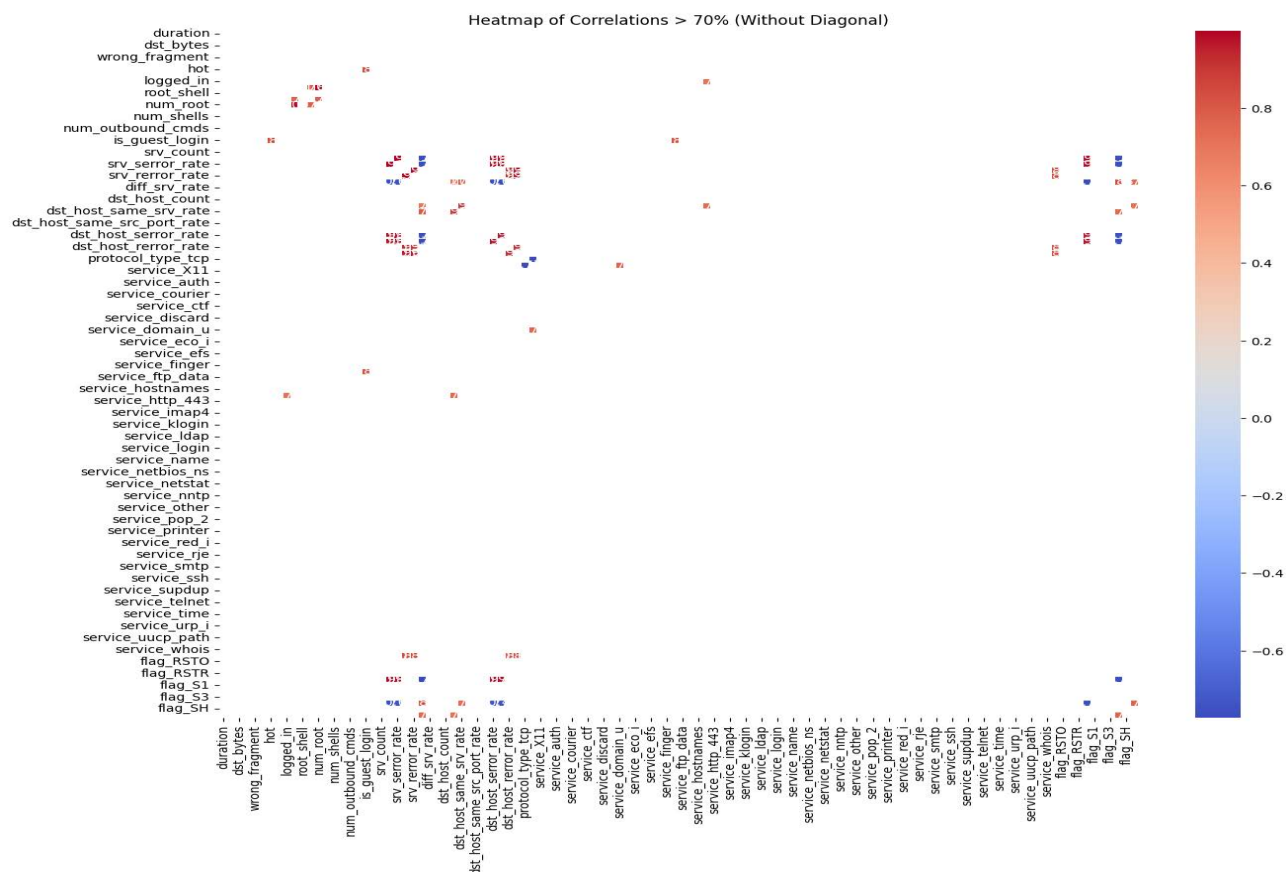
Από την ανάλυση των δεδομένων παρατηρείται ότι η πλειοψηφία των επιθέσεων χαρακτηρίζονται από μεγάλη διάρκεια και μικρό αριθμό bytes που αποστέλλονται από την πηγή. Αντίθετα, οι φυσιολογικές ροές τείνουν να έχουν μεγαλύτερο αριθμό bytes που αποστέλλονται από την πηγή και μικρότερη διάρκεια. Αυτή η παρατήρηση υποστηρίζεται επίσης από τα ιστογράμματα, τα box plots και τα pair plots, τα οποία αποκαλύπτουν με σαφήνεια τη σημαντικότητα των κοινών χαρακτηριστικών για την ανάλυση των ροών δεδομένων. Ακόμα και πριν από την εκτέλεση των πρακτικών πειραμάτων, αυτά τα γραφήματα δείχνουν την αναγνωρίσιμη σημασία των συγκεκριμένων χαρακτηριστικών στην κατηγοριοποίηση των επιθέσεων και των φυσιολογικών ροών.



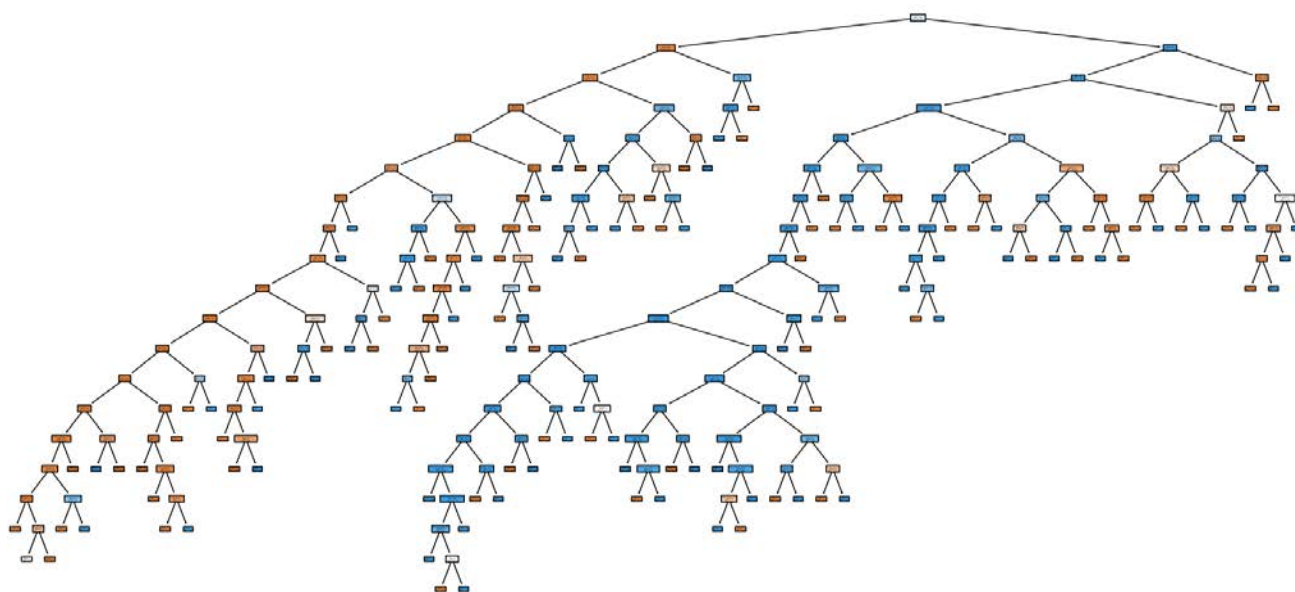
Εικόνα 5: Pair plots ανάλυσης δεδομένων



Εικόνα 6: Box plots ανάλυσης δεδομένων



Εικόνα 7: Υψηλές συσχετίσεις χαρακτηριστικών μέσω heatmap



Εικόνα 8: Οπτικοποίηση εκπαίδευσης του Decision Tree

6.2 ΣΥΓΚΡΙΤΙΚΗ ΑΝΑΛΥΣΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΜΕΤΑΞΥ ΑΛΓΟΡΙΘΜΩΝ

Στο πρώτο στάδιο της έρευνας, μελετήθηκε η σημασία των χαρακτηριστικών μεταξύ διαφορετικών αλγορίθμων ταξινόμησης, καθώς και η επίδραση των αλλαγών στις παραμέτρους τους. Τα αποτελέσματα της ανάλυσης, έπειτα από την προεπεξεργασία των δεδομένων για αποφυγή του bias, έδειξαν ότι υπάρχει ένα κοινώς κορυφαίο χαρακτηριστικά μεταξύ όλων των μοντέλων που εξετάστηκαν, το "dst_host_srv_count", το οποίο αναφέρεται στον αριθμό συνδέσεων προς έναν συγκεκριμένο προορισμό (host). Αυτό το γεγονός υποδεικνύει ότι όλα τα μοντέλα συγκλίνουν απόλυτα στη σημαντικότητα του χαρακτηριστικού, ανεξάρτητα από τις διαφορές στις μεθόδους ανάλυσης που εφαρμόζονται σε κάθε αλγόριθμος. Η περαιτέρω ανάλυση που έγινε, έδειξε ότι συγκρίνοντας όλα τα μοντέλα ανά ζεύγη μεταξύ τους, καταλήγουμε σε περισσότερα αποτελέσματα σχετικά με τη σημαντικότητα και άλλων χαρακτηριστικών. Πράγματι, το ίδιο συνέβη και με την περαιτέρω διερεύνηση της συσχέτισης των χαρακτηριστικών με δοκιμή όλων των πιθανών συνδυασμών από τριάδες των αλγορίθμων για να ελέγξουμε τα κοινά τους χαρακτηριστικά. Αυτή η διαδικασία επέτρεψε την πιο ολοκληρωμένη κατανόηση του τρόπου με τον οποίο διάφοροι αλγόριθμοι αξιολογούν τη σημασία των χαρακτηριστικών και εντοπίστηκαν τα πιο συνήθη εξαγόμενα χαρακτηριστικά.

Common top features across models: ['dst_host_srv_count']

Εικόνα 9: Κοινά κορυφαία χαρακτηριστικά όλων των αλγορίθμων μαζί

6.3 ΣΥΜΠΕΡΑΣΜΑΤΑ ΣΥΓΚΡΙΣΗΣ ΑΝΑ ΖΕΥΓΗ ΚΑΙ ΑΝΑ ΤΡΙΑΔΑ

Η ανάλυση των αποτελεσμάτων από τις συγκρίσεις ανά ζεύγη και ανά τριάδα αλγορίθμων αποκάλυψε σημαντικές παρατηρήσεις. Κοινά χαρακτηριστικά μπορεί να υπάρχουν μεταξύ διαφορετικών αλγορίθμων, ωστόσο η συχνότητα εμφάνισής τους μπορεί να διαφέρει ανάλογα με τον αριθμό των αλγορίθμων που συγκρίνονται. Ορισμένα χαρακτηριστικά, δηλαδή τα dst_bytes, src_bytes και dst_host_srv_count εμφανίζονται συχνά ως σημαντικά σε πολλούς αλγόριθμους. Αυτό υποδεικνύει ότι τα χαρακτηριστικά αυτά έχουν σημαντικό ρόλο στην

διάκριση μεταξύ των κατηγοριών ("normal" και "anomaly"). Κάθε αλγόριθμος έχει τις δικές του μεθόδους για τον υπολογισμό της σημασίας των χαρακτηριστικών και την εύρεση του βέλτιστου συνδυασμού χαρακτηριστικών για το συγκεκριμένο πρόβλημα. Το γεγονός όμως ότι χαρακτηριστικά 'dst_bytes', 'src_bytes' και 'dst_host_srv_count' εμφανίζονται συνέχεια ως σημαντικά υποδηλώνει την ευαισθησία τους στη διαφοροποίηση των δεδομένων. Αυτά τα χαρακτηριστικά παρέχουν κρίσιμη πληροφορία για την πρόβλεψη και θεωρούνται αξιόπιστα για την κατανόηση των δεδομένων και την ανίχνευση ανωμαλιών.

```
Common top features between Decision Tree Classifier and Random Forest Classifier: ['src_bytes', 'dst_bytes', 'dst_host_srv_count']
Common top features between Decision Tree Classifier and Gradient Boosting Classifier: ['src_bytes', 'service_ecr_i', 'dst_bytes', 'dst_host_srv_count']
Common top features between Decision Tree Classifier and Extra Trees Classifier: ['dst_host_srv_count']
Common top features between Random Forest Classifier and Gradient Boosting Classifier: ['src_bytes', 'dst_bytes', 'dst_host_srv_count']
Common top features between Random Forest Classifier and Extra Trees Classifier: ['same_srv_rate', 'dst_host_srv_count']
Common top features between Gradient Boosting Classifier and Extra Trees Classifier: ['dst_host_srv_count']
Model training and feature importance analysis are complete.
```

Εικόνα 10: Κοινά κορυφαία χαρακτηριστικά αλγορίθμων ανά ζεύγη

```
Common top features between Decision Tree Classifier, Random Forest Classifier and Gradient Boosting Classifier: ['dst_host_srv_count', 'src_bytes', 'dst_bytes']
Common top features between Decision Tree Classifier, Random Forest Classifier and Extra Trees Classifier: ['dst_host_srv_count']
Common top features between Decision Tree Classifier, Gradient Boosting Classifier and Extra Trees Classifier: ['dst_host_srv_count']
Common top features between Random Forest Classifier, Gradient Boosting Classifier and Extra Trees Classifier: ['dst_host_srv_count']
Model training and feature importance analysis are complete.
```

Εικόνα 11: Κοινά κορυφαία χαρακτηριστικά αλγορίθμων ανά τριάδες

6.4 ΕΡΜΗΝΕΥΣΙΜΟΤΗΤΑ

Η αναγνώριση σημαντικών χαρακτηριστικών ενισχύει την ερμηνευσιμότητα του μοντέλου. Τα χαρακτηριστικά 'src_bytes', 'dst_bytes' και 'dst_host_srv_count' που σχετίζονται με τον όγκο των δεδομένων που αποστέλλονται και λαμβάνονται, αλλά και τον αριθμό συνδέσεων προς έναν συγκεκριμένο προορισμό, μπορούν να αποτελέσουν ενδείξεις για επιθέσεις δικτύου ή ανωμαλίες. Είναι σημαντικές ενδείξεις της κατευθυνόμενης κίνησης σε ένα δίκτυο. Η αντίληψη της επιρροής αυτών των χαρακτηριστικών μπορεί να βελτιώσει την ερμηνεία των προβλέψεων των μοντέλων και την κατανόηση των μηχανισμών των επιθέσεων. Πιο συγκεκριμένα, η συχνή εμφάνιση του χαρακτηριστικού src_bytes μπορεί να υποδεικνύει ανωμαλίες δικτύου που σχετίζονται με μεγάλες ποσότητες δεδομένων που αποστέλλονται από την πηγή. Αυτό μπορεί να βοηθήσει στην αναγνώριση επιθέσεων όπως το Denial of Service (DoS), όπου μεγάλες ποσότητες δεδομένων αποστέλλονται για να υπερφορτωθεί το σύστημα. Ομοίως, η παρακολούθηση του dst_host_srv_count μπορεί να προσφέρει χρήσιμες πληροφορίες για τον ίδιο τύπο επιθέσεων, καθώς όσο περισσότερες είναι οι συνδέσεις προς έναν προορισμό, τόσο πιο πιθανό είναι αυτός ο προορισμός να αποτελεί στόχο ή να είναι κεντρικό σημείο σε μια πιθανή επίθεση. Αντίστοιχα, το χαρακτηριστικό των dst_bytes εντοπίζεται ως σημαντικό και υποδηλώνοντας ότι οι ανωμαλίες σχετίζονται με μεγάλες ποσότητες δεδομένων που λαμβάνονται από τον προορισμό. Αυτό μπορεί να βοηθήσει στην ανίχνευση ανωμαλιών όπως η κλοπή δεδομένων, όπου μεγάλοι όγκοι δεδομένων μεταφέρονται από τον προορισμό. Η ανάλυση της σημασίας των χαρακτηριστικών μέσω συγκρίσεων μεταξύ σχετικών αλγορίθμων καταδεικνύει τη σημασία της προσεκτικής επιλογής τους, όπως και της επιλογής των παραμέτρων εκπαίδευσης για την ενίσχυση της απόδοσης των μοντέλων. Τα αποτελέσματα υποδεικνύουν ότι η αναγνώριση κοινών σημαντικών χαρακτηριστικών και η κατανόηση του ρόλου τους μπορεί να βελτιώσει τη διαγνωστική ακρίβεια των μοντέλων και να παρέχει πιο αξιόπιστα και ερμηνεύσιμα αποτελέσματα.

6.5 ΑΠΟΔΟΣΗ ΜΟΝΤΕΛΟΥ ΤΩΝ ΤΡΙΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ

Το νέο μοντέλο που προτάθηκε, στηρίζει την εκπαίδευσή του μόνο στα τρία κορυφαία κοινά χαρακτηριστικά που προτάθηκαν (δηλαδή τα src_bytes, dst_bytes και dst_host_srv_count). Το μοντέλο καταφέρνει να διαχωρίσει με ικανοποιητική ακρίβεια, της τάξεως του 97 – 98.9% και

f1 score της τάξεως 96-98% σε κάθε αλγόριθμο, τις κανονικές ροές από τις ανωμαλίες του συστήματος. Ειδικότερα, η μέση ακρίβεια του Decision Tree Classifier βρίσκεται στο 0.9896, και το f1 score βρίσκεται στο 0.9888, του Random Forest Classifier η ακρίβεια στο 0.9880 και το f1 score στο 0.9871, του Gradient Boosting Classifier η ακρίβεια στο 0.9877 και το f1 score στο 0.9868 και του Extra Trees Classifier η ακρίβεια στο 0.9704 και το f1 score στο 0.9683.

```
Decision Tree Classifier:
Average Validation Accuracy: 0.9900
Average Test Accuracy: 0.9896
Average Test F1 Score: 0.9888

Random Forest Classifier:
Average Validation Accuracy: 0.9884
Average Test Accuracy: 0.9880
Average Test F1 Score: 0.9871

Gradient Boosting Classifier:
Average Validation Accuracy: 0.9880
Average Test Accuracy: 0.9877
Average Test F1 Score: 0.9868

Extra Trees Classifier:
Average Validation Accuracy: 0.9715
Average Test Accuracy: 0.9704
Average Test F1 Score: 0.9683
```

Εικόνα 12: Ακρίβεια και f1 score

6.6 ΣΥΓΚΡΙΣΗ ΤΩΝ ΔΥΟ ΜΟΝΤΕΛΩΝ ΕΚΠΑΙΔΕΥΣΗΣ

Οι δύο διαφορετικές προσεγγίσεις εκπαίδευσης των αλγορίθμων μηχανικής μάθησης για την ανίχνευση επιθέσεων δικτύων, συγκρίθηκαν μεταξύ τους με μετρικές χρόνου και απόδοσης. Η βασική τους διαφοροποίηση φυσικά, έγκειται στον αριθμό των χαρακτηριστικών παραμέτρων που λήφθηκαν υπόψη κατά την εκπαίδευση. Η πρώτη προσέγγιση εκπαιδεύει τους αλγορίθμους χρησιμοποιώντας 27 από τα 41 τα διαθέσιμα χαρακτηριστικά του συνόλου δεδομένων, μια διαδικασία που περιλαμβάνει την επεξεργασία τόσο αριθμητικών όσο και κατηγορικών χαρακτηριστικών. Γίνεται αυτόματη επεξεργασία των αριθμητικών δεδομένων, κωδικοποίηση των κατηγορικών χαρακτηριστικών, καθώς και αφαίρεση ορισμένων «προβληματικών» χαρακτηριστικών για τη δημιουργία του τελικού πίνακα χαρακτηριστικών. Στη δεύτερη προσέγγιση, οι αλγόριθμοι εκπαιδεύονται μόνο με τα τρία επιλεγμένα χαρακτηριστικά, βασιζόμενοι στην παρατήρηση ότι αυτά τα χαρακτηριστικά εντοπίζονται συχνά ως σημαντικά σε μοντέλα που εκπαιδεύτηκαν πρωτίστως με όλα τα δυνατά χαρακτηριστικά. Η εκπαίδευση

των αλγορίθμων μόνο με τα τρία χαρακτηριστικά μειώνει σημαντικά την πολυπλοκότητα και τον χρόνο εκπαίδευσης. Αυτό επιτυγχάνεται λόγω του μικρού αριθμού των χαρακτηριστικών και την απλούστευση της προεπεξεργασίας των δεδομένων. Οι αλγόριθμοι δηλαδή, δεν χρειάζεται να επεξεργαστούν, να προεπεξεργαστούν ή να αναλύσουν όλα τα υπόλοιπα χαρακτηριστικά του συνόλου δεδομένων.

Εστιάζοντας στην ακρίβεια των αλγορίθμων, παρατηρούμε ότι το πρώτο μοντέλο μεγαλύτερης πολυπλοκότητας έχει ακρίβεια περίπου 99.7%, ενώ στο δεύτερο μοντέλο η ακρίβεια μειώνεται και φτάνει μέχρι περίπου 98.9%.

Εστιάζοντας στη χρονική μετρική της σύγκρισης των δύο προσεγγίσεων, ο χρόνος εκπαίδευσης στην δεύτερη περίπτωση είναι σημαντικά μειωμένος, δεδομένου του απαραίτητου μόνο πλήθους των χαρακτηριστικών, που δεν χρειάζονται επεξεργασία και ανάλυση. Αυτή η μείωση του χρόνου είναι ιδανική για περιπτώσεις όπου η ταχύτητα ανάπτυξης και εφαρμογής ενός μοντέλου ανίχνευσης ανωμαλιών είναι ζωτικής σημασίας.

Ωστόσο, αναφορικά με την ανιχνευτική ικανότητα των δύο μοντέλων, ενδεχομένως, να μην καλύπτονται όλες οι πιθανές περιπτώσεις επιθέσεων, ιδίως εάν άλλοι τύποι επιθέσεων επηρεάζουν στοχευμένα άλλα χαρακτηριστικά του δικτύου. Σε περιβάλλοντα ασφάλειας δικτύων, η σημασία της διαφοράς στην ακρίβεια εξαρτάται από το πλαίσιο εφαρμογής και το αποδεκτό επίπεδο κινδύνου.

Μοντέλο	Ακρίβεια με όλα τα χαρακτηριστικά / Χρόνος με όλα τα χαρακτηριστικά	Ακρίβεια με 3 χαρακτηριστικά / Χρόνος με 3 χαρακτηριστικά
Decision Tree Classifier	0.9972 0.2159 seconds	0.9896 0.1415 seconds
Random Forest Classifier	0.9984 3.9234 seconds	0.9880 2.7530 seconds
Extra Trees Classifier	0.9984 1.5049 seconds	0.9704 0.8891 seconds
Μέσος χρόνος εκπαίδευσης	3.0705 seconds	1.2629 seconds

Εικόνα 13: Σύγκριση των δύο μοντέλων πρόβλεψης

7 ΣΥΜΠΕΡΑΣΜΑ

Με την εφαρμογή αλγορίθμων μηχανικής μάθησης, είναι δυνατόν να εντοπιστούν και να αναλυθούν οι κρίσιμες παράμετροι που επηρεάζουν τις επιθέσεις δικτύων. Η σημαντικότητα των χαρακτηριστικών εξαρτάται από την στρατηγική βαρύτητα που δίνει ο αλγόριθμος στις παραμέτρους κατά την εκπαίδευσή του. Η ανάλυση αυτή επιτρέπει τον εντοπισμό των σημαντικότερων χαρακτηριστικών που εμπλέκονται σε κακόβουλες δραστηριότητες, γεγονός που μπορεί να οδηγήσει σε πιο αποδοτικά και αποτελεσματικά συστήματα ανίχνευσης και αντιμετώπισης απειλών. Εστιάζοντας σε αυτά τα κορυφαία κοινά χαρακτηριστικά μπορούν να αναπτυχθούν συστήματα ασφαλείας με χαμηλότερη πολυπλοκότητα, τα οποία δεν απαιτούν την επεξεργασία μεγάλου όγκου δεδομένων για την ανίχνευση απειλών. Αυτό έχει ως αποτέλεσμα ταχύτερους χρόνους απόκρισης σε πραγματικό χρόνο και βελτιωμένη ακρίβεια στην αναγνώριση και την αντιμετώπιση επιθέσεων. Τέτοιου είδους βελτιώσεις όχι μόνο ενισχύουν την προστασία των δικτύων, αλλά επίσης διευκολύνουν την ανάπτυξη και τη συντήρηση συστημάτων ασφαλείας, καθιστώντας τα πιο αποδοτικά και προσαρμοσμένα στις συνεχώς εξελισσόμενες απειλές.

8 ΣΥΖΗΤΗΣΗ

Τα χαρακτηριστικά που αναγνωρίζονται ως σημαντικά από πολλούς αλγόριθμους υποδεικνύουν τη σημασία τους στη διάκριση μεταξύ των κλάσεων. Αυτά τα χαρακτηριστικά παρέχουν κρίσιμη πληροφορία για την πρόβλεψη της ετικέτας και θεωρούνται αξιόπιστα για την κατανόηση των δεδομένων. Η αναγνώριση αυτών των χαρακτηριστικών μπορεί να ενισχύσει την ερμηνευσιμότητα των μοντέλων και να βελτιώσει την απόδοσή τους. Αυτό καθίσταται εφικτό αφού απλοποιείται η διαδικασία μοντελοποίησης και μειώνεται η πολυπλοκότητα. Τα μοντέλα γίνονται πιο κατανοητά και διαχειρίσιμα.

Πράγματι, στην περίπτωση μας, αναδείχθηκαν τρία κορυφαία κοινά χαρακτηριστικά, που στηρίζονται στον όγκο της ροής των εισερχόμενων και εξερχόμενων δεδομένων, καθώς και στο πλήθος συνδέσεων προς έναν host. Η εκμετάλλευσή τους σε ένα νέο μοντέλο μας έδωσε

μια πολύ καλή τιμή ακριβείας, αυτής της τάξεως του 97-98%. Η δεδομένη τιμή αποτελεί ένα αρκετά καλό γενικό ποσοστό, εφόσον τα συνήθη αντίστοιχα ποσοστά ανίχνευσης επιθέσεων στην κυβερνοασφάλεια κυμαίνονται από 90-99%. Επιπλέον, f1 score - τάξεως 96-98% - μας δείχνει μια αρμονική εικόνα απόδοσης μεταξύ των μετρικών της ακρίβειας και της ανάκλησης. Τα αποτελέσματα της έρευνας υποδεικνύουν ότι η εξαγωγή σημαντικών χαρακτηριστικών από ένα εύρος πολλών παραμέτρων και η εστίαση σε αυτά, παρέχει μια απλοποιημένη και ταυτόχρονα αποτελεσματική προσέγγιση για τον εντοπισμό επιθέσεων, επιτυγχάνοντας καλή ακρίβεια με μειωμένη πολυπλοκότητα και χρόνο εκπαίδευσης. Παρά το γεγονός ότι η προσέγγιση αυτή προσφέρει σαφή πλεονεκτήματα όσον αφορά την ταχύτητα και την ερμηνευσιμότητα, ενδέχεται να μην είναι εξίσου αποτελεσματική σε πιο περίπλοκες ή ποικιλόμορφες επιθέσεις, οι οποίες μπορεί να απαιτούν έναν ευρύτερο σύνολο χαρακτηριστικών για την ακριβή ανίχνευση.

Μια πιθανή κατεύθυνση για περαιτέρω έρευνα στο κομμάτι που μελετήθηκε, μπορεί να είναι η δοκιμή διαφορετικών αλγορίθμων για την εξαγωγή χαρακτηριστικών και η σύγκρισή τους όσον αφορά την απόδοση των νέων μοντέλων. Ενδιαφέρον θα αποτελούσε και η επέκταση του συνόλου των χαρακτηριστικών με άλλα χαρακτηριστικά που μπορεί να μην εμφανίζονται ως η πλειοψηφία σε όλους τους αλγορίθμους, αλλά εντοπίζονται ως εξίσου κορυφαία από κάποιους μεμονομένους αλγορίθμους.

Οι εφαρμογές της μηχανικής μάθησης στην ασφάλεια δικτύων είναι πολυάριθμες και η βελτιστοποίηση των ήδη υπαρχόντων προγραμμάτων ανίχνευσης επιθέσεων αποτελεί αναγκαιότητα. Καθώς οι απαιτήσεις για ασφάλεια αυξάνονται συνεχώς, προκύπτουν νέες προκλήσεις από καινοφανείς παραλλαγές επιθέσεων. Η διαρκής εξέλιξη των τεχνικών μηχανικής μάθησης είναι κρίσιμη για την ανάπτυξη αποτελεσματικότερων λύσεων που μπορούν να ανταποκριθούν σε αυτές τις νέες απειλές, διασφαλίζοντας την ακεραιότητα και την ασφάλεια των δικτύων σε έναν ολοένα και πιο περίπλοκο ψηφιακό κόσμο.

9 ΚΩΔΙΚΕΣ ΥΛΟΠΟΙΗΣΗΣ

9.1 Κώδικες Data Analysis

Πληροφορίες dataset:

```
import pandas as pd

# Φόρτωση των δεδομένων
data = pd.read_csv("/content/Train_Data.csv")

# Εμφάνιση των πραγματικών ονομάτων των χαρακτηριστικών
print(data.head())
# ή
print(data.columns)
```

1. Εισαγωγή βιβλιοθηκών

2. Φόρτωση των δεδομένων

3. Εμφάνιση των πρώτων 5 γραμμών του DataFrame:

- Η μέθοδος data.head() εμφανίζει τις πρώτες 5 γραμμές του DataFrame. Διακρίνονται τα ονόματα των στηλών (χαρακτηριστικά), και οι πρώτες τιμές κάθε χαρακτηριστικού.

4. Εμφάνιση όλων των ονομάτων των στηλών (χαρακτηριστικών):

- Η μέθοδος data.columns επιστρέφει όλα τα ονόματα των στηλών του DataFrame, χωρίς τις τιμές των δεδομένων, σε μορφή Index object.

Scatter plots:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Φόρτωση των δεδομένων
data = pd.read_csv("/content/Train_Data.csv")

# Εντοπισμός ακραίων τιμών με βάση το κριτήριο IQR
Q1 = data['src_bytes'].quantile(0.25)
Q3 = data['src_bytes'].quantile(0.75)
IQR = Q3 - Q1
```

```

# Ορισμός των κατωφλίων για το κριτήριο IQR
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Φιλτράρισμα ακραίων τιμών
filtered_data = data[(data['src_bytes'] >= lower_bound) &
                     (data['src_bytes'] <= upper_bound)]

# Οπτικοποίηση των δεδομένων με scatter plots χωρίς τις ακραίες τιμές
plt.figure(figsize=(10, 6))
sns.scatterplot(data=filtered_data, x='duration', y='src_bytes',
                hue='class', palette='viridis')
plt.title('Scatter Plot of Duration vs. Src_bytes without Outliers',
          fontsize=16)
plt.xlabel('Duration', fontsize=12)
plt.ylabel('Src_bytes', fontsize=12)
plt.legend(title='Class', loc='upper right')
plt.show()

```

1. Εισαγωγή βιβλιοθηκών:

- ‘pandas’ για τη φόρτωση και την επεξεργασία των δεδομένων.
- ‘seaborn’ για την οπτικοποίηση δεδομένων.
- ‘matplotlib.pyplot’ για τη δημιουργία γραφημάτων.

2. Φόρτωση των δεδομένων:

- Φορτώνονται τα δεδομένα από ένα αρχείο CSV που βρίσκεται στη διαδρομή /content/Train_Data.csv και τα αποθηκεύονται στο DataFrame data.

3. Εντοπισμός ακραίων τιμών με βάση το κριτήριο IQR:

- Ορίζονται τα πρώτα και τρίτα τεταρτημόρια (Q1 και Q3) της στήλης src_bytes.
- Το IQR (Interquartile Range) υπολογίζεται ως η διαφορά μεταξύ του Q3 και του Q1. Το IQR είναι ένα μέτρο της διάσπασης των τιμών στο μέσο του dataset και χρησιμοποιείται για τον εντοπισμό των ακραίων τιμών

4.Ορισμός των κατωφλίων για το κριτήριο IQR:

- Υπολογίζονται τα κατώφλια των ακραίων τιμών:
 - Το lower_bound είναι το κατώτερο όριο τιμών, κάτω από το οποίο οι τιμές θεωρούνται ακραίες.

- Το `upper_bound` είναι το ανώτερο όριο τιμών, πάνω από το οποίο οι τιμές θεωρούνται ακραίες.

5. Φιλτράρισμα ακραίων τιμών:

- Το DataFrame `filtered_data` δημιουργείται περιέχοντας μόνο τις γραμμές των δεδομένων που έχουν τιμές στη στήλη `src_bytes` εντός των ορίων που ορίστηκαν (δηλαδή, χωρίς ακραίες τιμές).

6. Οπτικοποίηση των δεδομένων με scatter plot χωρίς τις ακραίες τιμές:

- Δημιουργείται ένα scatter plot με τη χρήση της βιβλιοθήκης Seaborn:
 - Στον άξονα x απεικονίζεται η στήλη `duration`.
 - Στον άξονα y απεικονίζεται η στήλη `src_bytes` χωρίς τις ακραίες τιμές.
 - Τα σημεία χρωματίζονται ανάλογα με την κατηγορία τους (`class`).
 - Ορίζεται τίτλος και ετικέτες για τους άξονες και δημιουργείται ένας θρύλος που δείχνει την αντιστοίχιση των χρωμάτων στις κατηγορίες.
 - Το plot εμφανίζεται μέσω της `plt.show()`.

Ιστογράμματα, box plots και pair plots:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Φόρτωση των δεδομένων
data = pd.read_csv("/content/Train_Data.csv")

# Επιλογή αριθμητικών στηλών
numeric_cols = data.select_dtypes(include=['int64', 'float64']).columns

# Αντιμετώπιση ακραίων τιμών στα histograms μόνο για τα αριθμητικά
χαρακτηριστικά
plt.figure(figsize=(12, 10))

# Ορισμός των percentiles για την clip
lower_percentile = 0.05
upper_percentile = 0.95

# Υπολογισμός των νέων τιμών για το clip
```

```

lower_bound = data[numeric_cols].quantile(lower_percentile)
upper_bound = data[numeric_cols].quantile(upper_percentile)

# Εφαρμογή του clip
data_clipped = data.copy()
data_clipped[numeric_cols] = data[numeric_cols].clip(lower=lower_bound,
upper=upper_bound, axis=1)

# Εμφάνιση των ιστογραμμάτων
data_clipped.hist(bins=20, figsize=(20, 15))
plt.suptitle('Histograms of Features (Outliers Clipped)', fontsize=16)
plt.show()

# Αντιμετώπιση ακραίων τιμών στα box plots με λογαριθμική κλίμακα
# Εφαρμογή λογαριθμικής κλίμακας στα αριθμητικά δεδομένα
data_log = data_clipped.copy()
data_log[numeric_cols] = np.log1p(data_clipped[numeric_cols])

# Δημιουργία box plots με λογαριθμική κλίμακα
plt.figure(figsize=(15, 10))
sns.boxplot(data=data_log.select_dtypes(include=['int64', 'float64']))
plt.title('Box Plots of Numerical Features (Log Scale, Outliers
Clipped)', fontsize=16)
plt.xticks(rotation=90)
plt.show()

# Αντιμετώπιση ακραίων τιμών στα pair plots
sns.pairplot(data_clipped, hue='class', vars=['duration', 'src_bytes',
'dst_bytes', 'count'])
plt.suptitle('Pair Plots of Selected Features (Outliers Clipped)',
y=1.02, fontsize=16)
plt.show()

```

1. Εισαγωγή βιβλιοθηκών

2. Φόρτωση των δεδομένων

3. Επιλογή αριθμητικών στηλών:

- Εδώ επιλέγονται μόνο οι αριθμητικές στήλες (int64 και float64) από το DataFrame και αποθηκεύονται τα ονόματά τους στη λίστα numeric_cols. Αυτό γίνεται για να περιοριστεί η ανάλυση στα αριθμητικά χαρακτηριστικά.

4. Ορισμός των percentiles για την αντιμετώπιση ακραίων τιμών:

- Εδώ ορίζονται τα ποσοστά lower_percentile και upper_percentile στο 5% και 95% αντίστοιχα. Αυτά τα ποσοστά χρησιμοποιούνται για να αντιμετωπιστούν

οι ακραίες τιμές. Η διαδικασία αυτή επιτρέπει τον περιορισμό των ακραίων τιμών μεταξύ αυτών των percentiles.

5. Υπολογισμός των νέων ορίων:

- Υπολογίζονται τα όρια για το clip των αριθμητικών στηλών:
 - lower_bound: περιέχει το 5ο ποσοστό για κάθε αριθμητική στήλη.
 - upper_bound: περιέχει το 95ο ποσοστό για κάθε αριθμητική στήλη.

6. Εφαρμογή της συνάρτησης clip:

- Δημιουργείται ένα αντίγραφο του αρχικού DataFrame (data_clipped) και χρησιμοποιείται η συνάρτηση clip για να περιοριστούν οι τιμές των αριθμητικών στηλών εντός των ορίων που υπολογίστηκαν στο βήματα 5.

7. Εμφάνιση των ιστογραμμάτων:

- Δημιουργούνται και εμφανίζονται ιστογράμματα για κάθε αριθμητική στήλη του data_clipped. Τα ιστογράμματα δείχνουν τη συχνότητα εμφάνισης των τιμών εντός των ορίων που ορίστηκαν με το clip.

8. Εμφάνιση των box plots:

- Δημιουργείται ένα box plot σε λογαριθμική κλίμακα για κάθε αριθμητική στήλη, εμφανίζοντας τη διάμεσο, τα quartiles και τις τυχόν ακραίες τιμές μετά το clipping. Αυτό μας βοηθά να κατανοήσουμε καλύτερα την κατανομή των τιμών.

9. Εμφάνιση των pair plots:

- Δημιουργούνται pair plots για συγκεκριμένες αριθμητικές στήλες (duration, src_bytes, dst_bytes, count). Το hue='class' σημαίνει ότι τα σημεία θα χρωματίζονται ανάλογα με την τιμή της στήλης class, κάτι που βοηθά στην οπτική κατηγοριοποίηση. Τα pair plots δείχνουν τις σχέσεις μεταξύ των χαρακτηριστικών αυτών.

Προσδιορισμός συσχετίσεων των χαρακτηριστικών μέσω heatmap:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```

# Φόρτωση των δεδομένων
data = pd.read_csv("/content/Train_Data.csv")

# Μετατροπή κατηγορικών μεταβλητών σε αριθμητικές (one-hot encoding)
encoded_data = pd.get_dummies(data, drop_first=True)

# Υπολογισμός του πίνακα συσχέτισης
correlation_matrix_full = encoded_data.corr()

# Φιλτράρισμα των συσχετίσεων με απόλυτη τιμή μεγαλύτερη από 0.70
high_corr = correlation_matrix_full[(correlation_matrix_full >= 0.7) |
(correlation_matrix_full <= -0.7)]

# Δημιουργία heatmap για τις υψηλές συσχετίσεις
plt.figure(figsize=(15, 12))
sns.heatmap(high_corr, annot=True, cmap="coolwarm", fmt='.2f',
linewidths=0.5)
plt.title('Heatmap of Correlations > 70%')
plt.show()

# Εμφάνιση συσχετίσεων > 70% μία προς μία
high_corr_pairs = high_corr.unstack().dropna()

# Κρατάμε μόνο τα μοναδικά ζεύγη (αποφυγή συμμετρικών διπλοτύπων)
unique_pairs =
high_corr_pairs[high_corr_pairs.index.get_level_values(0) !=
high_corr_pairs.index.get_level_values(1)]
unique_pairs = unique_pairs[abs(unique_pairs) >
0.7].sort_values(ascending=False)

# Εμφάνιση των ζευγών με συσχέτιση > 70%
for pair, correlation in unique_pairs.items():
    print(f'Correlation between {pair[0]} and {pair[1]}:
{correlation:.2f}')

```

1. Εισαγωγή βιβλιοθηκών

2. Φόρτωση των δεδομένων

3. Κωδικοποίηση κατηγορικών δεδομένων σε αριθμητικά (one-hot encoding):

- `encoded_data = pd.get_dummies(data, drop_first=True)`: Όλα τα κατηγορικά δεδομένα μετατρέπονται σε αριθμητικές τιμές χρησιμοποιώντας την τεχνική one-hot encoding. Αυτό σημαίνει ότι κάθε κατηγορική μεταβλητή θα διασπαστεί σε δυαδικές στήλες. Η επιλογή `drop_first=True` αποφεύγει το πρόβλημα της συγγραμμικότητας (dummy variable trap), διαγράφοντας την πρώτη κατηγορία.

4. Υπολογισμός του πίνακα συσχέτισης:

- `correlation_matrix_full = encoded_data.corr()`: Υπολογίζεται ο πίνακας συσχέτισης μεταξύ όλων των χαρακτηριστικών στο `encoded_data`. Αυτός ο πίνακας περιέχει τιμές που κυμαίνονται από -1 (τέλεια αρνητική συσχέτιση) έως 1 (τέλεια θετική συσχέτιση), με 0 να δείχνει καμία συσχέτιση.

5. Φιλτράρισμα συσχετίσεων με απόλυτη τιμή μεγαλύτερη από 0.70:

- `high_corr = correlation_matrix_full[(correlation_matrix_full >= 0.7) | (correlation_matrix_full <= -0.7)]`: Φιλτράρονται οι συσχετίσεις που έχουν απόλυτη τιμή μεγαλύτερη ή ίση με 0.70 (είτε θετικές είτε αρνητικές). Δηλαδή, κρατούνται μόνο οι ισχυρές συσχετίσεις.

6. Αφαίρεση της διαγωνίου του πίνακα:

- `np.fill_diagonal`: αντικαθίστανται οι τιμές της διαγωνίου με NaN. Η διαγώνιος περιέχει πάντα την τιμή 1, επειδή κάθε χαρακτηριστικό συσχετίζεται τέλεια με τον εαυτό του. Αντικαθιστώντας αυτές τις τιμές με NaN, αποφεύγεται η συμμετοχή τους στα επόμενα βήματα.

7. Δημιουργία heatmap για τις υψηλές συσχετίσεις:

- `sns.heatmap(high_corr, annot=True, cmap="coolwarm", fmt='.2f', linewidths=0.5)`: Χρησιμοποιείται η βιβλιοθήκη `seaborn` για να δημιουργηθεί ένα heatmap που οπτικοποιεί τις συσχετίσεις μεγαλύτερες του 70%. Στο heatmap αυτό, οι αποχρώσεις χρωμάτων απεικονίζουν τη δύναμη και την κατεύθυνση της συσχέτισης (θετική ή αρνητική), ενώ οι αριθμητικές τιμές εμφανίζονται πάνω στο γράφημα με δύο δεκαδικά ψηφία.

8. Εύρεση και φιλτράρισμα των συσχετίσεων για εμφάνιση:

- `high_corr_pairs = high_corr.unstack().dropna()`: Τα δεδομένα του πίνακα συσχέτισης "ξετυλίγονται" (`unstack`), ώστε να εμφανίζονται ως ζεύγη αντί για πίνακα. Στη συνέχεια, οι κενές τιμές αφαιρούνται με τη μέθοδο `dropna()`.
- `unique_pairs = high_corr_pairs[high_corr_pairs.index.get_level_values(0) != high_corr_pairs.index.get_level_values(1)]`: Φιλτράρονται οι συσχετίσεις που εμφανίζονται δύο φορές λόγω συμμετρίας (π.χ., συσχέτιση A-B και B-A). Το

get_level_values(0) και get_level_values(1) εξασφαλίζει ότι τα ζεύγη είναι μοναδικά.

- unique_pairs = unique_pairs[abs(unique_pairs) > 0.7].sort_values(ascending=False): Κρατούνται μόνο οι συσχετίσεις που έχουν απόλυτη τιμή μεγαλύτερη από 0.7 και ταξινομούνται με φθίνουσα σειρά.

9. Εμφάνιση των ζευγών με συσχέτιση > 70%:

- for pair, correlation in unique_pairs.items(): Τέλος, εμφανίζονται ένα προς ένα τα ζεύγη των χαρακτηριστικών που έχουν συσχέτιση πάνω από 70%, με τις συσχετίσεις τους να τυπώνονται με ακρίβεια δύο δεκαδικών.

Υπολογισμός τυπικής απόκλισης:

```
import pandas as pd

# Φόρτωση το dataset
df = pd.read_csv("/content/Train_Data.csv")

# Υπολόγισε την τυπική απόκλιση για τις αριθμητικές στήλες
numeric_std_devs = df.select_dtypes(include=['number']).std()

# Εμφάνισε την τυπική απόκλιση για τις αριθμητικές στήλες
print("Τυπική απόκλιση για αριθμητικές στήλες:")
print(numeric_std_devs)

# Εμφάνισε μήνυμα για στήλες που δεν είναι αριθμητικές
non_numeric_columns = df.select_dtypes(exclude=['number']).columns
print("\nΣτήλες με μη αριθμητικά δεδομένα:")
for column in non_numeric_columns:
    print(f"{column}: Δεν μπορεί να υπολογιστεί η τυπική απόκλιση")
```

1. Εισαγωγή βιβλιοθηκών
2. Φόρτωση των δεδομένων
3. Επιλογή και υπολογισμός τυπικής απόκλισης για τις αριθμητικές στήλες:

- `df.select_dtypes(include=['number'])`: Η μέθοδος με την οποία επιλέγονται μόνο οι στήλες που περιέχουν αριθμητικά δεδομένα από το DataFrame.
- `.std()`: Υπολογίζεται η τυπική απόκλιση (standard deviation) για κάθε μία από τις αριθμητικές στήλες.

[Η τυπική απόκλιση δείχνει πόσο διαφέρουν οι τιμές μιας στήλης από τον μέσο όρο της. Όσο μεγαλύτερη είναι η τυπική απόκλιση, τόσο πιο διασκορπισμένες είναι οι τιμές γύρω από τον μέσο όρο. Αποτελεί επίσης δείκτη της προσφερόμενης πληροφορίας.]

4. Εκτύπωση των αποτελεσμάτων για τις αριθμητικές στήλες

5. Εντοπισμός και εμφάνιση των μη αριθμητικών στηλών:

- `df.select_dtypes(exclude=['number'])`: Επιλέγονται οι στήλες που δεν περιέχουν αριθμητικά δεδομένα (κείμενο, κατηγορικές μεταβλητές).
- `.columns`: Παίρνει τα ονόματα αυτών των στηλών.

Η τυπική απόκλιση δεν μπορεί να υπολογιστεί για κατηγορικά ή μη αριθμητικά δεδομένα.

9.2 Κώδικες Εξαγωγής Κορυφαίων Χαρακτηριστικών

Εξέταση κορυφαίων χαρακτηριστικών ανάμεσα σε όλα τα μοντέλα

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Διαβάζουμε τα δεδομένα από το CSV αρχείο
train_data = pd.read_csv("/content/Train_Data.csv")

# Χωρίζουμε τα χαρακτηριστικά (X) και την ετικέτα (y)
X = train_data.drop('class', axis=1) # Χαρακτηριστικά
y = train_data['class'] # Ετικέτα
```

```

# Χωρίζουμε τα δεδομένα σε training και validation set (10% για
validation)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1,
random_state=42)

# Χωρίζουμε τα αριθμητικά και μη αριθμητικά χαρακτηριστικά
numeric_cols = X_train.select_dtypes(include=['float64',
'int64']).columns
categorical_cols = X_train.select_dtypes(include=['object']).columns

# Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά με το
μέσο όρο
imputer_numeric = SimpleImputer(strategy='mean')
X_train_numeric =
pd.DataFrame(imputer_numeric.fit_transform(X_train[numeric_cols]),
columns=numeric_cols)
X_val_numeric =
pd.DataFrame(imputer_numeric.transform(X_val[numeric_cols]),
columns=numeric_cols)

# Κωδικοποίηση OneHot στα μη αριθμητικά χαρακτηριστικά
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
X_train_categorical_encoded =
encoder.fit_transform(X_train[categorical_cols])
X_val_categorical_encoded = encoder.transform(X_val[categorical_cols])

# Δημιουργία νέων ονομάτων χαρακτηριστικών για τα κωδικοποιημένα
κατηγορικά δεδομένα
new_feature_names = encoder.get_feature_names_out(categorical_cols)
X_train_categorical = pd.DataFrame(X_train_categorical_encoded,
columns=new_feature_names)
X_val_categorical = pd.DataFrame(X_val_categorical_encoded,
columns=new_feature_names)

# Συνδυασμός αριθμητικών και δυαδικών χαρακτηριστικών
X_train_final = pd.concat([X_train_numeric, X_train_categorical],
axis=1)
X_val_final = pd.concat([X_val_numeric, X_val_categorical], axis=1)

# Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά
columns_to_remove = [
    'num_outbound_cmds', 'is_host_login', 'num_root',
    'srv_serror_rate', 'srv_rerror_rate', 'dst_host_srv_serror_rate',
    'flag_S0', 'dst_host_same_srv_rate', 'is_guest_login',
    'service_domain_u', 'flag_SF', 'service_http',
    'protocol_type_udp', 'su_attempted', 'service_ftp',

```

```

    'error_rate', 'dst_host_srv_error_rate', 'dst_host_rerror_rate',
    'error_rate'
]

# Εξασφαλίζουμε ότι οι στήλες που θέλουμε να αφαιρέσουμε υπάρχουν
columns_to_remove = [col for col in columns_to_remove if col in
X_train_final.columns]

X_train_final = X_train_final.drop(columns=columns_to_remove)
X_val_final = X_val_final.drop(columns=columns_to_remove)

# Δημιουργία λεξικού για να αποθηκεύσουμε τις μέσες ακρίβειες
accuracy_results = {}

# Εκπαίδευση μοντέλων και υπολογισμός της ακρίβειας
n_iterations = 20

# Decision Tree Classifier
dt_accuracies = []
print("\nDecision Tree Classifier:")
for i in range(n_iterations):
    dt_classifier = DecisionTreeClassifier(random_state=42)
    dt_classifier.fit(X_train_final, y_train)
    dt_predictions = dt_classifier.predict(X_val_final)
    dt_accuracy = accuracy_score(y_val, dt_predictions)
    dt_accuracies.append(dt_accuracy)
    print(f"Iteration {i + 1}: Accuracy = {dt_accuracy:.4f}")

accuracy_results['Decision Tree'] = sum(dt_accuracies) / n_iterations
print(f'Average accuracy for Decision Tree: {accuracy_results["Decision
Tree"]:.4f}\n')

# Random Forest Classifier
rf_accuracies = []
print("Random Forest Classifier:")
for i in range(n_iterations):
    rf_classifier = RandomForestClassifier(random_state=42)
    rf_classifier.fit(X_train_final, y_train)
    rf_predictions = rf_classifier.predict(X_val_final)
    rf_accuracy = accuracy_score(y_val, rf_predictions)
    rf_accuracies.append(rf_accuracy)
    print(f"Iteration {i + 1}: Accuracy = {rf_accuracy:.4f}")

accuracy_results['Random Forest'] = sum(rf_accuracies) / n_iterations
print(f'Average accuracy for Random Forest: {accuracy_results["Random
Forest"]:.4f}\n')

```

```

# Gradient Boosting Classifier
gb_accuracies = []
print("Gradient Boosting Classifier:")
for i in range(n_iterations):
    gb_classifier = GradientBoostingClassifier(random_state=42)
    gb_classifier.fit(X_train_final, y_train)
    gb_predictions = gb_classifier.predict(X_val_final)
    gb_accuracy = accuracy_score(y_val, gb_predictions)
    gb_accuracies.append(gb_accuracy)
    print(f"Iteration {i + 1}: Accuracy = {gb_accuracy:.4f}")

accuracy_results['Gradient Boosting'] = sum(gb_accuracies) /
n_iterations
print(f'Average accuracy for Gradient Boosting:
{accuracy_results["Gradient Boosting"]:.4f}\n')

# Extra Trees Classifier
et_accuracies = []
print("Extra Trees Classifier:")
for i in range(n_iterations):
    et_classifier = ExtraTreesClassifier(random_state=42)
    et_classifier.fit(X_train_final, y_train)
    et_predictions = et_classifier.predict(X_val_final)
    et_accuracy = accuracy_score(y_val, et_predictions)
    et_accuracies.append(et_accuracy)
    print(f"Iteration {i + 1}: Accuracy = {et_accuracy:.4f}")

accuracy_results['Extra Trees'] = sum(et_accuracies) / n_iterations
print(f'Average accuracy for Extra Trees: {accuracy_results["Extra
Trees"]:.4f}\n')

# Εξέταση κοινών κορυφαίων χαρακτηριστικών ανάμεσα στα μοντέλα
dt_feature_importances = pd.Series(dt_classifier.feature_importances_,
index=X_train_final.columns)
rf_feature_importances = pd.Series(rf_classifier.feature_importances_,
index=X_train_final.columns)
gb_feature_importances = pd.Series(gb_classifier.feature_importances_,
index=X_train_final.columns)
et_feature_importances = pd.Series(et_classifier.feature_importances_,
index=X_train_final.columns)

dt_top_features =
dt_feature_importances.sort_values(ascending=False).index[:5]
rf_top_features =
rf_feature_importances.sort_values(ascending=False).index[:5]
gb_top_features =
gb_feature_importances.sort_values(ascending=False).index[:5]

```

```
et_top_features =  
et_feature_importances.sort_values(ascending=False).index[:5]  
  
common_features = set(dt_top_features).intersection(rf_top_features,  
gb_top_features, et_top_features)  
print("Common top features across models:", list(common_features))
```

1. Εισαγωγή Βιβλιοθηκών:

- **‘pandas’**: Για την επεξεργασία δεδομένων.
- **‘sklearn.preprocessing.OneHotEncoder’**: Για την κωδικοποίηση κατηγορικών χαρακτηριστικών.
- **‘sklearn.impute.SimpleImputer’**: Για την αντιμετώπιση απουσιάζουσών τιμών.
- **‘sklearn.ensemble’ και ‘sklearn.tree’**: Για τη χρήση διαφόρων αλγορίθμων ταξινόμησης.
- **‘sklearn.model_selection.train_test_split’**: Για τη διαίρεση των δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης.
- **‘accuracy_score’**: Μετρική αξιολόγησης της απόδοσης των μοντέλων.

2. Φόρτωση των δεδομένων:

- Τα δεδομένα φορτώνονται από το αρχείο CSV και αποθηκεύονται σε ένα DataFrame data.

3. Διαχωρισμός χαρακτηριστικών και ετικέτας:

- Διαχωρίζονται τα χαρακτηριστικά (X) από την ετικέτα (y). Η στήλη class είναι η ετικέτα που προσπαθούμε να προβλέψουμε, ενώ τα υπόλοιπα είναι χαρακτηριστικά.

4. Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης:

- Διαχωρισμός των δεδομένων εκπαίδευσης σε δύο σύνολα: X_train και y_train για εκπαίδευση του μοντέλου, και X_val και y_val για επικύρωση. Το 10% των δεδομένων χρησιμοποιείται για επικύρωση.

5. Διαχωρισμός αριθμητικών και κατηγορικών χαρακτηριστικών:

- Εντοπίζονται τα αριθμητικά και κατηγορικά χαρακτηριστικά στα δεδομένα εκπαίδευσης.

6. Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά:

- Χρήση του SimpleImputer για να αντικατασταθούν οι απουσιάζουσες τιμές στις αριθμητικές στήλες με τον μέσο όρο τους. Εφαρμόζεται ο ίδιος μετασχηματιστής στα δεδομένα επικύρωσης.

7. Κωδικοποίηση κατηγορικών χαρακτηριστικών:

- Κωδικοποιούνται τα κατηγορικά χαρακτηριστικά σε δυαδικές τιμές χρησιμοποιώντας τον OneHotEncoder. Η παράμετρος `handle_unknown='ignore'` επιτρέπει τη διαχείριση νέων κατηγοριών στα δεδομένα επικύρωσης που δεν εμφανίζονται στα δεδομένα εκπαίδευσης.

8. Δημιουργία Νέων Ονομάτων Χαρακτηριστικών:

- Δημιουργούνται νέα ονόματα χαρακτηριστικών για τα κωδικοποιημένα κατηγορικά δεδομένα.

9. Συνδυασμός αριθμητικών και κατηγορικών χαρακτηριστικών:

- Συνδυάζονται τα προεπεξεργασμένα αριθμητικά και κατηγορικά χαρακτηριστικά σε ένα τελικό DataFrame για εκπαίδευση και επικύρωση των μοντέλων.

10. Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά:

- Ορισμός στηλών προς αφαίρεση: Δημιουργείται μια λίστα με στήλες που θεωρείται ότι πρέπει να αφαιρεθούν λόγω bias.
- Αφαίρεση από τα δεδομένα: Με την εντολή `drop`, οι στήλες αφαιρούνται από τα σύνολα χαρακτηριστικών για το training set (`X_train_final`) και το validation set (`X_val_final`). Αυτό διασφαλίζει ότι και τα δύο σύνολα δεδομένων περιέχουν τα ίδια χαρακτηριστικά για σωστή εκπαίδευση και αξιολόγηση.
- Έλεγχος ύπαρξης στηλών: Πριν αφαιρεθούν, ελέγχεται αν οι στήλες υπάρχουν στο dataset για να αποφευχθούν σφάλματα.

11. Εκπαίδευση μοντέλων και υπολογισμός της ακρίβειας εκπαίδευσης και της σημαντικότητας των χαρακτηριστικών:

➤ Εκπαιδεύονται οι

- DecisionTreeClassifier
- Random Forest Classifier
- Gradient Boosting Classifier
- Extra Trees Classifier

20 φορές και αποθηκεύεται η ακρίβεια κάθε επανάληψης. Στο τέλος υπολογίζεται η μέση ακρίβεια και η σημαντικότητα των χαρακτηριστικών.

12. Εξέταση κοινών κορυφαίων χαρακτηριστικών ανάμεσα στα μοντέλα:

- Εξετάζονται τα κορυφαία 5 χαρακτηριστικά για κάθε μοντέλο, με βάση τη σημαντικότητα, με τη χρήση της 'feature_importancies' και έπειτα, χρησιμοποιώντας set intersection, βρίσκονται τα κοινά σημαντικά χαρακτηριστικά και εκτυπώνονται.

Εξέταση κορυφαίων χαρακτηριστικών ανάμεσα στους αλγορίθμους ανά ζεύγη

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from itertools import combinations

# Διαβάζουμε τα δεδομένα από τα CSV αρχεία
train_data = pd.read_csv("/content/Train_Data.csv")

# Χωρίζουμε τα χαρακτηριστικά (X) και την ετικέτα (y) από τα δεδομένα
εκπαίδευσης
X = train_data.drop('class', axis=1) # Χαρακτηριστικά
y = train_data['class'] # Ετικέτα

# Χωρίζουμε τα δεδομένα σε training και validation set (10% για
validation)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1,
random_state=42)

# Χωρίζουμε τα αριθμητικά και μη αριθμητικά χαρακτηριστικά
numeric_cols = X_train.select_dtypes(include=['float64',
'int64']).columns
categorical_cols = X_train.select_dtypes(include=['object']).columns

# Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά με το
μέσο όρο
```

```

imputer_numeric = SimpleImputer(strategy='mean')
X_train_numeric =
pd.DataFrame(imputer_numeric.fit_transform(X_train[numeric_cols]),
columns=numeric_cols)
X_val_numeric =
pd.DataFrame(imputer_numeric.transform(X_val[numeric_cols]),
columns=numeric_cols)

# Κωδικοποίηση OneHot στα μη αριθμητικά χαρακτηριστικά
encoder = OneHotEncoder(sparse=False, handle_unknown='ignore')
X_train_categorical_encoded =
encoder.fit_transform(X_train[categorical_cols])
X_val_categorical_encoded = encoder.transform(X_val[categorical_cols])

# Δημιουργία νέων ονομάτων χαρακτηριστικών για τα κωδικοποιημένα
κατηγορικά δεδομένα
new_feature_names = encoder.get_feature_names_out(categorical_cols)
X_train_categorical = pd.DataFrame(X_train_categorical_encoded,
columns=new_feature_names)
X_val_categorical = pd.DataFrame(X_val_categorical_encoded,
columns=new_feature_names)

# Συνδυασμός αριθμητικών και δυαδικών χαρακτηριστικών
X_train_final = pd.concat([X_train_numeric, X_train_categorical],
axis=1)
X_val_final = pd.concat([X_val_numeric, X_val_categorical], axis=1)

# Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά
columns_to_remove = [
    'num_outbound_cmds', 'is_host_login', 'num_root',
    'srv_error_rate', 'srv_rerror_rate', 'dst_host_srv_error_rate',
    'flag_S0', 'dst_host_same_srv_rate', 'is_guest_login',
    'service_domain_u', 'flag_SF', 'service_http',
    'protocol_type_udp', 'su_attempted', 'service_ftp',
    'error_rate', 'dst_host_srv_rerror_rate', 'dst_host_rerror_rate',
    'error_rate'
]

# Εξασφαλίζουμε ότι οι στήλες που θέλουμε να αφαιρέσουμε υπάρχουν
columns_to_remove = [col for col in columns_to_remove if col in
X_train_final.columns]

X_train_final = X_train_final.drop(columns=columns_to_remove)
X_val_final = X_val_final.drop(columns=columns_to_remove)

```

```

# Εκπαίδευση μοντέλων και υπολογισμός της σημαντικότητας των
χαρακτηριστικών

# Οι αλγόριθμοι που θα χρησιμοποιηθούν
classifiers = [
    ("Decision Tree Classifier",
    DecisionTreeClassifier(random_state=42)),
    ("Random Forest Classifier",
    RandomForestClassifier(random_state=42)),
    ("Gradient Boosting Classifier",
    GradientBoostingClassifier(random_state=42)),
    ("Extra Trees Classifier", ExtraTreesClassifier(random_state=42))
]

# Διατρέχουμε κάθε δυνατό ζευγάρι αλγορίθμων
for (clf1_name, clf1_model), (clf2_name, clf2_model) in
combinations(classifiers, 2):
    # Εκπαίδευση του πρώτου αλγορίθμου
    clf1_model.fit(X_train_final, y_train)
    clf1_feature_importances =
pd.Series(clf1_model.feature_importances_, index=X_train_final.columns)

    # Εκπαίδευση του δεύτερου αλγορίθμου
    clf2_model.fit(X_train_final, y_train)
    clf2_feature_importances =
pd.Series(clf2_model.feature_importances_, index=X_train_final.columns)

    # Εύρεση των κοινών χαρακτηριστικών
    clf1_top_features = set(clf1_feature_importances.nlargest(5).index)
    clf2_top_features = set(clf2_feature_importances.nlargest(5).index)
    common_features = clf1_top_features.intersection(clf2_top_features)

    # Εκτύπωση των κοινών χαρακτηριστικών ανά ζευγάρι αλγορίθμων
    print(f"Common top features between {clf1_name} and {clf2_name}:
    {list(common_features)}")

print("Model training and feature importance analysis are complete.")

```

1. Εισαγωγή Βιβλιοθηκών
2. Φόρτωση Δεδομένων
3. Διαχωρισμός Χαρακτηριστικών και Ετικετών
4. Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης
5. Επεξεργασία Αριθμητικών και Κατηγορικών Χαρακτηριστικών
6. Δημιουργία DataFrames με Κωδικοποιημένα Κατηγορικά Χαρακτηριστικά

7. Δημιουργία Νέων Ονομάτων Χαρακτηριστικών

8. Συνδυασμός Αριθμητικών και Κατηγορικών Χαρακτηριστικών

9. Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά

10. Εκπαίδευση Μοντέλων και Ανάλυση Σημαντικότητας Χαρακτηριστικών:

- Διατρέχουμε κάθε δυνατό ζευγάρι αλγορίθμων: Χρησιμοποιείται η συνάρτηση combinations από τη βιβλιοθήκη itertools για να δημιουργηθούν όλα τα πιθανά ζευγάρια αλγορίθμων από τη λίστα classifiers.
- Εκπαίδευση του πρώτου αλγορίθμου: Ο πρώτος αλγόριθμος του ζευγαριού εκπαιδεύεται χρησιμοποιώντας τα δεδομένα εκπαίδευσης X_train_final και y_train. Μετά την εκπαίδευση, υπολογίζονται οι σημαντικότητες των χαρακτηριστικών (feature importances), δηλαδή πόσο σημαντικό θεωρεί το μοντέλο κάθε χαρακτηριστικό για την πρόβλεψη του στόχου. Αυτό αποθηκεύεται σε ένα pd.Series με τα ονόματα των χαρακτηριστικών ως δείκτες.
- Εκπαίδευση του δεύτερου αλγορίθμου: Αντίστοιχα, ο δεύτερος αλγόριθμος εκπαιδεύεται με τα ίδια δεδομένα εκπαίδευσης και υπολογίζεται η σπουδαιότητα των χαρακτηριστικών για αυτόν τον αλγόριθμο, που αποθηκεύονται με παρόμοιο τρόπο.
- Εύρεση των κορυφαίων χαρακτηριστικών: Για κάθε αλγόριθμο, επιλέγονται τα 5 χαρακτηριστικά με τη μεγαλύτερη σημαντικότητα χρησιμοποιώντας τη μέθοδο .nlargest(5). Τα κορυφαία χαρακτηριστικά αποθηκεύονται ως σύνολα (set), για να διευκολυνθεί η επόμενη σύγκριση.
- Εύρεση κοινών κορυφαίων χαρακτηριστικών: Χρησιμοποιείται η συνάρτηση intersection για να βρεθούν τα κοινά χαρακτηριστικά (μεταξύ των κορυφαίων 5 χαρακτηριστικών) και των δύο αλγορίθμων.

11.Εκτύπωση αποτελεσμάτων

Εξέταση κορυφαίων χαρακτηριστικών ανάμεσα στους αλγορίθμους ανά τριάδες

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
```

```

from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from itertools import combinations

# Διαβάζουμε τα δεδομένα από τα CSV αρχεία
train_data = pd.read_csv("/content/Train_Data.csv")

# Χωρίζουμε τα χαρακτηριστικά (X) και την ετικέτα (y) από τα δεδομένα
# εκπαίδευσης
X = train_data.drop('class', axis=1) # Χαρακτηριστικά
y = train_data['class'] # Ετικέτα

# Χωρίζουμε τα δεδομένα σε training και validation set (10% για
# validation)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1,
random_state=42)

# Χωρίζουμε τα αριθμητικά και μη αριθμητικά χαρακτηριστικά
numeric_cols = X_train.select_dtypes(include=['float64',
'int64']).columns
categorical_cols = X_train.select_dtypes(include=['object']).columns

# Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά με το
# μέσο όρο
imputer_numeric = SimpleImputer(strategy='mean')
X_train_numeric =
pd.DataFrame(imputer_numeric.fit_transform(X_train[numeric_cols]),
columns=numeric_cols)
X_val_numeric =
pd.DataFrame(imputer_numeric.transform(X_val[numeric_cols]),
columns=numeric_cols)

# Κωδικοποίηση OneHot στα μη αριθμητικά χαρακτηριστικά
encoder = OneHotEncoder(sparse=False, handle_unknown='ignore')
X_train_categorical_encoded =
encoder.fit_transform(X_train[categorical_cols])
X_val_categorical_encoded = encoder.transform(X_val[categorical_cols])

# Δημιουργία νέων ονομάτων χαρακτηριστικών για τα κωδικοποιημένα
# κατηγορικά δεδομένα
new_feature_names = encoder.get_feature_names_out(categorical_cols)
X_train_categorical = pd.DataFrame(X_train_categorical_encoded,
columns=new_feature_names)

```

```

X_val_categorical = pd.DataFrame(X_val_categorical_encoded,
                                columns=new_feature_names)

# Συνδυασμός αριθμητικών και δυαδικών χαρακτηριστικών
X_train_final = pd.concat([X_train_numeric, X_train_categorical],
                           axis=1)
X_val_final = pd.concat([X_val_numeric, X_val_categorical], axis=1)

# Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά
columns_to_remove = [
    'num_outbound_cmds', 'is_host_login', 'num_root',
    'srv_error_rate', 'srv_rerror_rate', 'dst_host_srv_error_rate',
    'flag_S0', 'dst_host_same_srv_rate', 'is_guest_login',
    'service_domain_u', 'flag_SF', 'service_http',
    'protocol_type_udp', 'su_attempted', 'service_ftp',
    'error_rate', 'dst_host_srv_rerror_rate', 'dst_host_rerror_rate',
    'rerror_rate'
]

# Εξασφαλίζουμε ότι οι στήλες που θέλουμε να αφαιρέσουμε υπάρχουν στο
# τελικό dataset
columns_to_remove = [col for col in columns_to_remove if col in
X_train_final.columns]

X_train_final = X_train_final.drop(columns=columns_to_remove)
X_val_final = X_val_final.drop(columns=columns_to_remove)

# Εκπαίδευση μοντέλων και υπολογισμός της σημαντικότητας των
# χαρακτηριστικών

# Οι αλγόριθμοι που θα χρησιμοποιηθούν
classifiers = [
    ("Decision Tree Classifier",
DecisionTreeClassifier(random_state=42)),
    ("Random Forest Classifier",
RandomForestClassifier(random_state=42)),
    ("Gradient Boosting Classifier",
GradientBoostingClassifier(random_state=42)),
    ("Extra Trees Classifier", ExtraTreesClassifier(random_state=42))
]

# Δημιουργία όλων των πιθανών συνδυασμών τριάδων
for (clf1_name, clf1_model), (clf2_name, clf2_model), (clf3_name,
clf3_model) in combinations(classifiers, 3):
    # Εκπαίδευση του πρώτου αλγορίθμου
    clf1_model.fit(X_train_final, y_train)

```

```

    clf1_feature_importances =
pd.Series(clf1_model.feature_importances_, index=X_train_final.columns)

# Εκπαίδευση του δεύτερου αλγορίθμου
clf2_model.fit(X_train_final, y_train)
clf2_feature_importances =
pd.Series(clf2_model.feature_importances_, index=X_train_final.columns)

# Εκπαίδευση του τρίτου αλγορίθμου
clf3_model.fit(X_train_final, y_train)
clf3_feature_importances =
pd.Series(clf3_model.feature_importances_, index=X_train_final.columns)

# Εύρεση των κοινών χαρακτηριστικών
clf1_top_features = set(clf1_feature_importances.nlargest(5).index)
clf2_top_features = set(clf2_feature_importances.nlargest(5).index)
clf3_top_features = set(clf3_feature_importances.nlargest(5).index)
common_features = clf1_top_features.intersection(clf2_top_features,
clf3_top_features)

# Εκτύπωση των κοινών χαρακτηριστικών
print(f"Common top features between {clf1_name}, {clf2_name} and
{clf3_name}: {list(common_features)}")

print("Model training and feature importance analysis are complete.")

```

1. Εισαγωγή Βιβλιοθηκών:
2. Φόρτωση Δεδομένων
3. Διαχωρισμός Χαρακτηριστικών και Ετικετών
4. Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης
5. Επεξεργασία Αριθμητικών και Κατηγορικών Χαρακτηριστικών
6. Δημιουργία DataFrames με Κωδικοποιημένα Κατηγορικά Χαρακτηριστικά
7. Δημιουργία Νέων Ονομάτων Χαρακτηριστικών
8. Συνδυασμός Αριθμητικών και Κατηγορικών Χαρακτηριστικών
9. Εκπαίδευση Μοντέλων και Ανάλυση Σημαντικότητας Χαρακτηριστικών:
 - Δημιουργία τριάδων αλγορίθμων: Χρησιμοποιείται η συνάρτηση combinations από τη βιβλιοθήκη itertools για να παραχθούν όλες οι πιθανές τριάδες των classifiers.
 - Εκπαίδευση του πρώτου αλγορίθμου και υπολογισμός της σημαντικότητας των χαρακτηριστικών (feature importances) του πρώτου αλγορίθμου, που

αποθηκεύονται σε ένα `pd.Series`. Αντίστοιχα γίνεται και η εκπαίδευση του δεύτερου και του τρίτου αλγορίθμου.

- Εύρεση κορυφαίων χαρακτηριστικών
- Εύρεση κοινών χαρακτηριστικών

10. Εκτύπωση των κοινών χαρακτηριστικών

9.3 Κώδικας μοντέλου πρόβλεψης τριών χαρακτηριστικών

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier

# Διαβάζουμε τα δεδομένα από το CSV αρχείο
data = pd.read_csv("/content/Train_Data.csv")

# Μετατροπή της ετικέτας σε αριθμητικές τιμές (0 και 1)
data['class'] = data['class'].map({'normal': 0, 'anomaly': 1}) #
Αλλαγή ανάλογα με τις ετικέτες στο dataset σας

# Επιλέγουμε τα χαρακτηριστικά 'src_bytes', 'dst_bytes' και
'dst_host_srv_count' και την ετικέτα
X = data[['src_bytes', 'dst_bytes', 'dst_host_srv_count']]
y = data['class']

# Αντιμετώπιση απουσιάζουσων τιμών με το μέσο όρο
X = X.fillna(X.mean())

# Δημιουργία ενός dictionary για την αποθήκευση των μέσων αποτελεσμάτων
results = {'clf_name': {'accuracy': [], 'f1': [], 'test_accuracy': []}}
for clf_name, _ in [
    ("Decision Tree Classifier", DecisionTreeClassifier()),
    ("Random Forest Classifier", RandomForestClassifier()),
    ("Gradient Boosting Classifier", GradientBoostingClassifier()),
    ("Extra Trees Classifier", ExtraTreesClassifier())
]
}]

# Εκτέλεση της διαδικασίας 20 φορές
for run in range(20):
```

```

# Διαχωρισμός των δεδομένων σε σύνολα εκπαίδευσης (train) και
δοκιμής (test)
X_train_full, X_test, y_train_full, y_test = train_test_split(X, y,
test_size=0.2, random_state=None)

# Δημιουργία του validation set (10% του αρχικού training set)
X_train, X_val, y_train, y_val = train_test_split(X_train_full,
y_train_full, test_size=0.1, random_state=None)

# Οι αλγόριθμοι που θα χρησιμοποιηθούν
classifiers = [
    ("Decision Tree Classifier", DecisionTreeClassifier()),
    ("Random Forest Classifier", RandomForestClassifier()),
    ("Gradient Boosting Classifier", GradientBoostingClassifier()),
    ("Extra Trees Classifier", ExtraTreesClassifier())
]

# Εκπαίδευση και αξιολόγηση κάθε μοντέλου
for clf_name, clf_model in classifiers:
    # Εκπαίδευση του μοντέλου
    clf_model.fit(X_train, y_train)

    # Αξιολόγηση του μοντέλου με το validation set
    y_val_pred = clf_model.predict(X_val)
    val_accuracy = accuracy_score(y_val, y_val_pred)
    results[clf_name]['accuracy'].append(val_accuracy)

    # Πρόβλεψη με το σύνολο δοκιμής
    y_test_pred = clf_model.predict(X_test)

    # Υπολογισμός της ακρίβειας και του F1 score
    test_accuracy = accuracy_score(y_test, y_test_pred)
    f1 = f1_score(y_test, y_test_pred)
    results[clf_name]['test_accuracy'].append(test_accuracy)
    results[clf_name]['f1'].append(f1)

    print(f"Run {run + 1} - {clf_name}:")
    print(f"  Validation Accuracy: {val_accuracy:.4f}")
    print(f"  Test Accuracy: {test_accuracy:.4f}")
    print(f"  Test F1 Score: {f1:.4f}")

# Υπολογισμός και εκτύπωση των μέσων αποτελεσμάτων
for clf_name in results:
    avg_accuracy = sum(results[clf_name]['accuracy']) /
len(results[clf_name]['accuracy'])
    avg_test_accuracy = sum(results[clf_name]['test_accuracy']) /
len(results[clf_name]['test_accuracy'])

```

```

    avg_f1 = sum(results[clf_name]['f1']) /
len(results[clf_name]['f1'])
    print(f"\n{clf_name}:")
    print(f"   Average Validation Accuracy: {avg_accuracy:.4f}")
    print(f"   Average Test Accuracy: {avg_test_accuracy:.4f}")
    print(f"   Average Test F1 Score: {avg_f1:.4f}")

```

1. Εισαγωγή Βιβλιοθηκών:

- ‘pandas’ για επεξεργασία δεδομένων.
- ‘train_test_split’ από το sklearn για διαχωρισμό των δεδομένων σε σύνολα εκπαίδευσης, validation, και δοκιμής.
- ‘accuracy_score’, ‘f1_score’: Μετρικές για την αξιολόγηση της απόδοσης των μοντέλων.
- Οι διάφοροι ταξινομητές από το sklearn για την εκπαίδευση και πρόβλεψη των μοντέλων.

2. Φόρτωση Δεδομένων:

- Φορτώνει τα δεδομένα από το αρχείο CSV.

3. Μετατροπή Ετικετών:

- Μετατρέπει τις κατηγορίες των ετικετών normal και anomaly, σε αριθμητικές τιμές 0 και 1 αντίστοιχα, απαραίτητο για τον υπολογισμό του f1 score.

4. Προετοιμασία Δεδομένων:

- Επιλογή μόνο των στηλών src_bytes, dst_bytes και dst_host_srv_count ως χαρακτηριστικά (X) και την class ως ετικέτα (y).
- Αντικατάσταση των απουσιαζουσών τιμών με το μέσο όρο των αντίστοιχων χαρακτηριστικών.

5. Δημιουργία Dictionary για Αποθήκευση Αποτελεσμάτων:

- Δημιουργείται ένα dictionary για την αποθήκευση των αποτελεσμάτων κάθε αλγορίθμου σε κάθε εκτέλεση (run).

6. Εκτέλεση της Διαδικασίας 20 Φορές:

- Ο υπόλοιπος κώδικας εκτελείται 20 φορές με σκοπό τη συλλογή στατιστικών αποτελεσμάτων.

7. Διαχωρισμός Δεδομένων:

- Διαχωρισμός των δεδομένων σε training set (X_train_full, y_train_full) και test set (X_test, y_test).

- Το `test_size=0.2` σημαίνει ότι το 20% των δεδομένων θα χρησιμοποιηθεί ως test set.
- Δημιουργείται ένα validation set από το αρχικό training set.
- Το `test_size=0.1` σημαίνει ότι το 10% του training set θα χρησιμοποιηθεί ως validation set.

8. Ορισμός και Εκπαίδευση Μοντέλων:

- Ορίζεται μια λίστα με τα μοντέλα που θα χρησιμοποιηθούν.

9. Εκπαίδευση και Αξιολόγηση Μοντέλων:

➤ Για κάθε μοντέλο στη λίστα classifiers:

- Εκπαίδευση: Εκπαιδεύεται το μοντέλο με τα δεδομένα του training set.
- Αξιολόγηση με το Validation Set: Γίνονται προβλέψεις στο validation set και υπολογίζεται η ακρίβεια.
- Πρόβλεψη με το Test Set: Γίνονται προβλέψεις στο test set και υπολογίζεται η ακρίβεια και το F1 score. Τα αποτελέσματα αποθηκεύονται.

10. Εκτυπώνονται τα αποτελέσματα κάθε run καθώς και ο μέσος όρος των μετρικών.

9.4 Κώδικας μέτρησης χρόνου απόκρισης του μοντέλου χρήσης όλων των χαρακτηριστικών και του μοντέλου χρήσης των δύο κορυφαίων χαρακτηριστικών

Μέτρηση χρόνου του μοντέλου χρήσης όλων των χαρακτηριστικών

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
import time

# Διαβάζουμε τα δεδομένα από το CSV αρχείο
train_data = pd.read_csv("/content/Train_Data.csv")
```

```

# Χωρίζουμε τα χαρακτηριστικά (X) και την ετικέτα (y)
X = train_data.drop('class', axis=1) # Χαρακτηριστικά
y = train_data['class'] # Ετικέτα

# Χωρίζουμε τα δεδομένα σε training και validation set (10% για
validation)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1,
random_state=42)

# Χωρίζουμε τα αριθμητικά και μη αριθμητικά χαρακτηριστικά
numeric_cols = X_train.select_dtypes(include=['float64',
'int64']).columns
categorical_cols = X_train.select_dtypes(include=['object']).columns

# Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά με το
μέσο όρο
imputer_numeric = SimpleImputer(strategy='mean')
X_train_numeric =
pd.DataFrame(imputer_numeric.fit_transform(X_train[numeric_cols]),
columns=numeric_cols)
X_val_numeric =
pd.DataFrame(imputer_numeric.transform(X_val[numeric_cols]),
columns=numeric_cols)

# Κωδικοποίηση OneHot στα μη αριθμητικά χαρακτηριστικά
encoder = OneHotEncoder(sparse=False, handle_unknown='ignore') #
Χρησιμοποιούμε handle_unknown='ignore'
X_train_categorical_encoded =
encoder.fit_transform(X_train[categorical_cols])
X_val_categorical_encoded = encoder.transform(X_val[categorical_cols])

# Δημιουργία νέων ονομάτων χαρακτηριστικών για τα κωδικοποιημένα
κατηγορικά δεδομένα
new_feature_names = encoder.get_feature_names_out(categorical_cols)
X_train_categorical = pd.DataFrame(X_train_categorical_encoded,
columns=new_feature_names)
X_val_categorical = pd.DataFrame(X_val_categorical_encoded,
columns=new_feature_names)

# Συνδυασμός αριθμητικών και δυαδικών χαρακτηριστικών
X_train_final = pd.concat([X_train_numeric, X_train_categorical],
axis=1)
X_val_final = pd.concat([X_val_numeric, X_val_categorical], axis=1)

# Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά
columns_to_remove = [
    'num_outbound_cmds', 'is_host_login', 'num_root',

```

```

'srv_error_rate', 'srv_error_rate', 'dst_host_srv_error_rate',
'flag_S0', 'dst_host_same_srv_rate', 'is_guest_login',
'service_domain_u', 'flag_SF', 'service_http',
'protocol_type_udp', 'su_attempted', 'service_ftp',
'error_rate', 'dst_host_srv_error_rate', 'dst_host_error_rate,
error_rate'
]

# Εξασφαλίζουμε ότι οι στήλες που θέλουμε να αφαιρέσουμε υπάρχουν
columns_to_remove = [col for col in columns_to_remove if col in
X_train_final.columns]

X_train_final = X_train_final.drop(columns=columns_to_remove)
X_val_final = X_val_final.drop(columns=columns_to_remove)

# Εκπαίδευση μοντέλων και υπολογισμός της σημαντικότητας των
χαρακτηριστικών

# Μέτρηση χρόνου εκτέλεσης για το Decision Tree Classifier
start_time_dt = time.time()
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train_final, y_train)
dt_feature_importances = pd.Series(dt_classifier.feature_importances_,
index=X_train_final.columns)
end_time_dt = time.time()
execution_time_dt = end_time_dt - start_time_dt

# Μέτρηση χρόνου εκτέλεσης για το Random Forest Classifier
start_time_rf = time.time()
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train_final, y_train)
rf_feature_importances = pd.Series(rf_classifier.feature_importances_,
index=X_train_final.columns)
end_time_rf = time.time()
execution_time_rf = end_time_rf - start_time_rf

# Μέτρηση χρόνου εκτέλεσης για το Gradient Boosting Classifier
start_time_gb = time.time()
gb_classifier = GradientBoostingClassifier(random_state=42)
gb_classifier.fit(X_train_final, y_train)
gb_feature_importances = pd.Series(gb_classifier.feature_importances_,
index=X_train_final.columns)
end_time_gb = time.time()
execution_time_gb = end_time_gb - start_time_gb

# Μέτρηση χρόνου εκτέλεσης για το Extra Trees Classifier
start_time_et = time.time()

```

```

et_classifier = ExtraTreesClassifier(random_state=42)
et_classifier.fit(X_train_final, y_train)
et_feature_importances = pd.Series(et_classifier.feature_importances_,
index=X_train_final.columns)
end_time_et = time.time()
execution_time_et = end_time_et - start_time_et

# Υπολογισμός μέσου χρόνου εκπαίδευσης
mean_execution_time = (execution_time_dt + execution_time_rf +
execution_time_gb + execution_time_et) / 4

# Εκτύπωση του χρόνου εκτέλεσης και του μέσου χρόνου για κάθε μοντέλο
print(f"Execution time for Decision Tree Classifier:
{execution_time_dt:.4f} seconds")
print(f"Execution time for Random Forest Classifier:
{execution_time_rf:.4f} seconds")
print(f"Execution time for Gradient Boosting Classifier:
{execution_time_gb:.4f} seconds")
print(f"Execution time for Extra Trees Classifier:
{execution_time_et:.4f} seconds")
print(f"Mean execution time across all models:
{mean_execution_time:.4f} seconds")

# Εξέταση κοινών κορυφαίων χαρακτηριστικών ανάμεσα στα μοντέλα
dt_top_features =
dt_feature_importances.sort_values(ascending=False).index[:5]
rf_top_features =
rf_feature_importances.sort_values(ascending=False).index[:5]
gb_top_features =
gb_feature_importances.sort_values(ascending=False).index[:5]
et_top_features =
et_feature_importances.sort_values(ascending=False).index[:5]

common_features = set(dt_top_features).intersection(rf_top_features,
gb_top_features, et_top_features)
print("Common top features across models:", list(common_features))

```

1. Εισαγωγή Βιβλιοθηκών:

- ‘time’: Χρησιμοποιείται για την μέτρηση χρόνου εκτέλεσης.

2. Φόρτωση Δεδομένων

3. Διαχωρισμός Χαρακτηριστικών και Ετικετών

4. Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης

5. Επεξεργασία Αριθμητικών και Κατηγορικών Χαρακτηριστικών

6. Δημιουργία DataFrames με Κωδικοποιημένα Κατηγορικά Χαρακτηριστικά
7. Δημιουργία Νέων Ονομάτων Χαρακτηριστικών
8. Συνδυασμός Αριθμητικών και Κατηγορικών Χαρακτηριστικών
9. Αφαίρεση Συγκεκριμένων Στηλών
10. Εκπαίδευση Μοντέλων και Μέτρηση Χρόνου Εκτέλεσης:
 - Επαναλαμβάνεται η ίδια διαδικασία για όλα τα μοντέλα:
 - Μετράται ο χρόνο έναρξης.
 - Εκπαιδεύεται το μοντέλο.
 - Υπολογίζεται η σημαντικότητα των χαρακτηριστικών.
 - Μετράται ο χρόνος λήξης.
 - Υπολογίζεται ο χρόνος εκτέλεσης.
11. Υπολογισμός Μέσου Χρόνου Εκπαίδευσης:
 - Υπολογίζεται ο μέσος χρόνος εκπαίδευσης για όλα τα μοντέλα.
12. Εκτύπωση Χρόνου Εκτέλεσης και Κοινών Χαρακτηριστικών

Μέτρηση χρόνου του μοντέλου χρήσης των τριών χαρακτηριστικών

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
import time

# Διαβάζουμε τα δεδομένα από το CSV αρχείο
data = pd.read_csv("/content/Train_Data.csv")

# Επιλέγουμε μόνο τα χαρακτηριστικά 'src_bytes' 'dst_bytes' και
'dst_host_srv_count' και την ετικέτα
X = data[['src_bytes', 'dst_bytes', 'dst_host_srv_count']]
y = data['class']

# Αντιμετώπιση απουσιάζουσων τιμών με το μέσο όρο
X = X.fillna(X.mean())
```

```

# Διαχωρισμός των δεδομένων σε σύνολα εκπαίδευσης (train) και δοκιμής (test)
X_train_full, X_test, y_train_full, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Δημιουργία του validation set (10% του αρχικού training set)
X_train, X_val, y_train, y_val = train_test_split(X_train_full,
y_train_full, test_size=0.1, random_state=42)

# Οι αλγόριθμοι που θα χρησιμοποιηθούν
classifiers = [
    ("Decision Tree Classifier", DecisionTreeClassifier()),
    ("Random Forest Classifier", RandomForestClassifier()),
    ("Gradient Boosting Classifier", GradientBoostingClassifier()),
    ("Extra Trees Classifier", ExtraTreesClassifier())
]

# Αποθήκευση των χρόνων εκτέλεσης και ακρίβειας για κάθε μοντέλο
results = []

# Εκπαίδευση και αξιολόγηση κάθε μοντέλου
for clf_name, clf_model in classifiers:
    start_time = time.time() # Αρχή μέτρησης χρόνου

    # Εκπαίδευση του μοντέλου
    clf_model.fit(X_train, y_train)

    # Πρόβλεψη με το σύνολο δοκιμής
    y_pred = clf_model.predict(X_test)

    # Υπολογισμός της ακρίβειας
    accuracy = accuracy_score(y_test, y_pred)

    end_time = time.time() # Τέλος μέτρησης χρόνου
    execution_time = end_time - start_time # Υπολογισμός συνολικού
χρόνου εκτέλεσης

    # Αποθήκευση αποτελεσμάτων
    results.append((clf_name, accuracy, execution_time))

    # Εκτύπωση του αποτελέσματος
    print(f"Accuracy of {clf_name}: {accuracy:.4f}")
    print(f"Execution time of {clf_name}: {execution_time:.4f}
seconds\n")

# Υπολογισμός και εκτύπωση μέσου χρόνου εκπαίδευσης

```

```
mean_training_time = sum([result[2] for result in results]) /
len(results)
print(f"Mean training time across all classifiers:
{mean_training_time:.4f} seconds")
```

1. Εισαγωγή Βιβλιοθηκών
2. Φόρτωση Δεδομένων
3. Προετοιμασία Δεδομένων
4. Διαχωρισμός Δεδομένων
5. Ορισμός και Εκπαίδευση Μοντέλων
6. Εκπαίδευση, Αξιολόγηση και Μέτρηση Χρόνου Εκτέλεσης Μοντέλων:
 - Για κάθε αλγόριθμο:
 - Ξεκινά η μέτρηση χρόνου.
 - Εκπαιδεύεται το μοντέλο με το training set.
 - Γίνονται προβλέψεις με το test set.
 - Υπολογίζεται η ακρίβεια των προβλέψεων.
 - Τερματίζεται η μέτρηση χρόνου και υπολογίζεται ο χρόνος εκτέλεσης.
 - Αποθηκεύονται τα αποτελέσματα (όνομα αλγορίθμου, ακρίβεια, χρόνο εκτέλεσης) στη λίστα results.
 - Εκτυπώνεται η ακρίβεια και ο χρόνος εκτέλεσης για κάθε μοντέλο.
7. Υπολογισμός Μέσου Χρόνου Εκπαίδευσης

9.5 Κώδικες Παραμέτρων

Default παράμετροι των αλγορίθμων

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier

# Αρχικοποίηση ταξινομητών
dt_clf = DecisionTreeClassifier()
rf_clf = RandomForestClassifier()
gb_clf = GradientBoostingClassifier()
et_clf = ExtraTreesClassifier()
```

```
# Εκτύπωση των default παραμέτρων για τον καθένα
print("Default parameters for DecisionTreeClassifier:")
print(dt_clf.get_params(), "\n")

print("Default parameters for RandomForestClassifier:")
print(rf_clf.get_params(), "\n")

print("Default parameters for GradientBoostingClassifier:")
print(gb_clf.get_params(), "\n")

print("Default parameters for ExtraTreesClassifier:")
print(et_clf.get_params(), "\n")
```

1. Εισαγωγή Βιβλιοθηκών:

- Εισαγωγή των ταξινομητών (classifiers) από τη βιβλιοθήκη scikit-learn.

2. Αρχικοποίηση των Ταξινομητών:

- Δημιουργία αντικειμένων για κάθε έναν από τους ταξινομητές χρησιμοποιώντας τις προεπιλεγμένες παραμέτρους τους:
 - dt_clf: Ταξινομητής βασισμένος στο δέντρο απόφασης.
 - rf_clf: Ταξινομητής βασισμένος στο τυχαίο δάσος.
 - gb_clf: Ταξινομητής βασισμένος στη διαβάθμιση ενίσχυσης.
 - et_clf: Ταξινομητής βασισμένος στο εξαιρετικά τυχαίο δάσος.

3. Εκτύπωση των Προεπιλεγμένων Παραμέτρων για Κάθε Ταξινομητή:

- Χρησιμοποιείται η μέθοδος get_params() για να εκτυπώθούν οι προεπιλεγμένες παράμετροι για κάθε ταξινομητή. Η get_params() είναι μια ενσωματωμένη μέθοδος του scikit-learn, η οποία επιστρέφει ένα λεξικό (dictionary) με όλες τις παραμέτρους και τις προεπιλεγμένες τιμές τους.

Δοκιμές Αλλαγής Ορισμένων Παραμέτρων

Πρώτος κύκλος δοκιμών:

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
```

```

from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from itertools import combinations

# Διαβάζουμε τα δεδομένα από το CSV αρχείο
data = pd.read_csv("/content/Train_Data.csv")

# Χωρίζουμε τα χαρακτηριστικά (X) και την ετικέτα (y)
X = data.drop('class', axis=1) # Χαρακτηριστικά
y = data['class'] # Ετικέτα

# Χωρίζουμε τα αριθμητικά και μη αριθμητικά χαρακτηριστικά
numeric_cols = X.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = X.select_dtypes(include=['object']).columns

# Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά με το
μέσο όρο
imputer_numeric = SimpleImputer(strategy='mean')
X_numeric =
pd.DataFrame(imputer_numeric.fit_transform(X[numeric_cols]),
columns=numeric_cols)

# Κωδικοποίηση OneHot στα μη αριθμητικά χαρακτηριστικά
encoder = OneHotEncoder(sparse=False)
X_categorical_encoded = encoder.fit_transform(X[categorical_cols])
new_feature_names = encoder.get_feature_names_out(categorical_cols)
X_categorical = pd.DataFrame(X_categorical_encoded,
columns=new_feature_names)

# Συνδυασμός αριθμητικών και δυαδικών χαρακτηριστικών
X_final = pd.concat([X_numeric, X_categorical], axis=1)

# Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά
columns_to_remove = [
    'num_outbound_cmds', 'is_host_login', 'num_root',
    'srv_error_rate', 'srv_rerror_rate', 'dst_host_srv_error_rate',
    'flag_S0', 'dst_host_same_srv_rate', 'is_guest_login',
    'service_domain_u', 'flag_SF', 'service_http',
    'protocol_type_udp', 'su_attempted', 'service_ftp',
    'error_rate', 'dst_host_srv_rerror_rate', 'dst_host_rerror_rate',
    'rerror_rate'
]

# Εξασφαλίζουμε ότι οι στήλες που θέλουμε να αφαιρέσουμε υπάρχουν στο
τελικό dataset

```

```

columns_to_remove = [col for col in columns_to_remove if col in
X_final.columns]

# Αφαιρούμε τις στήλες από το X_final
X_final = X_final.drop(columns=columns_to_remove)

# Οι αλγόριθμοι που θα χρησιμοποιηθούν με αλλαγή παραμέτρων
classifiers_with_params = [
    ("Decision Tree Classifier", DecisionTreeClassifier(max_depth=5,
random_state=42)),
    ("Random Forest Classifier",
RandomForestClassifier(n_estimators=200, max_depth=10,
random_state=42)),
    ("Gradient Boosting Classifier",
GradientBoostingClassifier(learning_rate=0.05, n_estimators=200,
random_state=42)),
    ("Extra Trees Classifier", ExtraTreesClassifier(n_estimators=200,
max_features='log2', random_state=42))
]

# Δημιουργία όλων των πιθανών συνδυασμών τριάδων
for clf1, clf2, clf3 in combinations(classifiers_with_params, 3):
    clf1_name, clf1_model = clf1
    clf2_name, clf2_model = clf2
    clf3_name, clf3_model = clf3

    # Εκπαίδευση του πρώτου αλγορίθμου
    clf1_model.fit(X_final, y)
    clf1_feature_importances =
pd.Series(clf1_model.feature_importances_, index=X_final.columns)

    # Εκπαίδευση του δεύτερου αλγορίθμου
    clf2_model.fit(X_final, y)
    clf2_feature_importances =
pd.Series(clf2_model.feature_importances_, index=X_final.columns)

    # Εκπαίδευση του τρίτου αλγορίθμου
    clf3_model.fit(X_final, y)
    clf3_feature_importances =
pd.Series(clf3_model.feature_importances_, index=X_final.columns)

    # Εύρεση των κοινών χαρακτηριστικών
    common_features =
set(clf1_feature_importances.nlargest(5).index).intersection(
    clf2_feature_importances.nlargest(5).index,
    clf3_feature_importances.nlargest(5).index
)

```

```
# Εκτύπωση των κοινών χαρακτηριστικών
print(f"Common top features between {clf1_name}, {clf2_name} and {clf3_name}: {list(common_features)}")
```

Αλλαγές στις παραμέτρους:

- **Decision Tree Classifier:**
 - `max_depth=5`: Περιορίζεται το μέγιστο βάθος του δέντρου σε 5 επίπεδα. Αυτή η παράμετρος ελέγχει πόσο βαθύ μπορεί να είναι το δέντρο, δηλαδή πόσες διαδοχικές αποφάσεις μπορούν να ληφθούν.
- **Random Forest Classifier:**
 - `n_estimators=200`: Ορίζεται ο αριθμός των δέντρων στο δάσος σε 200.
 - `max_depth=10`: Περιορίζεται το μέγιστο βάθος κάθε δέντρου σε 10.
- **Gradient Boosting Classifier:**
 - `learning_rate=0.05`: Έτσι ελέγχεται ο ρυθμός με τον οποίο το μοντέλο μαθαίνει από τα δεδομένα. Χαμηλότερες τιμές κάνουν το μοντέλο να μαθαίνει πιο αργά αλλά και πιο σταθερά.
 - `n_estimators=200`: Καθορίζεται ο αριθμός των ενισχυτικών σταδίων που θα χρησιμοποιηθούν. Περισσότερα στάδια επιτρέπουν στο μοντέλο να μάθει καλύτερα τα δεδομένα, αν και μπορεί να αυξήσουν τον χρόνο εκπαίδευσης.
- **Extra Trees Classifier:**
 - `n_estimators=200`: Όπως και στον Random Forest, αυξάνεται έτσι ο αριθμός των δέντρων στο μοντέλο σε 200.
 - `max_features='log2'`: Καθορίζεται ο μέγιστος αριθμός χαρακτηριστικών που θα χρησιμοποιούνται για τον υπολογισμό της καλύτερης διαίρεσης σε κάθε κόμβο, χρησιμοποιώντας τη συνάρτηση `log2`.

Δεύτερος κύκλος δοκιμών:

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
```

```

from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from itertools import combinations

# Διαβάζουμε τα δεδομένα από το CSV αρχείο
data = pd.read_csv("/content/Train_Data.csv")

# Χωρίζουμε τα χαρακτηριστικά (X) και την ετικέτα (y)
X = data.drop('class', axis=1) # Χαρακτηριστικά
y = data['class'] # Ετικέτα

# Χωρίζουμε τα αριθμητικά και μη αριθμητικά χαρακτηριστικά
numeric_cols = X.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = X.select_dtypes(include=['object']).columns

# Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά με το
μέσο όρο
imputer_numeric = SimpleImputer(strategy='mean')
X_numeric =
pd.DataFrame(imputer_numeric.fit_transform(X[numeric_cols]),
columns=numeric_cols)

# Κωδικοποίηση OneHot στα μη αριθμητικά χαρακτηριστικά
encoder = OneHotEncoder(sparse=False)
X_categorical_encoded = encoder.fit_transform(X[categorical_cols])
new_feature_names = encoder.get_feature_names_out(categorical_cols)
X_categorical = pd.DataFrame(X_categorical_encoded,
columns=new_feature_names)

# Συνδυασμός αριθμητικών και δυαδικών χαρακτηριστικών
X_final = pd.concat([X_numeric, X_categorical], axis=1)

# Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά
columns_to_remove = [
    'num_outbound_cmds', 'is_host_login', 'num_root',
    'srv_error_rate', 'srv_rerror_rate', 'dst_host_srv_error_rate',
    'flag_S0', 'dst_host_same_srv_rate', 'is_guest_login',
    'service_domain_u', 'flag_SF', 'service_http',
    'protocol_type_udp', 'su_attempted', 'service_ftp',
    'error_rate', 'dst_host_srv_rerror_rate', 'dst_host_rerror_rate',
    'rerror_rate'
]

# Εξασφαλίζουμε ότι οι στήλες που θέλουμε να αφαιρέσουμε υπάρχουν στο
τελικό dataset

```

```

columns_to_remove = [col for col in columns_to_remove if col in
X_final.columns]

# Αφαιρούμε τις στήλες από το X_final
X_final = X_final.drop(columns=columns_to_remove)

# Ορίζουμε τους αλγόριθμους με τις τροποποιημένες παραμέτρους
classifiers = [
    ("Decision Tree Classifier",
DecisionTreeClassifier(min_samples_split=5, min_samples_leaf=5,
random_state=42)),
    ("Random Forest Classifier",
RandomForestClassifier(min_samples_split=5, min_samples_leaf=5,
random_state=42)),
    ("Gradient Boosting Classifier",
GradientBoostingClassifier(subsample=0.8, max_features=0.5,
random_state=42)),
    ("Extra Trees Classifier",
ExtraTreesClassifier(min_samples_split=5, min_samples_leaf=5,
random_state=42))
]

# Δημιουργία όλων των πιθανών συνδυασμών τριάδων
for clf1, clf2, clf3 in combinations(classifiers, 3):
    clf1_name, clf1_model = clf1
    clf2_name, clf2_model = clf2
    clf3_name, clf3_model = clf3

    # Εκπαίδευση του πρώτου αλγορίθμου
    clf1_model.fit(X_final, y)
    clf1_feature_importances =
pd.Series(clf1_model.feature_importances_, index=X_final.columns)

    # Εκπαίδευση του δεύτερου αλγορίθμου
    clf2_model.fit(X_final, y)
    clf2_feature_importances =
pd.Series(clf2_model.feature_importances_, index=X_final.columns)

    # Εκπαίδευση του τρίτου αλγορίθμου
    clf3_model.fit(X_final, y)
    clf3_feature_importances =
pd.Series(clf3_model.feature_importances_, index=X_final.columns)

    # Εύρεση των κοινών χαρακτηριστικών
    common_features =
set(clf1_feature_importances.nlargest(5).index).intersection(
    clf2_feature_importances.nlargest(5).index,

```

```

        clf3_feature_importances.nlargest(5).index
    )

    # Εκτύπωση των κοινών χαρακτηριστικών
    print(f"Common top features between {clf1_name}, {clf2_name} and {clf3_name}: {list(common_features)}")

```

Αλλαγές στις παραμέτρους:

○ **Decision Tree Classifier:**

- `min_samples_split=5`: Αυτή η παράμετρος καθορίζει τον ελάχιστο αριθμό δειγμάτων που πρέπει να υπάρχουν σε ένα κόμβο για να μπορεί αυτός ο κόμβος να σπάσει σε δύο άλλους κόμβους. Αυξάνοντας αυτήν την τιμή από την προεπιλογή (2), το μοντέλο γίνεται λιγότερο ευαίσθητο σε μικρές παραλλαγές των δεδομένων.
- `min_samples_leaf=5`: Ορίζεται ο ελάχιστος αριθμός δειγμάτων που πρέπει να υπάρχουν σε ένα φύλλο του δέντρου.

○ **Random Forest Classifier:**

- `min_samples_split=5`
- `min_samples_leaf=5`

○ **Gradient Boosting Classifier:**

- `subsample=0.8`: Καθορίζεται το ποσοστό των δειγμάτων εκπαίδευσης που χρησιμοποιούνται για την εκπαίδευση κάθε ενισχυτικού μοντέλου (booster). Το 0.8 δηλώνει ότι μόνο το 80% των δειγμάτων χρησιμοποιείται σε κάθε βήμα εκπαίδευσης.
- `max_features=0.5`: Καθορίζεται ο αριθμός των χαρακτηριστικών που θα χρησιμοποιούνται για την εύρεση της καλύτερης διαίρεσης σε κάθε κόμβο. Το 0.5 δηλώνει ότι το 50% των διαθέσιμων χαρακτηριστικών θα χρησιμοποιούνται σε κάθε ενισχυτικό μοντέλο.

○ **Extra Trees Classifier:**

- `min_samples_split=5`
- `min_samples_leaf=5`

Τρίτος κύκλος δοκιμών:

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from itertools import combinations

# Διαβάζουμε τα δεδομένα από το CSV αρχείο
data = pd.read_csv("/content/Train_Data.csv")

# Χωρίζουμε τα χαρακτηριστικά (X) και την ετικέτα (y)
X = data.drop('class', axis=1) # Χαρακτηριστικά
y = data['class'] # Ετικέτα

# Χωρίζουμε τα αριθμητικά και μη αριθμητικά χαρακτηριστικά
numeric_cols = X.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = X.select_dtypes(include=['object']).columns

# Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά με το μέσο όρο
imputer_numeric = SimpleImputer(strategy='mean')
X_numeric =
pd.DataFrame(imputer_numeric.fit_transform(X[numeric_cols]),
columns=numeric_cols)

# Κωδικοποίηση OneHot στα μη αριθμητικά χαρακτηριστικά
encoder = OneHotEncoder(sparse=False)
X_categorical_encoded = encoder.fit_transform(X[categorical_cols])
new_feature_names = encoder.get_feature_names_out(categorical_cols)
X_categorical = pd.DataFrame(X_categorical_encoded,
columns=new_feature_names)

# Συνδυασμός αριθμητικών και δυαδικών χαρακτηριστικών
X_final = pd.concat([X_numeric, X_categorical], axis=1)

# Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά
columns_to_remove = [
    'num_outbound_cmds', 'is_host_login', 'num_root',
    'srv_error_rate', 'srv_rerror_rate', 'dst_host_srv_error_rate',
    'flag_S0', 'dst_host_same_srv_rate', 'is_guest_login',
    'service_domain_u', 'flag_SF', 'service_http',
    'protocol_type_udp', 'su_attempted', 'service_ftp',
    'error_rate', 'dst_host_srv_rerror_rate', 'dst_host_rerror_rate',
    'rerror_rate'
```

```

]

# Εξασφαλίζουμε ότι οι στήλες που θέλουμε να αφαιρέσουμε υπάρχουν στο
τελικό dataset
columns_to_remove = [col for col in columns_to_remove if col in
X_final.columns]

# Αφαιρούμε τις στήλες από το X_final
X_final = X_final.drop(columns=columns_to_remove)

# Ορίζουμε τους αλγόριθμους με διαφορετικές παραμέτρους
classifiers = [
    ("Decision Tree Classifier", DecisionTreeClassifier(max_depth=7,
criterion='entropy', random_state=42)),
    ("Random Forest Classifier",
RandomForestClassifier(n_estimators=150, min_samples_split=10,
random_state=42)),
    ("Gradient Boosting Classifier",
GradientBoostingClassifier(learning_rate=0.1, n_estimators=100,
max_depth=3, random_state=42)),
    ("Extra Trees Classifier", ExtraTreesClassifier(n_estimators=100,
max_features='sqrt', min_samples_leaf=10, random_state=42))
]

# Δημιουργία όλων των πιθανών συνδυασμών τριάδων
for clf1, clf2, clf3 in combinations(classifiers, 3):
    clf1_name, clf1_model = clf1
    clf2_name, clf2_model = clf2
    clf3_name, clf3_model = clf3

    # Εκπαίδευση του πρώτου αλγορίθμου
    clf1_model.fit(X_final, y)
    clf1_feature_importances =
pd.Series(clf1_model.feature_importances_, index=X_final.columns)

    # Εκπαίδευση του δεύτερου αλγορίθμου
    clf2_model.fit(X_final, y)
    clf2_feature_importances =
pd.Series(clf2_model.feature_importances_, index=X_final.columns)

    # Εκπαίδευση του τρίτου αλγορίθμου
    clf3_model.fit(X_final, y)
    clf3_feature_importances =
pd.Series(clf3_model.feature_importances_, index=X_final.columns)

    # Εύρεση των κοινών χαρακτηριστικών

```

```

common_features =
set(clf1_feature_importances.nlargest(5).index).intersection(
    clf2_feature_importances.nlargest(5).index,
    clf3_feature_importances.nlargest(5).index
)

# Εκτύπωση των κοινών χαρακτηριστικών
print(f"Common top features between {clf1_name}, {clf2_name} and
{clf3_name}: {list(common_features)}")

print("Model training and feature importance analysis are complete.")

```

Αλλαγές στις παραμέτρους:

- **Decision Tree Classifier:**
 - `max_depth=7`: Ορίζεται έτσι το μέγιστο βάθος του δέντρου απόφασης. Ένας περιορισμός στο βάθος μπορεί να αποτρέψει την υπερεκπαίδευση (overfitting) του μοντέλου.
 - `criterion='entropy'`: Καθορίζεται ο τρόπος υπολογισμού της ποιότητας των διαχωρισμών στο δέντρο. (Η εντροπία μετρά την αταξία και χρησιμοποιείται για να μεγιστοποιήσει την καθαρότητα των υποομάδων.)
- **Random Forest Classifier:**
 - `n_estimators=150`
 - `min_samples_split=10`
- **Gradient Boosting Classifier:**
 - `learning_rate=0.1`
 - `n_estimators=100`
 - `max_depth=3`
- **Extra Trees Classifier:**
 - `n_estimators=100`
 - `max_features='sqrt'`: Ορίζεται έτσι ο αριθμός των χαρακτηριστικών που θα εξετάζονται κατά τον διαχωρισμό κάθε κόμβου. Το 'sqrt' σημαίνει ότι θα χρησιμοποιείται η τετραγωνική ρίζα του συνολικού αριθμού των χαρακτηριστικών.
 - `min_samples_leaf=10`

Δοκιμασία αλλαγής παραμέτρων και εμφάνισης αποτελεσμάτων ξεχωριστά για κάθε αλγόριθμο:

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from itertools import combinations

# Διαβάζουμε τα δεδομένα από το CSV αρχείο
data = pd.read_csv("/content/Train_Data.csv")

# Χωρίζουμε τα χαρακτηριστικά (X) και την ετικέτα (y)
X = data.drop('class', axis=1) # Χαρακτηριστικά
y = data['class'] # Ετικέτα

# Χωρίζουμε τα αριθμητικά και μη αριθμητικά χαρακτηριστικά
numeric_cols = X.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = X.select_dtypes(include=['object']).columns

# Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά με το μέσο όρο
imputer_numeric = SimpleImputer(strategy='mean')
X_numeric =
pd.DataFrame(imputer_numeric.fit_transform(X[numeric_cols]),
columns=numeric_cols)

# Κωδικοποίηση OneHot στα μη αριθμητικά χαρακτηριστικά
encoder = OneHotEncoder(sparse=False)
X_categorical_encoded = encoder.fit_transform(X[categorical_cols])
new_feature_names = encoder.get_feature_names_out(categorical_cols)
X_categorical = pd.DataFrame(X_categorical_encoded,
columns=new_feature_names)

# Συνδυασμός αριθμητικών και δυαδικών χαρακτηριστικών
X_final = pd.concat([X_numeric, X_categorical], axis=1)

# Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά
columns_to_remove = [
    'num_outbound_cmds', 'is_host_login', 'num_root',
    'srv_error_rate', 'srv_rerror_rate', 'dst_host_srv_error_rate',
    'flag_S0', 'dst_host_same_srv_rate', 'is_guest_login',
    'service_domain_u', 'flag_SF', 'service_http',
    'protocol_type_udp', 'su_attempted', 'service_ftp',
```

```

    'error_rate', 'dst_host_srv_error_rate', 'dst_host_rerror_rate',
    'error_rate'
]

# Εξασφαλίζουμε ότι οι στήλες που θέλουμε να αφαιρέσουμε υπάρχουν στο
τελικό dataset
columns_to_remove = [col for col in columns_to_remove if col in
X_final.columns]

# Αφαιρούμε τις στήλες από το X_final
X_final = X_final.drop(columns=columns_to_remove)

# Ορίζουμε τους αλγόριθμους με διαφορετικές παραμέτρους
parameter_sets = {
    "Decision Tree Classifier": [
        {"min_samples_split": 2, "min_samples_leaf": 1},
        {"min_samples_split": 5, "min_samples_leaf": 5},
        {"max_depth": None},
        {"max_depth": 7}
    ],
    "Random Forest Classifier": [
        {"min_samples_split": 2, "min_samples_leaf": 1},
        {"min_samples_split": 5, "min_samples_leaf": 5},
        {"max_depth": None},
        {"max_depth": 7}
    ],
    "Gradient Boosting Classifier": [
        {"min_samples_split": 2, "min_samples_leaf": 1},
        {"min_samples_split": 5, "min_samples_leaf": 5},
        {"max_depth": 3},
        {"max_depth": 7}
    ],
    "Extra Trees Classifier": [
        {"min_samples_split": 2, "min_samples_leaf": 1},
        {"min_samples_split": 5, "min_samples_leaf": 5},
        {"max_depth": None},
        {"max_depth": 7}
    ]
}

# Εκτέλεση των πειραμάτων
results = []

for clf_name, params_list in parameter_sets.items():
    for params in params_list:
        # Δημιουργία και εκπαίδευση του μοντέλου με τις τρέχουσες
        παραμέτρους

```

```

if clf_name == "Decision Tree Classifier":
    clf = DecisionTreeClassifier(**params)
elif clf_name == "Random Forest Classifier":
    clf = RandomForestClassifier(**params)
elif clf_name == "Gradient Boosting Classifier":
    clf = GradientBoostingClassifier(**params)
elif clf_name == "Extra Trees Classifier":
    clf = ExtraTreesClassifier(**params)

clf.fit(X_final, y)
feature_importances = pd.Series(clf.feature_importances_,
index=X_final.columns)

# Εύρεση κορυφαίων χαρακτηριστικών
top_features = feature_importances.nlargest(5).index.tolist()
results.append({
    "Classifier": clf_name,
    "Parameters": params,
    "Top Features": top_features
})

# Εκτύπωση αποτελεσμάτων
for result in results:
    print(f"Classifier: {result['Classifier']}")
    print(f"Parameters: {result['Parameters']}")
    print(f"Top Features: {result['Top Features']}")
    print("\n")

```

Αλλαγές στις παραμέτρους:

Σε όλους του αλγορίθμους δοκιμάστηκαν τρεις αλλαγές: min_samples_split=5, min_samples_leaf=5 και max_depth= 7.

Οπτικοποίηση εκπαίδευσης Decision Tree και αναλυτικά αποτελέσματα χαρακτηριστικών

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree

```

```

import matplotlib.pyplot as plt

# Διαβάζουμε τα δεδομένα από το CSV αρχείο
train_data = pd.read_csv("/content/Train_Data.csv")

# Χωρίζουμε τα χαρακτηριστικά (X) και την ετικέτα (y)
X = train_data.drop('class', axis=1) # Χαρακτηριστικά
y = train_data['class'] # Ετικέτα

# Χωρίζουμε τα δεδομένα σε training και validation set (10% για validation)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1,
random_state=42)

# Χωρίζουμε τα αριθμητικά και μη αριθμητικά χαρακτηριστικά
numeric_cols = X_train.select_dtypes(include=['float64',
'int64']).columns
categorical_cols = X_train.select_dtypes(include=['object']).columns

# Αντιμετώπιση απουσιάζουσων τιμών στα αριθμητικά χαρακτηριστικά με το μέσο όρο
imputer_numeric = SimpleImputer(strategy='mean')
X_train_numeric =
pd.DataFrame(imputer_numeric.fit_transform(X_train[numeric_cols]),
columns=numeric_cols)
X_val_numeric =
pd.DataFrame(imputer_numeric.transform(X_val[numeric_cols]),
columns=numeric_cols)

# Κωδικοποίηση OneHot στα μη αριθμητικά χαρακτηριστικά
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
X_train_categorical_encoded =
encoder.fit_transform(X_train[categorical_cols])
X_val_categorical_encoded = encoder.transform(X_val[categorical_cols])

# Δημιουργία νέων ονομάτων χαρακτηριστικών για τα κωδικοποιημένα
κατηγορικά δεδομένα
new_feature_names = encoder.get_feature_names_out(categorical_cols)
X_train_categorical = pd.DataFrame(X_train_categorical_encoded,
columns=new_feature_names)
X_val_categorical = pd.DataFrame(X_val_categorical_encoded,
columns=new_feature_names)

# Συνδυασμός αριθμητικών και δυαδικών χαρακτηριστικών
X_train_final = pd.concat([X_train_numeric, X_train_categorical],
axis=1)
X_val_final = pd.concat([X_val_numeric, X_val_categorical], axis=1)

```

```

# Αφαίρεση συγκεκριμένων στηλών από τα τελικά χαρακτηριστικά
columns_to_remove = [
    'num_outbound_cmds', 'is_host_login', 'num_root',
    'srv_error_rate', 'srv_rerror_rate', 'dst_host_srv_error_rate',
    'flag_S0', 'dst_host_same_srv_rate', 'is_guest_login',
    'service_domain_u', 'flag_SF', 'service_http',
    'protocol_type_udp', 'su_attempted', 'service_ftp',
    'error_rate', 'dst_host_srv_rerror_rate', 'dst_host_rerror_rate',
    'rerror_rate'
]

# Εξασφαλίζουμε ότι οι στήλες που θέλουμε να αφαιρέσουμε υπάρχουν
columns_to_remove = [col for col in columns_to_remove if col in
X_train_final.columns]

X_train_final = X_train_final.drop(columns=columns_to_remove)
X_val_final = X_val_final.drop(columns=columns_to_remove)

# Εκπαίδευση Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train_final, y_train)

# Οπτικοποίηση του Decision Tree
plt.figure(figsize=(20, 10)) # Ρύθμιση μεγέθους εικόνας
plot_tree(dt_classifier,
           feature_names=X_train_final.columns, # Χρησιμοποιούμε τα
ονόματα των χαρακτηριστικών
           class_names=y_train.unique(), # Χρησιμοποιούμε απευθείας τις
κατηγορίες
           filled=True, # Χρώμα στους κόμβους
           rounded=True) # Στρογγυλεμένοι κόμβοι
plt.show()

# Εκτύπωση των αποφάσεων με τη σειρά
def print_decision_steps(tree, feature_names):
    """Εκτυπώνει τις αποφάσεις του δέντρου με τη σειρά που εμφανίζονται
    στους κόμβους"""
    left = tree.tree_.children_left
    right = tree.tree_.children_right
    threshold = tree.tree_.threshold
    features = [feature_names[i] if i != -2 else "leaf" for i in
tree.tree_.feature]

    n_nodes = tree.tree_.node_count
    decisions = []

```

```

# Διατρέχουμε τους κόμβους
for node in range(n_nodes):
    if left[node] != -1: # Αν ο κόμβος δεν είναι φύλλο
        decisions.append(f"Node {node}: Split on feature
'features[{node}]' ")

# Εκτύπωση των αποφάσεων
for decision in decisions:
    print(decision)

# Καλούμε τη συνάρτηση για να εκτυπώσουμε τις αποφάσεις με τη σειρά
print_decision_steps(dt_classifier, X_train_final.columns)

```

1. Εισαγωγή Βιβλιοθηκών:

2. Φόρτωση Δεδομένων

3. Διαχωρισμός Χαρακτηριστικών και Ετικετών

4. Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης

5. Επεξεργασία Αριθμητικών και Κατηγορικών Χαρακτηριστικών

6. Δημιουργία DataFrames με Κωδικοποιημένα Κατηγορικά Χαρακτηριστικά

7. Δημιουργία Νέων Ονομάτων Χαρακτηριστικών

8. Συνδυασμός Αριθμητικών και Κατηγορικών Χαρακτηριστικών

9. Εκπαίδευση Decision Tree

10. Οπτικοποίηση του Decision Tree:

- `plot_tree()` για να οπτικοποιήσεις το Decision Tree, με χρώμα στους κόμβους που αναπαριστά τις αποφάσεις του δέντρου.

11. Εκτύπωση των αποφάσεων με τη σειρά:

- `print_decision_steps()`: Εκτυπώνει τις αποφάσεις σε κάθε κόμβο του δέντρου.
- `tree.tree_.children_left`: Αριστερά παιδιά κάθε κόμβου.
- `tree.tree_.children_right`: Δεξιά παιδιά κάθε κόμβου.
- `tree.tree_.threshold`: Το κατώφλι απόφασης για κάθε κόμβο.
- `features`: Λίστα με τα χαρακτηριστικά που χρησιμοποιούνται για τη διαίρεση (split) σε κάθε κόμβο.
- Διατρέχονται οι κόμβοι και εκτυπώνονται τα χαρακτηριστικά που χρησιμοποιούνται για κάθε split.

Αποτελέσματα επιλογής χαρακτηριστικών:

Node 0: Split on feature 'src_bytes'
Node 1: Split on feature 'service_http'
Node 2: Split on feature 'dst_bytes'
Node 3: Split on feature 'logged_in'
Node 4: Split on feature 'flag_S1'
Node 5: Split on feature 'service_smtp'
Node 6: Split on feature 'count'
Node 7: Split on feature 'service_ftp_data'
Node 8: Split on feature 'service_finger'
Node 9: Split on feature 'flag_RST0'
Node 10: Split on feature 'service_auth'
Node 11: Split on feature 'service_X11'
Node 12: Split on feature 'service_time'
Node 13: Split on feature 'service_telnet'
Node 14: Split on feature 'service_other'
Node 15: Split on feature 'service_ecr_i'
Node 17: Split on feature 'dst_host_count'
Node 20: Split on feature 'flag_REJ'
Node 23: Split on feature 'dst_host_count'
Node 26: Split on feature 'dst_host_same_srv_rate'
Node 29: Split on feature 'dst_host_same_srv_rate'
Node 34: Split on feature 'src_bytes'
Node 35: Split on feature 'flag_RST0'
Node 36: Split on feature 'srv_diff_host_rate'
Node 37: Split on feature 'count'
Node 39: Split on feature 'srv_count'
Node 45: Split on feature 'flag_SF'
Node 46: Split on feature 'srv_diff_host_rate'
Node 51: Split on feature 'count'
Node 52: Split on feature 'serror_rate'
Node 53: Split on feature 'dst_host_rerror_rate'
Node 57: Split on feature 'diff_srv_rate'
Node 61: Split on feature 'count'
Node 64: Split on feature 'dst_host_rerror_rate'
Node 65: Split on feature 'dst_bytes'
Node 66: Split on feature 'service_ftp'
Node 67: Split on feature 'hot'
Node 68: Split on feature 'service_smtp'
Node 69: Split on feature 'rerror_rate'
Node 71: Split on feature 'logged_in'
Node 77: Split on feature 'dst_host_same_srv_rate'
Node 78: Split on feature 'dst_bytes'
Node 82: Split on feature 'dst_host_diff_srv_rate'
Node 85: Split on feature 'dst_host_same_srv_rate'
Node 86: Split on feature 'dst_host_count'
Node 89: Split on feature 'dst_host_srv_serror_rate'
Node 92: Split on feature 'service_ecr_i'
Node 93: Split on feature 'hot'
Node 94: Split on feature 'dst_host_same_src_port_rate'
Node 95: Split on feature 'service_private'
Node 96: Split on feature 'service_pm_dump'
Node 97: Split on feature 'service_imap4'
Node 98: Split on feature 'flag_RSTOS0'
Node 99: Split on feature 'is_guest_login'
Node 100: Split on feature 'dst_host_rerror_rate'

Node 101: Split on feature 'rerror_rate'
Node 102: Split on feature 'flag_S2'
Node 103: Split on feature 'dst_host_srv_serror_rate'
Node 104: Split on feature 'num_compromised'
Node 105: Split on feature 'dst_host_diff_srv_rate'
Node 106: Split on feature 'dst_host_same_src_port_rate'
Node 108: Split on feature 'dst_host_same_src_port_rate'
Node 109: Split on feature 'src_bytes'
Node 110: Split on feature 'dst_host_same_srv_rate'
Node 115: Split on feature 'protocol_type_icmp'
Node 116: Split on feature 'dst_host_rerror_rate'
Node 117: Split on feature 'dst_host_srv_diff_host_rate'
Node 118: Split on feature 'dst_host_srv_serror_rate'
Node 120: Split on feature 'dst_host_count'
Node 123: Split on feature 'dst_host_srv_count'
Node 126: Split on feature 'dst_host_srv_count'
Node 129: Split on feature 'dst_host_srv_count'
Node 132: Split on feature 'num_compromised'
Node 135: Split on feature 'dst_host_same_src_port_rate'
Node 137: Split on feature 'dst_host_count'
Node 140: Split on feature 'src_bytes'
Node 143: Split on feature 'src_bytes'
Node 144: Split on feature 'same_srv_rate'
Node 148: Split on feature 'service_telnet'
Node 151: Split on feature 'dst_host_count'
Node 157: Split on feature 'dst_host_same_src_port_rate'
Node 158: Split on feature 'src_bytes'
Node 161: Split on feature 'dst_bytes'
Node 164: Split on feature 'service_ftp_data'
Node 165: Split on feature 'service_private'
Node 166: Split on feature 'service_tim_i'
Node 167: Split on feature 'dst_host_srv_count'
Node 168: Split on feature 'src_bytes'
Node 170: Split on feature 'dst_bytes'
Node 175: Split on feature 'src_bytes'
Node 178: Split on feature 'dst_host_srv_diff_host_rate'
Node 179: Split on feature 'src_bytes'
Node 180: Split on feature 'src_bytes'
Node 183: Split on feature 'duration'
Node 186: Split on feature 'num_root'
Node 189: Split on feature 'src_bytes'
Node 190: Split on feature 'duration'
Node 191: Split on feature 'dst_host_srv_count'
Node 192: Split on feature 'dst_bytes'
Node 195: Split on feature 'num_access_files'
Node 198: Split on feature 'duration'
Node 199: Split on feature 'src_bytes'
Node 202: Split on feature 'dst_host_diff_srv_rate'
Node 203: Split on feature 'dst_bytes'
Node 204: Split on feature 'hot'
Node 210: Split on feature 'src_bytes'

10 ΒΙΒΛΙΟΓΡΑΦΙΑ- ΑΝΑΦΟΡΕΣ

Ελληνική Βιβλιογραφία:

- 1 Τασουλής, Σ. (χ.χ.). Διαλέξεις μαθήματος 'Βαθιά Μάθηση'. Πανεπιστήμιο Θεσσαλίας.

Ξενόγλωσση Βιβλιογραφία:

- 2 Delua, J. (2021, Μαρτίου 12). *Supervised versus unsupervised learning: What's the difference?* IBM. Ανακτήθηκε από <https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning>.
- 3 Gormont, N. Z., Selamat, A., & Krejcar, O. (2021, Νοεμβρίου 16). *A recent research on malware detection using machine learning algorithm: Current challenges and future works*. Ανακτήθηκε από https://link.springer.com/chapter/10.1007/978-3-030-90235-3_41.
- 4 Gormont, N. Z., Selamat, A., Cheng, L. K., et al. (2023, Μαρτίου 14). *Machine learning algorithm for malware detection: Taxonomy, current challenges, and future directions*. IEEE. Ανακτήθηκε από <https://ieeexplore.ieee.org/abstract/document/10068497>.
- 5 Harris, B., & Hunt, R. (1999, Ιουνίου). *TCP/IP security threats and attack methods*. Ανακτήθηκε από <https://www.sciencedirect.com/science/article/abs/pii/S014036649900064X>.
- 6 Imperva. (χ.χ.). *User Datagram Protocol (UDP)*. Ανακτήθηκε από <https://www.imperva.com/learn/ddos/udp-user-datagram-protocol/>.
- 7 Kaggle. (χ.χ.). *Network intrusion detection [Dataset]*. Ανακτήθηκε από <https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection/data>.
- 8 Li, Y., & Liu, Q. (2021, Νοεμβρίου 7). *A comprehensive review study of cyber-attacks and cyber security: Emerging trends and recent developments*. Ανακτήθηκε από <https://www.sciencedirect.com/science/article/pii/S2352484721007289>.
- 9 Okta. (χ.χ.). *ICMP Protocol (Internet Control Message Protocol): A Guide*. Ανακτήθηκε από <https://www.okta.com/identity-101/icmp/>.
- 10 Scikit-learn. (χ.χ.). *Decision Trees*. Ανακτήθηκε από <https://scikit-learn.org/stable/modules/tree.html>.
- 11 Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., & Lin, W.-Y. (2009, Δεκεμβρίου). *Intrusion detection by machine learning: A review*. Ανακτήθηκε από https://www.sciencedirect.com/science/article/abs/pii/S0957417409004801?casa_token=s4A3EEhX_2gAAAAA
- 12 Wikipedia. (χ.χ.). *Ασφάλεια δικτύων υπολογιστών*. Ανακτήθηκε από https://el.wikipedia.org/wiki/Ασφάλεια_δικτύων_υπολογιστών.
- 13 Coursera. (χ.χ.). *Σεμινάριο Ασφάλεια Δικτύων*.