
# ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
**ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ**

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΑΣΦΑΛΕΙΑ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ


# Δημιουργία ψευδούς κίνησης δικτύου για συγκάλυψη επιθέσεων με χρήση μηχανικής μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του


Ευστάθιου Κοκολάκη

**Επιβλέπων:** Γεώργιος Στεργιόπουλος

**Μέλη εξεταστικής επιτροπής:**



Σάμος, Μάρτιος 2022

Η σελίδα αυτή είναι σκόπιμα λευκή.

# Πρόλογος και ευχαριστίες

Θα ήθελα να ευχαριστήσω αρχικά τους γονείς μου που με βοηθήσανε με τον τρόπους τους αυτήν την δύσκολη για όλους περίοδο με τις συμβουλές τους. Επίσης θα ήθελα να ευχαριστήσω τους φίλους μου για τις συμβουλές τους και την υποστήριξή τους.

Τέλος, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του μεταπτυχιακού οι οποίοι, ο καθένας με τον τρόπο του, ενίσχυσαν το πάθος μας για το αντικείμενο. Κυρίως όμως, ευχαριστώ τον επιβλέποντα καθηγητή μου, Δρ. Στεργιόπουλο Γεώργιο, τόσο για την εμπιστοσύνη που μου έδειξε, με την ανάθεση του συγκεκριμένου θέματος, όσο και για την ενδελεχή υποστήριξή και καθοδήγησή του καθ' όλη την διάρκεια εκπόνησης της Διπλωματικής μου εργασίας.


ΕΥΣΤΑΘΙΟΥ ΚΟΚΟΛΑΚΗ
Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

# Πίνακας Περιεχομένων

**Λίστα Εικόνων**

# Περίληψη

Η μετάβαση στην εποχή της Πληροφορίας την έχει καταστήσει σε ένα από τα σημαντικότερα αγαθά μιας σύγχρονης κοινωνίας. Ως φυσικό επακόλουθο, το αγαθό αυτό βρίσκεται σε συνεχή απειλή από κυβερνοεγκληματίες, κακόβουλες οντότητες που σκοπό έχουν το κέρδος ή την πρόκληση ζημιάς στους πληροφοριακούς πόρους ενός οργανισμού ή ενός κυβερνητικού φορέα. Μία από τις σημαντικές στρατηγικές αντιμετώπισης κυβερνοεπιθέσεων, είναι η συνεχής δοκιμή των συστημάτων εκείνων που αποσκοπούν στην αναγνώριση και εξουδετέρωση απειλών. Η Κυβερνοασφάλεια, είναι ένα υποπεδίο της Επιστήμης της Πληροφορικής που μελετά και σχεδιάζει τέτοιες στρατηγικές και μεθοδολογίες αντιμετώπισης κυβερνοεπιθέσεων.

Ένα σημαντικό εργαλείο που χρησιμοποιείται πλέον πολύ πιο συχνά σε πολλά επιστημονικά πεδία είναι η Μηχανική Μάθηση-Μ.Μ.. Με τις σύγχρονες επιθέσεις να γίνονται όλο και πιο πολύπλοκες, αφήνοντας και ένα μεγαλύτερο ψηφιακό αποτύπωμα, οι ερευνητές ασφαλείας έχουν αρχίσει και εκμεταλλεύονται τις δυνατότητές της Μ.Μ. ώστε να αναγνωρίσουν και να εξουδετερώσουν τέτοιες επιθέσεις.

Στην παρούσα εργασία, θα αναπτύξουμε ένα μοντέλο Μηχανικής Μάθησης με το οποίο θα δημιουργήσουμε ψευδή κίνηση, τόσο κακόβουλη όσο και κανονική, και στην συνέχεια βασιζόμενοι πάνω σε υπάρχουσες έρευνες πάνω στην αναγνώριση κακόβουλης κίνησης, θα μελετήσουμε σε τι βαθμό η κίνηση που δημιουργήσαμε μπορεί να ανιχνευθεί.

**Λέξεις κλειδιά:** κυβερνοασφάλεια ♦ κακόβουλη κίνηση ♦ ανίχνευση κακόβουλης κίνησης ♦ δημιουργία ψευδούς κίνησης ♦ μηχανική μάθηση ♦ τυχαίο δάσος

# Abstract

The transition to the Information Age has made Information one of the most important assets of a modern society. As a natural consequence, this asset is under constant threat from cybercriminals, malicious entities that aim to profit or damage the information resources of an organization or government agency. One of the important strategies to deal with cyber-attacks is to continuously test those systems that aim to identify and neutralize threats. Cybersecurity is a subfield of Information Science that studies and designs such cyber-attack strategies and methodologies.

An important tool that is now used more often in many scientific fields is the Machine Learning-ML. With modern attacks becoming more and more complex, leaving a larger digital footprint, security researchers have begun to utilize Machine Learning potential to recognize and neutralize such attacks.

In the present study, we will develop a Machine Learning model that will create false traffic, both malicious and normal, and then based on existing research on malicious motion recognition, we will study the extent to which the traffic we have created can be detected.

**Keywords:** cybersecurity ♦ malicious traffic ♦ malware detection ♦ false traffic generation ♦ machine learning ♦ random forest

# 1 Εισαγωγή

## 1.1 Γενικά

Η μετάβαση στην εποχή της Πληροφορίας έχει περισσότερο από τρεις δεκαετίες που έχει επιτελεστεί. Την μετάβαση αυτή ωστόσο έχουν επιταχύνει τα τελευταία είκοσι χρόνια οι εκρηκτικοί ρυθμοί ανάπτυξης των επικοινωνιών και της επεξεργαστικής ισχύς. Με τις κυβερνήσεις και τις κοινωνίες να έχουν πλήρως προσαρμοστεί στην νέα πραγματικότητα, η Πληροφορία έχει μετατραπεί σε ένα από τα σημαντικότερα αγαθά ενός σύγχρονου κράτους. Τα μέσα, ωστόσο, που την έκαναν εύκολα προσβάσιμη σε όλους είναι τα ίδια εκείνα μέσα που η κακόβουλη χρήση τους την απειλούν άμεσα και διαρκώς.

Η Κυβερνοασφάλεια, ήταν το αποτέλεσμα της ανάγκης για προστασία της Πληροφορίας σε όλες τις σύγχρονες μορφές της. Σαν έννοια, όντας μη αυστηρά καθορισμένη, περιγράφει το σύνολο των διαδικασιών και μεθοδολογιών για την διασφάλιση των πληροφοριακών πόρων από επιθέσεις. Εξαιτίας του εύρους των επιθέσεων σε σχέση με τα επίπεδα του μοντέλου TCP/IP, το ίδιο ευρύ είναι και το φάσμα των στρατηγικών αντιμετώπισης.

Με την πρόσβαση, ωστόσο, σε Υπολογιστικά Συστήματα όλο και περισσότερων χρηστών, η πλειοψηφία των οποίων χαμηλής επίγνωσης γύρω από θέματα ασφάλειας πληροφοριακών συστημάτων, και παράλληλα με την εμφάνιση όλο και περισσότερων κενών ασφαλείας ημέρας μηδέν (zero day vulnerability), ως απόρροιας της αύξησης της πολυπλοκότητας του λογισμικού, οι σύγχρονες επιθέσεις έχουν περιθώρια να προσαρμοστούν κατάλληλα, δυσχεραίνοντας το έργο των παραδοσιακών λύσεων. Παράλληλα, η κρυπτογράφηση της δικτυακής κίνησης, ενώ συνέβαλε στην αυξημένη ασφάλεια των επικοινωνιών, αφενός παρείχε στους επιτιθέμενους νέους τρόπους απόκρυψης των επιθέσεών τους και αφετέρου δημιούργησε τροχοπέδη για την παρακολούθηση της κίνησης από εργαλεία λογισμικού [1].

Μια στρατηγική που ξεπερνά σε μεγάλο βαθμό τα προαναφερθέντα προβλήματα και πάνω στην ιδέα της οποίας βασίζεται και η παρούσα εργασία, είναι η μελέτη της δικτυακής κίνησης σε επίπεδο πακέτων. Πιο συγκεκριμένα μοντέλα μηχανικής μάθησης τροφοδοτούνται με χαρακτηριστικά πακέτων, τόσο κακόβουλης όσο και φυσιολογικής κίνησης, και εκπαιδεύονται ώστε να μπορούν σε πραγματικές συνθήκες να διακρίνουν την κακόβουλη κίνηση.

## 1.2 Διπλωματική Εργασία

### 1.2.1 Σκοπός

Στην παρούσα Διπλωματική εργασία, θα αναπτύξουμε ένα μοντέλο Μηχανικής Μάθησης για την δημιουργία δικτυακής κίνησης, ενώ για την εκπαίδευση του μοντέλου θα γίνει χρήση τόσο φυσιολογικής κίνησης όσο και κακόβουλης. Θα γίνει εκτενής αναφορά της μεθοδολογίας μας τόσο ως προς το ίδιο το μοντέλο όσο και ως προς το σετ δεδομένων και τις ιδιαιτερότητές του, ενώ στο τέλος παραθέτουμε τις παρατηρήσεις και τα συμπεράσματα μας.

### 1.2.2 Δομή

Η εργασία χωρίζεται σε έξι κεφάλαια:

- Το πρώτο κεφάλαιο που είναι και το εισαγωγικό.
- Στο δεύτερο κεφάλαιο, εισάγουμε τον αναγνώστη σε θεωρητικές έννοιες απαραίτητες για την κατανόηση της υλοποίησης καθώς και των επιμέρους δομών. Γίνεται αναφορά σε βασικές θεωρήσεις της Μηχανικής Μάθησης καθώς και στα πιο σημαντικά χαρακτηριστικά της. Καθώς τα δεδομένα μας αφορούν σε δικτυακή κίνηση θεωρήσαμε απαραίτητο να κάνουμε και σε αυτό τον τομέα μια σύντομη αλλά συνάμα περιεκτική ανάλυση.
- Στο τρίτο κεφάλαιο, παρουσιάζουμε αναλυτικά την μεθοδολογία μας αναφορικά με την συλλογή των αρχικών δεδομένων μας (raw data) καθώς και την μορφοποίησή αυτών ώστε να είναι κατάλληλα για να τροφοδοτήσουν το μοντέλο μας. Γίνεται ανάλυση του κώδικά μας καθώς και των όποιων δυσκολιών αντιμετωπίσαμε.
- Στο τέταρτο κεφάλαιο, παρουσιάζουμε αναλυτικά την μεθοδολογία μας αναφορικά με το μοντέλο της Μηχανικής Μάθησης που δημιουργήσαμε. Και εδώ, θα κάνουμε ανάλυση του κώδικά μας και των επιλογών που έγιναν.
- Στο πέμπτο κεφάλαιο, γίνεται αναφορά στα προβλήματα που προέκυψαν ως απόρροιας των ιδιαιτεροτήτων του σετ δεδομένων εκπαίδευσης καθώς και στις απαραίτητες διορθωτικές κινήσεις.
- Τέλος στο έκτο κεφάλαιο, θα παραθέσουμε τα ευρήματα της μελέτης μας καθώς και τα συμπεράσματά μας.

### 1.2.3 Σχετικές Έρευνες

Όπως αναφέραμε και στην ενότητα 1.2.1, για την δημιουργία του μοντέλου μας, οπότε και του σετ δεδομένων εισόδου, βασιστήκαμε σε υπάρχουσες έρευνες πάνω στην αναγνώριση κακόβουλης κίνησης μέσω της εξαγωγής χαρακτηριστικών από δικτυακή κίνηση. Τέτοιες έρευνες είναι οι [2], όπου γίνεται χρήση 28 χαρακτηριστικών ανά παρατήρηση αλλά επικεντρώνεται στο επίπεδο Λογισμικού της σουίτας πρωτοκόλλων TCP/IP (βλ. 2.2.1.1), καθώς και της CISCO [3], όπου παρουσιάζεται μια λύση ανάλυσης της κίνησης σε πραγματικό χρόνο με χρήση συγκεκριμένων χαρακτηριστικών. Η δική μας υλοποίηση βασίστηκε στην επιλογή χαρακτηριστικών (feature selection) της [3], όπου με 5 χαρακτηριστικά πέτυχαν υψηλή απόδοση στην πρόβλεψη της κακόβουλης κίνησης.

### 1.2.4 Περιβάλλον Ανάπτυξης & Επιλογές

Εφόσον η παρούσα εργασία στον πυρήνα της βασίζεται σε υλοποίηση μοντέλου Μηχανικής Μάθησης, κρίναμε απαραίτητο κλείνοντας το εισαγωγικό αυτό κεφάλαιο να κάνουμε αναφορά στο περιβάλλον ανάπτυξης καθώς και στο υλικό.

**Υλικό**

Χρησιμοποιήθηκε προσωπικός Η/Υ καθώς τα προαπαιτούμενα για την υλοποίηση στο νέφος (cloud) κρίναμε ότι απαιτούσαν αρκετά περισσότερο χρόνο από όσο θα θέλαμε. Αποτελείται από CPU AMD Ryzen 5600x 6-core/12-threads με 32GB RAM. Η κάρτα γραφικών είναι επίσης AMD, αλλά ούτως ή άλλως η *"sklearn"* βιβλιοθήκη, δεν εκμεταλλεύεται την κάρτα γραφικών όπως η βιβλιοθήκη *"tensor flow"*, η οποία χρησιμοποιείται για διαφορετικά μοντέλα Μηχανικής Μάθησης.

**Περιβάλλον Ανάπτυξης**

Ξεκινώντας από το Λειτουργικό Σύστημα, χρησιμοποιήθηκε τόσο Windows 10 όσο και Ubuntu Linux σε VM Ware image. Ο λόγος είναι ότι υπήρξε αδυναμία στην εγκατάσταση κάποιων απαραίτητων βιβλιοθηκών που χρειαστήκαμε κατά την φάση της εξόρυξης δεδομένων σε περιβάλλον Λ.Σ. Windows. Αυτό είχε ως αποτέλεσμα όλη η πρώτη φάση της υλοποίησης, που αφορούσε την δημιουργία του τελικού σετ δεδομένων, να ολοκληρωθεί σε περιβάλλον Ubuntu. Η δεύτερη φάση, ωστόσο, που αφορούσε την υλοποίηση του μοντέλου έγινε στα Windows καθώς κρίναμε απαραίτητη την αμεσότερη πρόσβαση στους πόρους του συστήματος σε σχέση με το Ubuntu-over-VM.

Σε κάθε περίπτωση, η γλώσσα προγραμματισμού που επιλέξαμε ήταν η Python, μια γλώσσα προγραμματισμού που μπορεί να θεωρηθεί η de facto επιλογή για την ανάπτυξη μοντέλων Μ.Μ. ενώ ως προγραμματιστικό περιβάλλον επιλέξαμε το PyCharm Community Edition. Οι δύο βασικές βιβλιοθήκες που χρησιμοποιήσαμε ήταν η *scapy* [4] για την πρώτη φάση και την sklearn [5] για την δεύτερη. Περισσότερες πληροφορίες στα αντίστοιχα κεφάλαια. Να σημειώσουμε στο σημείο αυτό ότι για την βιβλιοθήκη *scapy* χρειάστηκε να χρησιμοποιήσουμε Python v. 2.7 καθώς με την τελευταία έκδοση, v. 3.9 κατά την φάση της υλοποίησης, υπήρξαν ασυμβατότητες.

Τέλος για την σχεσιακή βάση, έγινε χρήση της Maria DB [6], fork της MySQL και του διαχειριστικού προγράμματος βάσεων δεδομένων, HeidiSQL.

# 2 Βασικές Έννοιες

## 2.1 Εισαγωγή

Το παρόν κεφάλαιο θα το αφιερώσουμε σε βασικές θεωρητικές έννοιες πάνω στις οποίες βασίστηκε η υλοποίησή μας. Εκτός από το προφανές αντικείμενο της Μηχανικής Μάθησης (Μ.Μ.), παραθέτουμε μια σύντομη αναφορά πάνω στην θεωρία Δικτύων δίνοντας σημασία στο μοντέλο TCP/IP. Ο τελικός σκοπός μας είναι, ο αναγνώστης να είναι σε θέση να κατανοήσει και τις δύο φάσεις της υλοποίησής μας.

## 2.2 Μηχανική Μάθηση

Στη παρούσα ενότητα, γίνεται ανάλυση των βασικών θεωριών και επιμέρους δομών σχετικά με την Μηχανική Μάθηση (Μ.Μ.) ενώ γίνεται αναφορά και στο μοντέλο του Τυχαίου Δάσους το οποίο επιλέξαμε και για την υλοποίησή μας.

### 2.2.1 Βασικές Θεωρήσεις

Η επιστήμη της Μηχανικής Μάθησης διερευνά την μελέτη και την κατασκευή αλγορίθμων που μπορούν να βελτιώνονται αυτόματα μέσα από εμπειρία και τη χρήσης δεδομένων. Γεννήθηκε μέσα από την έρευνα της Τεχνητής Νοημοσύνης, όπου υπήρξε ενδιαφέρον για μηχανές που μπορούν να «μάθουν» από δεδομένα [7], [8]. Σαν επιστήμη είναι στενά συνδεδεμένη με εκείνη της Στατιστικής [9]. Σύμφωνα με τον Arthur Smauel [10], που είναι και ο εφευρέτης του ονόματος, η Μηχανική Μάθηση είναι:

«*Το πεδίο μελέτης που δίνει την ικανότητα στους υπολογιστές να κάνουν προβλέψεις και να λαμβάνουν αποφάσεις χωρίς να είναι προγραμματισμένοι από πριν*».

Ένας πιο φορμαλιστικός ορισμός που δόθηκε από τον Tom Mitchell [11] έχει ως εξής:

*«Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από εμπειρία E ως προς μια κλάση εργασιών T και ένα μέτρο επίδοσης P, αν η επίδοσή του σε εργασίες της κλάσης T, όπως αποτιμάται από το μέτρο P, βελτιώνεται με την εμπειρία E»*

Από την επιστημονική σκοπιά της Στατιστικής έγινε μια προσέγγιση που, αν και δεν αναφερόταν άμεσα στην Μηχανική Μάθηση, η ιδέα που πραγματευόταν ήταν πρακτικά η ίδια. Οι Στατιστικολόγοι, Trevor Hastie,

Robert Tibshirani και Jerome Friedman, στο βιβλίο τους "The Elements of Statistical Learning" [12] διατύπωσαν το εξής:

«*Τεράστιες ποσότητες δεδομένων παράγονται σε πολλά επιστημονικά πεδία, και η δουλειά του Στατιστικολόγου είναι να τα κατανοεί, να εξάγει δηλαδή σημαντικά μοτίβα και τάσεις και να καταλαβαίνει «τι προσπαθούν να πουν». Αυτό το ονομάζουμε μάθηση από τα δεδομένα*»

### 2.2.2 Είδη Μάθησης

Ανάλογα με τον τρόπο τροφοδότησης του συστήματος εκμάθησης, η Μ.Μ. ταξινομείται σε τέσσερα είδη μάθησης [8] .

#### *2.2.2.1 Επιτηρούμενη Μάθηση (Supervised Learning)*

Πρόκειται για την διαδικασία εκμάθησης μια συνάρτησης η οποία αντιστοιχεί ένα στοιχείο εισόδου με ένα εξόδου βασιζόμενη σε ζευγάρια εισόδου-εξόδου που περιλαμβάνονται στο σετ δεδομένων εκπαίδευσης (βλ. ενότητα 2.2.6) [13], [14]. Ο αλγόριθμος Μ.Μ. αναλύει τα δεδομένα εκπαίδευσης και παράγει μια συνάρτηση που απεικονίζει δεδομένες εισόδους (σύνολο εκπαίδευσης) σε γνωστές επιθυμητές εξόδους, με απώτερο στόχο τη γενίκευση της συνάρτησης αυτής και για εισόδους με άγνωστη έξοδο (unseen data).[15] Τα μοντέλα που ανήκουν σε αυτό το είδος μάθησης χωρίζονται σε δύο υποκατηγορίες ανάλογα με την αναμενόμενη έξοδο, στα ταξινόμησης και στα παλινδρόμησης. Ένας επιπλέον διαχωρισμός αφορά στην διαχείριση του σετ δεδομένων εκπαίδευσης όπου η μάθηση διακρίνεται σε τεμπέλικη και ενθουσιώδη:

- Τεμπέλικη μάθηση (lazy learning)

  Στην συγκεκριμένη μάθηση, το σετ δεδομένων εκπαίδευσης αποθηκεύεται μέχρι να γίνει κάποιο αίτημα στο σύστημα. Ο σκοπός αυτής της επιλογής είναι για τις περιπτώσεις που απαιτείται το σετ δεδομένων να ενημερώνεται διαρκώς με νέες καταχωρήσεις (π.χ. συνδρομητικές υπηρεσίες πολυμέσων ή Διαδικτυακά καταστήματα). Οι «τεμπέληδες» αλγόριθμοι χαρακτηρίζονται από σύντομο χρόνο εκπαίδευσης αλλά χρειάζονται περισσότερο χρόνο για να κάνουν την πρόβλεψη.

  Το βασικό πλεονέκτημα της μεθόδους αυτής είναι η ευελιξία ως προς τις αλλαγές στο πεδίο του προβλήματος (problem domain) καθώς, η προσέγγιση της συνάρτησης στόχος (target function) γίνεται τοπικά στο σύστημα με αποτέλεσμα να επιτυγχάνεται επίλυση πολλών διαφορετικών προβλημάτων.

  Ένα αναμενόμενο μειονέκτημα που προκύπτει από την αποθήκευση τοπικά του σετ δεδομένων εκπαίδευσης, είναι το κόστος σε αποθηκευτικούς πόρους λόγω του όγκου δεδομένων όσο και από άποψη επεξεργαστικών πόρων λόγω των αιτημάτων αναζήτησης κατά την διαδικασία της πρόβλεψης. Επιπλέον το σετ δεδομένων ενδέχεται να αποκτήσει «θόρυβο» και να μεγαλώσει δυσανάλογα, καθώς κατά την φάση της εκπαίδευσης δεν πραγματοποιείται η διαδικασία της μάθησης.

  Γνωστοί αλγόριθμοι: K-NN(K-Nearest Neighbors), Local Regression, Lazy naïve Bayes

- Ενθουσιώδης μάθηση (eager learners)
  Σε αντίθεση με την τεμπέλικη, στην ενθουσιώδη μάθηση, η συνάρτηση στόχος προσδιορίζεται κατά την διάρκεια της εκπαίδευσης. Με αυτό τον τρόπο αυξάνεται μεν ο χρόνος εκπαίδευσης αλλά ο χρόνος πρόβλεψης μειώνεται ενώ παράλληλα επιτυγχάνεται οικονομία πόρων.

Γνωστοί αλγόριθμοι: Decision Trees, Artificial Neural Networks, Naïve Bayes

#### *2.2.2.2 Μη Επιτηρούμενη Μάθηση (Unsupervised Learning)*

Όταν, ωστόσο, δεν παρέχεται σετ δεδομένων εκπαίδευσης και το σύστημα μάθησης καλείται να αναγνωρίσει μοτίβα σε δεδομένα μη οργανωμένα ή ταξινομημένα, πρόκειται για μάθηση μη επιτηρούμενη. Ουσιαστικά ο αλγόριθμος μάθησης καλείται χωρίς καθοδήγηση να αναγνωρίσει συσχετισμούς ή απουσία αυτών ανάμεσα από τα δεδομένα εισόδου [16], [17] .

#### *2.2.2.3 Ημι-επιτηρούμενη Μάθηση (Semi-supervised Learning)*

Σε αυτή την κατηγορία μάθησης, παρέχονται στον αλγόριθμο δεδομένα εισόδου τόσο οργανωμένα και κατηγοριοποιημένα όσο και δεδομένα μη οργανωμένα. Με αυτόν τον συνδυασμό δεδομένων είναι εφικτό, πολλές φορές και αυτοσκοπός, να επιτευχθεί αναγνώριση κρυμμένων μοτίβων στα δεδομένα[18].

#### *2.2.2.4 Ενισχυτική Μάθηση (Reinforcement Learning)*

Στο συγκεκριμένο είδος μάθησης, έξυπνοι πράκτορες λαμβάνουν αποφάσεις για ενέργειες έχοντας την έννοια της ανταμοιβής ως κίνητρο. Με τον τρόπο αυτό μαθαίνουν βάσει προηγούμενης εμπειρίας καθώς παίρνουν αποφάσεις που στο παρελθόν είχαν αποδώσει περισσότερες επιβραβεύσεις παρά τιμωρίες. Σε αντίθεση με την επιτηρούμενη, ο σχεδιαστής διαμορφώνει την πολιτική της επιβράβευσης αλλά δεν παρέχει δεδομένα εισόδου στον αλγόριθμο [19].

### 2.2.3 Ταξινόμηση & Παλινδρόμηση

Ανάλογα με την έξοδο του μοντέλου, η επιτηρούμενη μάθηση χωρίζεται σε δύο υποκατηγορίες, σε προβλήματα ταξινόμησης και σε προβλήματα παλινδρόμησης.

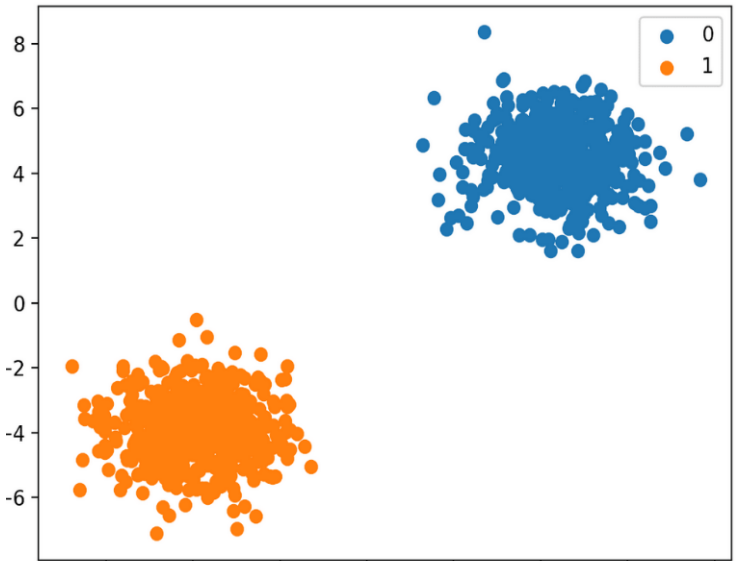
#### *2.2.3.1 Ταξινόμηση (Classification)*

Πρόκειται για την διαδικασία πρόβλεψης της κλάσης μίας παρατήρησης. Το πιο χαρακτηριστικό παράδειγμα που συναντάται στην βιβλιογραφία είναι ο χαρακτηρισμός ενός μηνύματος αλληλογραφίας ως spam ή νορμάλ. Απαιτείται ένα εκτενές σετ δεδομένων που να αντιπροσωπεύει το πρόβλημα επαρκώς ώστε το μοντέλο να καταφέρει να πετύχει υψηλή ακρίβεια ως προς την αντιστοίχιση των δεδομένων εισόδου με την σωστή κλάση[20]. Διακρίνεται στις παρακάτω βασικές κατηγορίες:

- **Δυαδική Ταξινόμηση (Binary Classification):** Αναφέρεται στα προβλήματα ταξινόμησης όπου η κλάση που θέλουμε να προβλέψουμε έχει δύο τιμές. Συνήθως υπάρχουν δύο καταστάσεις, μια κανονική και μια μη- κανονική στις οποίες αναθέτονται οι τιμές 0 και 1 αντίστοιχα. Ο σχεδιασμός των μοντέλων αυτών βασίζεται στην κατανομή Bernoulli [21], όπου μία ενέργεια μπορεί να έχει δύο ακριβώς εκβάσεις, 0 ή 1. Χαρακτηριστικό παράδειγμα η ρίψη ενός κέρματος.

  $$x \text{ in } \{0,1\}$$

Γνωστοί αλγόριθμοι: Λογιστική Παλινδρόμηση, k-nearest Neighbors, Δέντρα Απόφασης, Naive Bayes, Support Vector Machines

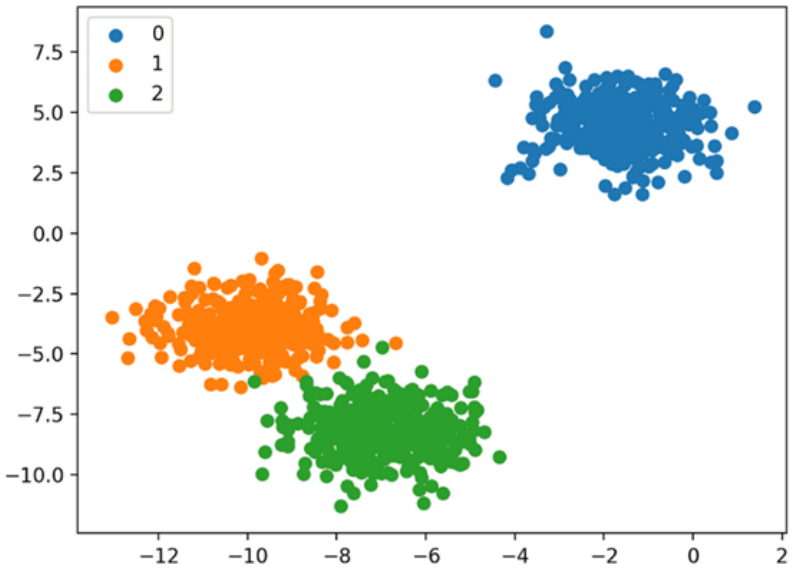



*Εικόνα 1 - Scatter plot δεδομένων Δυαδικής Ταξινόμησης*

- **Ταξινόμηση πολλών κλάσεων (Multi-Class Classification):** Αναφέρεται σε προβλήματα τα οποία ανήκουν σε μια κλάση ανάμεσα από πολλές και δεν ακολουθείται η έννοια κανονικής και μη κανονικής κατάστασης. Το πλήθος των κλάσεων στα πλαίσια ενός προβλήματος δύναται να είναι πολύ μεγάλος όπως σε παραδείγματα αναγνώρισης ακολουθίας λέξεων σε μεταφραστικά μοντέλα. Εδώ ο σχεδιασμός βασίζεται στην κατανομή Multinoulli [22], που ονομάζεται και κατηγορική κατανομή, όπου περιγράφει την πιθανότητα μια ενέργεια να έχει μια από *K* εκβάσεις.

$$x \text{ in } \{1,2,3,\ldots, K\}$$

Γνωστοί αλγόριθμοι: Random Forest, k-nearest Neighbors, Δέντρα Απόφασης, Naive Bayes, Gradient Boosting




*Εικόνα 2 - Scatter plot δεδομένων Ταξινόμησης πολλών κλάσεων*

- **Ταξινόμηση πολλαπλών ετικετών (Multi-label Classification):** Πρόκειται για την περίπτωση ταξινόμησης όπου κάθε παρατήρηση μπορεί να ανήκει σε περισσότερες από μια κλάσεις ταυτόχρονα. Ουσιαστικά, πρόκειται για πολλαπλές δυαδικές ταξινομήσεις για κάθε παρατήρηση.

#### *2.2.3.2 Παλινδρόμηση (Regression)*

Το δεύτερο είδος επιτηρούμενης μάθησης, η παλινδρόμηση, βασίζεται στην εύρεση μια συνάρτησης στόχου που αντιστοιχίζει τις μεταβλητές εισόδου, εξαρτημένες μεταβλητές (dependent variables), με μία ή περισσότερες μεταβλητές εξόδου, ανεξάρτητες μεταβλητές (independent variables) [23]. Σε αντίθεση με την ταξινόμηση, η μεταβλητή εξόδου παίρνει συνεχείς τιμές. Στην ενότητα 2.2.5 θα αναλύσουμε την θεωρία πάνω στα Δέντρα Απόφασης και στο Τυχαίο Δάσος, στα οποία βασίστηκε η υλοποίησή μας.

### 2.2.4 Δομικά στοιχεία της Μηχανικής Μάθησης

Η Μ.Μ. μπορεί να θεωρηθεί ένα μαύρο κουτί, στο οποίο δίνει κάποιος δεδομένα ως είσοδο και μέσα από μια διαδικασία εκπαίδευσης, παίρνει την επιθυμητή έξοδο. Πέρα από τα δεδομένα εισόδου αυτό το μαύρο κουτί αποτελείται από κάποια δομικά στοιχεία τα οποία είναι κοινά. Αυτά είναι το μοντέλο που θα χρησιμοποιηθεί, οι συναρτήσεις στόχου, κόστους και απώλειας καθώς και ο αλγόριθμος βελτιστοποίησης. Ακολουθεί σχετική ανάλυση[24], [25].

- Μοντέλο (Machine Learning Model)
  Το πρώτο στοιχείο ενός αλγόριθμου Μ.Μ. είναι το μοντέλο που θα επιλεγεί για το εκάστοτε πρόβλημα. Όπως αναφέραμε και σε προηγούμενη ενότητα ανάλογα με την επιθυμητή έξοδο, το μοντέλο μπορεί να είναι ταξινόμησης ή παλινδρόμησης ενώ η ύπαρξη ή μη σετ δεδομένων εκπαίδευσης καθορίζει αν πρόκειται για μοντέλο Επιτηρούμενης μάθησης ή Μη Επιτηρούμενης. Παρακάτω παραθέτουμε επιγραμματικά τα πιο γνωστά μοντέλα ανά κατηγορία:

Επιτηρούμενη Μάθηση-Παλινδρόμηση

- Γραμμική Παλινδρόμηση (Linear Regression)
- Δέντρα απόφασης (Decision Trees)
- Τυχαίο Δάσος (Random Forest)
- Νευρωνικά Δίκτυα (Neural Networks)

Επιτηρούμενη Μάθηση-Ταξινόμηση

- Λογιστική Παλινδρόμηση (Logistic Regression)
- Support Vector Machine
- Naïve Bayes
- Δέντρα απόφασης (Decision Trees) για Ταξινόμηση
- Τυχαίο Δάσος (Random Forest) για Ταξινόμηση
- Νευρωνικά Δίκτυα (Neural Networks) για Ταξινόμηση

Μη Επιτηρούμενη Μάθηση

- Συσταδοποίηση (Clustering)
- Dimensionality Reduction
- Principal Component Analysis (PCA)

Συνάρτηση Στόχος (Objective Function)

Είναι ένα από τα βασικότερα στοιχεία ενός προβλήματος Μ.Μ. καθώς είναι εκείνη που περιγράφει πλήρως το πρόβλημα. Είναι η συνάρτηση εκείνη που ανάλογα με το πρόβλημα η τιμή της θα πρέπει να μειώνεται ή να αυξάνεται μέσω της διαδικασίας της βελτιστοποίησης.

Συνάρτηση Κόστους (Cost Function) & Συνάρτηση Απώλειας (Loss Function)

Αν και εννοιολογικά το κόστος και η απώλεια είναι συνώνυμα, στα πλαίσια της Μ.Μ. διαφέρουν ως προς το εύρος. Ενώ η συνάρτηση απώλειας μετρά την απόκλιση της πρόβλεψής μας για ένα αντικείμενο σε ένα δεδομένο σετ δεδομένων, η συνάρτηση κόστους μετρά το συνολικό κόστος ενός μοντέλου.

Αλγόριθμος βελτιστοποίησης (Optimization Algorithm)

Σε κάθε πρόβλημα Μ.Μ. κατά την διάρκεια της εκπαίδευσης του μοντέλου τελείται μια διαδικασία βελτιστοποίησης με σκοπό την αύξηση της ακρίβειας της πρόβλεψής μας ή αλλιώς την ελάττωση της απόκλισης από την πραγματική τιμή. Η βελτιστοποίηση που πραγματοποιείται αφορά την προσαρμογή των υπερπαραμέτρων (hyperparameters) του μοντέλου που έχει επιλεγεί με σκοπό την μείωση της συνάρτησης κόστους (cost function) η οποία προσδιορίζει την απόκλιση του μοντέλου.

### 2.2.5 Τυχαίο Δάσος (Random Forest)

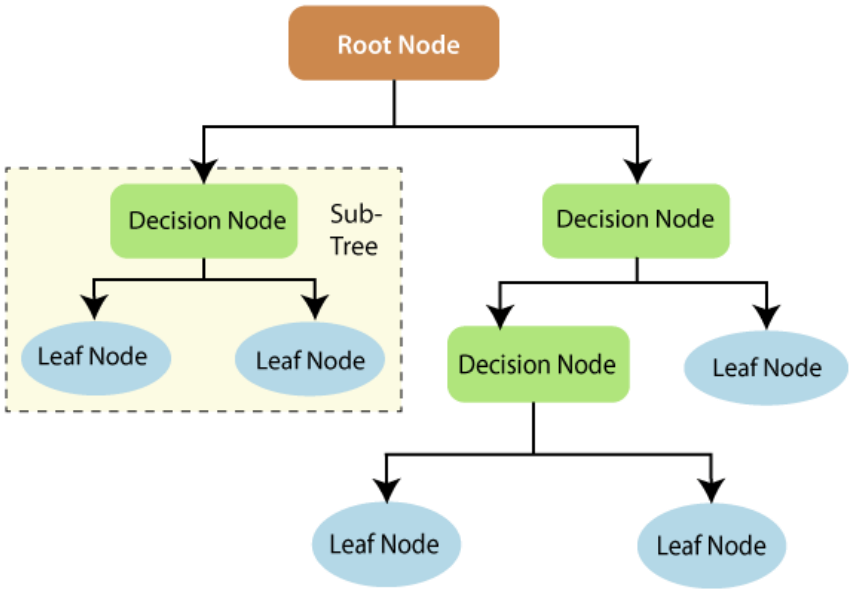
Έχοντας χρησιμοποιήσει στην υλοποίησή μας τον αλγόριθμο Τυχαίου Δάσους για παλινδρόμηση (Random Forest Regressor) που βασίζεται στο μοντέλο των Δέντρων Απόφασης (Decision Trees), θεωρήσαμε λογικό να αφιερώσουμε ξεχωριστή ενότητα για την αναφορά μας σε αυτά. Θα ξεκινήσουμε με την θεωρία των Δέντρων Απόφασης, ενώ στην συνέχεια θα περιγράψουμε την διαδικασία του bootstrapping και πως σχηματίζεται το Τυχαίο Δάσος.

### *2.2.5.1 Δέντρα Απόφασης - Γενική θεώρηση*

Πρόκειται για μια δομή που έχει ομοιότητες με ένα διάγραμμα ροής με την έννοια ότι υπάρχει μια κατεύθυνση και παρόμοια λογική[26]–[28]. Αποτελείται από τα εξής στοιχεία:

- Κόμβος ρίζα (Root Node): Πρόκειται για τον αρχικό κόμβο ενός δέντρου απόφασης και αντιπροσωπεύει το σύνολο του δείγματος το οποίο στην συνέχει χωρίζεται σε επιμέρους κόμβους.
- Κόμβος Απόφασης (Decision Node): Καλείται ο υπό-κόμβος που χωρίζεται περεταίρω σε επιπλέον υπό-κόμβους. Καλούνται «απόφασης» καθώς προσδιορίζουν το κριτήριο του χωρισμού.
- Τερματικός Κόμβος/Κόμβος Φύλλο (Terminal/Leaf Node): Πρόκειται για τους κόμβους που δεν χωρίζονται περεταίρω και περιέχουν μία από τις εκβάσεις μιας απόφασης.

Τα δέντρα απόφασης κάνουν χρήση διάφορων αλγορίθμων ώστε να χωρίσουν έναν κόμβο σε υπό-κόμβους. Στα πλαίσια της Μ.Μ., κάθε εσωτερικός κόμβος είναι ένα χαρακτηριστικό εισόδου (input feature) από τον οποίο προκύπτει είτε ένα διαφορετικός κόμβος με ένα άλλο χαρακτηριστικό είτε ένας τελικός κόμβος/φύλλο που θα περιέχει μία από τις τιμές του χαρακτηριστικού στόχος (target feature). Ανάλογα με την τιμή εξόδου, τα δέντρα απόφασης ταξινομούνται σε ταξινόμησης, αν πρόκειται για διακριτή τιμή, και παλινδρόμησης αν πρόκειται για πραγματικό αριθμό.




*Εικόνα 3 - Σχηματική αναπαράσταση ενός δέντρου απόφασης*

### *2.2.5.2 Ensemble Μέθοδοι – Bootstrap Aggregating*

Πρόκειται για μεθόδους της Μ.Μ. οι οποίες συνδυάζουν πολλούς αλγόριθμους μάθησης με σκοπό την επίτευξη καλύτερης ακρίβειας στην τελική πρόβλεψη θυσιάζοντας, ωστόσο, υπολογιστικούς πόρους κατά την φάση της αξιολόγησης της ακρίβειας πρόβλεψης (evaluation phase) καθώς η διαδικασία υλοποιείται σε περισσότερα από ένα μοντέλα μάθησης. Σύμφωνα με σχετικές έρευνες, ο τρόπος που θα συνδυαστούν τα μοντέλα σε μια μέθοδο έχει σημασία τόσο ως προς τον αριθμό, που σημαίνει πόσα διαφορετικά μοντέλα θα συνδυαστούν, όσο και ως

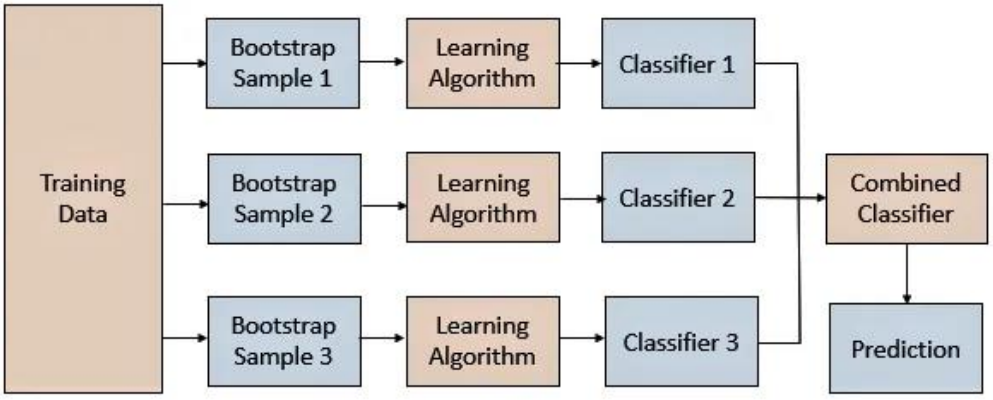
τον βαθμό που ποικίλουν μεταξύ τους. Υπάρχουν διάφοροι τύποι μεθόδων όπως ο Bayes optimal classifier, ο μετά-αλγόριθμος Boosting καθώς και άλλοι, ωστόσο εμείς θα σταθούμε στην μέθοδο Bootstrap aggregating στην οποία ανήκει και ο Random Forest regressor[29].

Bootstrap Aggregating ή Bagging

Πρόκειται, λοιπόν, για μια ensemble μέθοδο η οποία εφαρμόζεται κυρίως σε μεθόδους βασισμένες σε δέντρα απόφασης αλλά μπορεί να εφαρμοστεί και σε οποιαδήποτε άλλη. Το 1996 παρουσιάστηκε για πρώτη φορά ο σχετικός αλγόριθμος από τον Leo Breiman[30]. Περιλάμβανε τα εξής βήματα:

1) **Bootstrapping:** Στο πρώτο βήμα του αλγόριθμου, από το αρχικό σετ δεδομένων προκύπτουν άλλα δύο σετ δεδομένων, το bootstrap και το out-of-bag.
   - Bootstrap dataset: Το πρώτο σετ δεδομένων που δημιουργείται σε αυτό το βήμα, που πρέπει να έχει το ίδιο μέγεθος με το αρχικό, σχηματίζεται μέσω τυχαίας δειγματοληψίας με αντικατάσταση που σημαίνει ότι περιέχει διπλές παρατηρήσεις.
   - Out-of-bag dataset: Το συγκεκριμένο σετ δεδομένων περιλαμβάνει όλες τις παρατηρήσεις που δεν επιλέχθηκαν κατά την δημιουργία του bootstrap dataset και προκύπτει μέσω σύγκρισης του τελευταίου με το αρχικό.

   Με την δημιουργία των bootstrap σετ δεδομένων με τον παραπάνω τρόπο, επιτυγχάνεται μοναδικότητα ως προς τα επί μέρους σετ τα οποία στο επόμενο βήμα θα τροφοδοτήσουν τα δέντρα απόφασης.
2) **Παράλληλη εκπαίδευση (Parallel Training):** Σε αυτό το βήμα ουσιαστικά κάθε δέντρο απόφασης τροφοδοτείται με ένα bootstrap dataset.
3) **Συγκέντρωση (Aggregation):** Στο τελικό στάδιο, λαμβάνεται ο μέσος όρος των προβλέψεων, εάν πρόκειται για πρόβλημα παλινδρόμησης, ενώ στην περίπτωση προβλήματος ταξινόμησης λαμβάνεται υπόψη η κλάση με την πλειοψηφία των ψήφων.




*Εικόνα 4 - Αναπαράσταση της διαδικασίας Bootstrapping για πρόβλημα ταξινόμησης*

Τα σημαντικότερα πλεονεκτήματα του bootstrapping είναι η ευκολία στην υλοποίηση, καθώς υπάρχει πλήθος σχετικών βιβλιοθηκών, και η επίτευξη μείωσης της απόκλισης, περισσότερα στην ενότητα 2.2.6, στο αρχικό

σετ δεδομένων. Επιπλέον ο συνδυασμός πολλών μοντέλων επιτρέπει την χρήση αδύναμων εκπαιδευτών καθώς το αποτέλεσμα προκύπτει από την συνολική απόδοση του μοντέλου[31], [32].

Βασικό μειονέκτημα είναι το υπολογιστικό κόστος, στοιχείο που χαρακτηρίζει, όπως αναφέραμε και προηγούμενα, όλες τις ensemble μεθόδους. Η ιδέα του bagging, ο συνδυασμός δηλαδή πολλών εκπαιδευτών για την εξαγωγή μιας πρόβλεψης υψηλότερης ακρίβειας, δημιουργεί και δύο άλλα προβλήματα. Αρχικά το όποιο συμπέρασμα θα μπορούσε να εξαχθεί από την συμπεριφορά ενός μοντέλου δεν υφίσταται εξαιτίας της συνδυαστικότητας του αλγορίθμου στο τρίτο βήμα. Επιπλέον αν ένας εκπαιδευτής χαρακτηρίζεται ως αδύναμος και από υψηλή μεροληψία (bias), δύναται να μεταδώσει την μεροληψία αυτή στο σύνολο[31], [32].

#### *2.1.5.3 Random Forest*

Ως μια ensemble μέθοδος, ο κορμός του Τυχαίου Δάσους αποτελείται από επί μέρους δέντρα αποφάσεων κάθε ένα από τα οποία τροφοδοτείται με ένα bootstrap dataset που έχει προκύψει από το bootstrapping που περιγράψαμε στην προηγούμενη υποενότητα. Ουσιαστικά πρόκειται για βελτιωμένη έκδοση των bagged δέντρων απόφασης [33], [34].

Σε αντίθεση με μεθόδους όπως η CART (classification and regression trees), χάρη στην τυχαία δειγματοληψία για την δημιουργία των bootstrap datasets, το τυχαίο δάσος που προκύπτει περιέχει δέντρα απόφασης που διαφέρουν μεταξύ τους σε μεγαλύτερο βαθμό. Με αυτό τον τρόπο τα δέντρα είναι ικανά να παράγουν μεγαλύτερο εύρος απαντήσεων.

Βασικά πλεονεκτήματα, είναι η αποδοτική διαχείριση μεγάλων σετ δεδομένων και ιδιαίτερα σετ δεδομένων με υψηλή απόκλιση ενώ παρέχει υψηλότερη ακρίβεια από τον CART. Το βασικό μειονέκτημα είναι το κόστος σε χρόνο και υπολογιστικούς πόρους που προκύπτει από τον συνδυασμό πολλαπλών δέντρων απόφασης.

### 2.2.6 Σετ Δεδομένων

Ως άμεσο συμπέρασμα της θεωρίας των προηγούμενων ενοτήτων, είναι ο καθοριστικός ρόλος του σετ δεδομένων με τα οποία θα εκπαιδευτεί ένα μοντέλο Μ.Μ.. Το σετ αυτό, ωστόσο, ποτέ δεν βρίσκεται σε κατάλληλη μορφή. Μία από τις μεγαλύτερες προκλήσεις κατά την διαδικασία υλοποίησης ενός μοντέλου, είναι αρχικά η επιλογή της σωστής ακατέργαστης πληροφορίας (raw data) και στη συνέχεια η κατάλληλη μορφοποίησή της στο σετ δεδομένων εισόδου ανάλογα με το εκάστοτε μοντέλο Μ.Μ.. Τα ποιοτικά και ποσοτικά χαρακτηριστικά του σετ δεδομένων εισόδου , ανεξάρτητα από την προσέγγιση που θα επιλεγεί ως προς το μοντέλο, καθορίζουν σε μεγάλο βαθμό την συνολική απόδοσή του. Στην ενότητα αυτή θα αναλύσουμε βασικές έννοιες αναφορικά με την προετοιμασία των δεδομένων καθώς και παράγοντες που επηρεάζουν την απόδοση και το αποτέλεσμα του μοντέλου Μ.Μ.[35].

#### *2.2.6.1 Μορφή Δεδομένων Εισόδου*

Σε κάθε μοντέλο Μ.Μ., ο αλγόριθμος εκπαίδευσης μαθαίνει πως να αντιστοιχίζει δεδομένων εισόδου σε μια συγκεκριμένη μεταβλητή στόχο. Τέτοια δεδομένα εισόδου απεικονίζονται σε δομημένη μορφή (structured) και παρουσιάζονται σε πίνακες με γραμμές και στήλες (tabular). Κάθε γραμμή αναπαριστά μια περίπτωση ή παράδειγμα (instance ή example) από τον χώρο του προβλήματος (problem domain) και κάθε στήλη σε ένα

συγκεκριμένο χαρακτηριστικό ή γνώρισμα (feature ή attribute) αυτής της παρατήρησης. Επιπλέον, οι στήλες χωρίζονται σε μεταβλητές εισόδου (input variables), οι οποίες παρέχονται στο μοντέλο για να κάνει μια πρόβλεψη, και σε μεταβλητές εξόδου (output variable) που είναι η πρόβλεψη αυτή.

| longitude | latitude | housing_m... | total_rooms | total_bedr... | population | households | median_in... | median_ho... | ocean_pro... |
|---|---|---|---|---|---|---|---|---|---|
| -122.23 | 37.88 | 41.0 | 880.0 | 129.0 | 322.0 | 126.0 | 8.3252 | 452600.0 | NEAR BAY |
| -122.22 | 37.86 | 21.0 | 7099.0 | 1106.0 | 2401.0 | 1138.0 | 8.3014 | 358500.0 | NEAR BAY |
| -122.24 | 37.85 | 52.0 | 1467.0 | 190.0 | 496.0 | 177.0 | 7.2574 | 352100.0 | NEAR BAY |
| -122.25 | 37.85 | 52.0 | 1274.0 | 235.0 | 558.0 | 219.0 | 5.6431 | 341300.0 | NEAR BAY |
| -122.25 | 37.85 | 52.0 | 1627.0 | 280.0 | 565.0 | 259.0 | 3.8462 | 342200.0 | NEAR BAY |
| -122.25 | 37.85 | 52.0 | 919.0 | 213.0 | 413.0 | 193.0 | 4.0368 | 269700.0 | NEAR BAY |
| -122.25 | 37.84 | 52.0 | 2535.0 | 489.0 | 1094.0 | 514.0 | 3.6591 | 299200.0 | NEAR BAY |
| -122.25 | 37.84 | 52.0 | 3104.0 | 687.0 | 1157.0 | 647.0 | 3.12 | 241400.0 | NEAR BAY |
| -122.26 | 37.84 | 42.0 | 2555.0 | 665.0 | 1206.0 | 595.0 | 2.0804 | 226700.0 | NEAR BAY |
| -122.25 | 37.84 | 52.0 | 3549.0 | 707.0 | 1551.0 | 714.0 | 3.6912 | 261100.0 | NEAR BAY |
| -122.26 | 37.85 | 52.0 | 2202.0 | 434.0 | 910.0 | 402.0 | 3.2031 | 281500.0 | NEAR BAY |
| -122.26 | 37.85 | 52.0 | 3503.0 | 752.0 | 1504.0 | 734.0 | 3.2705 | 241800.0 | NEAR BAY |
| -122.26 | 37.85 | 52.0 | 2491.0 | 474.0 | 1098.0 | 468.0 | 3.075 | 213500.0 | NEAR BAY |
| -122.26 | 37.84 | 52.0 | 696.0 | 191.0 | 345.0 | 174.0 | 2.6736 | 191300.0 | NEAR BAY |

*Εικόνα 5 - Στιγμιότυπο tabular data από το dataset "California Housing Prices" (source Kaggle)*

### *2.2.6.2 Προετοιμασία Δεδομένων (Data pre-processing)*

Αφού οριστεί το πλαίσιο του προβλήματος, ο προσδιορισμός δηλαδή του τι ακριβώς θέλουμε να προβλέψουμε, θα πρέπει να αποφασιστεί ποια είναι εκείνα τα δεδομένα από τον χώρο του προβλήματος που θεωρούμε ότι θα βοηθήσουν τον αλγόριθμο στις προβλέψεις του. Σχεδόν πάντοτε, τα αρχικά αυτά δεδομένα είναι ακατέργαστα (raw) και θα πρέπει να μορφοποιηθούν κατάλληλα.

Μια πρώτη προϋπόθεση που θα πρέπει να ικανοποιηθεί έχει να κάνει με το γεγονός ότι τα δεδομένα εισόδου θα πρέπει να είναι αριθμητικά. Όπως αναφέραμε και στην αρχή του κεφαλαίου, η Μηχανική Μάθηση συνδέεται άρρηκτα με τα επιστημονικά πεδία των μαθηματικών και της στατιστικής. Κάθε αλγόριθμος της Μ.Μ. στον πυρήνα του εκτελεί πράξεις με αριθμούς. Λαμβάνει ως είσοδο αριθμητικά δεδομένα και παράγει προβλέψεις που είναι επίσης δεδομένα αριθμητικά. Οπότε, ένα πρώτο βήμα στην μορφοποίηση αυτή είναι η μετατροπή οποιωνδήποτε χαρακτηριστικών μιας παρατήρησης που παριστάνονται από χαρακτήρες ή ολόκληρες λέξεις, σε αριθμούς.

Ένα άλλο πρόβλημα, αφορά στον συνδυασμό του μοντέλου Μ.Μ. που θα επιλεγεί και των δεδομένων εισόδου. Θεωρώντας ότι η συνθήκη της προηγούμενης παραγράφου έχει ικανοποιηθεί, θα πρέπει να ληφθεί υπόψη ότι διαφορετικά μοντέλα έχουν διαφορετικό βαθμό ευαισθησίας στα δεδομένα με τα οποία θα εκπαιδευτούν με άμεσο αντίκτυπο στην απόδοσή τους. Μια επιπλέον μορφοποίηση είναι απαραίτητη ώστε να μειωθεί ο αντίκτυπος αυτός. Σημαντικοί παράγοντες που μπορεί να επηρεάσουν είναι οι εξής[36]:

- Υψηλός Συσχετισμός (High Correlation): Αφορά στο πόσο η τιμή ενός γνωρίσματος εξαρτάται από την τιμή ενός άλλου γνωρίσματος. Η ποσοτικοποίηση του συσχετισμού αυτού είναι σημαντικό βήμα που θα βοηθήσει στην καλύτερη προετοιμασία των δεδομένων.
- Κανονική κατανομή (Gaussian distribution): Καλείται η κατανομή όπου οι τυχαίες μεταβλητές πραγματικών τιμών τείνουν να συγκεντρώνονται γύρω από μία μέση τιμή. Πολλοί αλγόριθμοι

αναμένουν μια τέτοια κατανομή στις τιμές των γνωρισμάτων ενός σετ δεδομένων. Σε περίπτωση ύπαρξης τιμών οι οποίες είναι έκτοπες (outliers) ενδέχεται να επηρεάσουν την απόδοση τέτοιων αλγόριθμων και ως εκ τούτου οι τιμές αυτές θα πρέπει να αφαιρεθούν. Άλλοι αλγόριθμοι μπορεί να επηρεάζονται λιγότερο αλλά να απαιτούν περισσότερες παρατηρήσεις για να το επιτύχουν.

Ένας τελευταίος παράγοντας που δεν αφορά τόσο την μορφοποίηση, είναι η ποιότητα των ίδιων των δεδομένων. Ένα χαρακτηριστικός όρος που υπάρχει στην Επιστήμη των Υπολογιστών, και φυσικά αφορά και την Μηχανική Μάθηση, είναι ο «GIGO - garbage in,garbage out»[37]. Βασίζεται στην ιδέα ότι, αν τα δεδομένα εισόδου είναι «σκουπίδια» τότε και τα δεδομένα εξόδου θα είναι «σκουπίδια». Στα πλαίσια της Μ.Μ., ένα σετ δεδομένων, ακόμα και ένα επαρκούς μεγέθους, θα πρέπει να αναπαριστά επαρκώς και το πρόβλημα που θέλουμε να λύσουμε. Πολύπλοκα αρχικά δεδομένα ενδέχεται να χρήζουν περεταίρω ανάλυσης ώστε να αποκαλύψουν συσχετίσεις που σχετίζονται με το κεντρικό πρόβλημα. Επιπλέον προβληματικά δεδομένα, που περιέχουν λάθη, κενά και αντιφατικές παρατηρήσεις, οδηγούν τον αλγόριθμο μάθησης σε λάθος προβλέψεις.

Με τους αλγόριθμους μάθησης να είναι ευρέως διαθέσιμοι σε βιβλιοθήκες ανοιχτού κώδικα, είναι εμφανές ότι η τροφοδότησή τους με την σωστή πληροφορία, είναι η πιο χρονοβόρα και απαιτητική διαδικασία ενός έργου Μηχανικής Μάθησης και είναι ταυτόχρονα και εκείνο που την διαφοροποιεί από άλλα έργα.

## 2.3 Βασικές έννοιες Δικτύων

Καθώς για την ανάγκες της υλοποίησή μας, το σετ δεδομένων εισόδου αποτελείται από δικτυακή κίνηση και πιο συγκεκριμένα, χαρακτηριστικά πακέτων δικτύου, κρίναμε απαραίτητο να αφιερώσουμε την τελευταία ενότητα του κεφαλαίου αυτού σε βασικές έννοιες της οικογένειας πρωτοκόλλων TCP/IP.

### 2.3.1 Σουίτα πρωτοκόλλων TCP/IP (Internet Protocol suite)

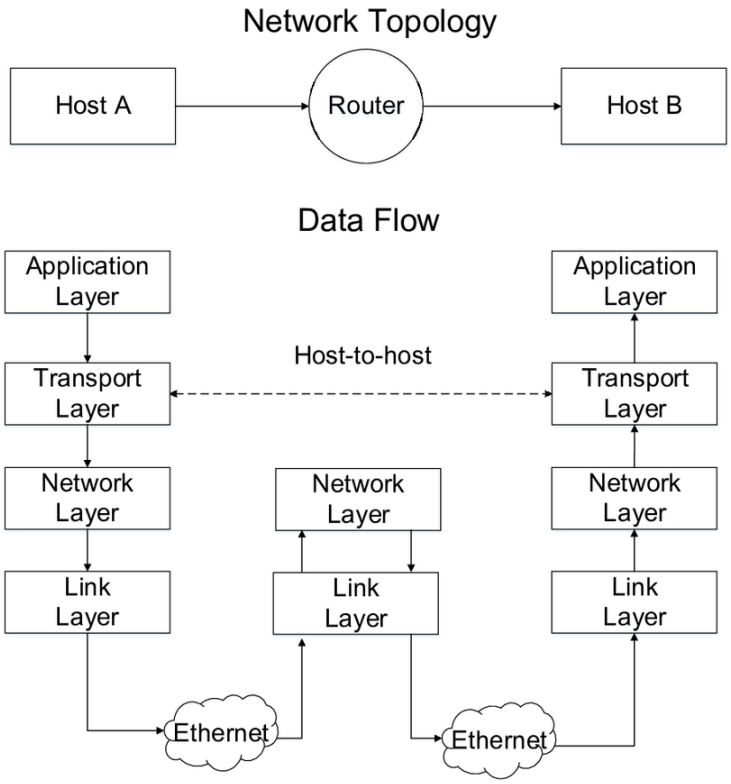
Η σουίτα πρωτοκόλλων TCP/IP είναι μια οικογένεια πρωτοκόλλων επικοινωνίας που χρησιμοποιούνται καθολικά στο Διαδίκτυο και σε δίκτυα υπολογιστών. Τα βασικά πρωτόκολλα είναι το TCP (Transmission Control Protocol) το IP (Internet Protocol) και το UDP (User Datagram Protocol). Η σουίτα αυτή, ήταν το αποτέλεσμα έρευνας και ανάπτυξης ενός παραρτήματος του Υπουργείου Αμύνης των ΗΠΑ στα τέλη της δεκαετίας του 60’. Αρκετά αργότερα θα ανέτίθετο στα Πανεπιστήμια Stanford και College of London καθώς και στην εταιρία BBN Technologies η ανάπτυξη μιας λειτουργικής έκδοσης των πρωτοκόλλων για δημόσια χρήση. Με αυτό τον τρόπο δημιουργήθηκε το Πρωτόκολλο Διαδικτύου έκδοσης 4 (Internet Protocol version 4) με την πιο πρόσφατη να είναι η έκδοση 6 (IPv6)[38].

#### *2.3.1.1 Αρχιτεκτονική*

Η σουίτα TCP/IP, σε αντίθεση με το μοντέλο OSI (OSI model)[39] που είναι μεταγενέστερο, έχει μια ιεραρχική δομή που αποτελείται από πέντε συνολικά επίπεδα. Συνοπτικά αποτελείται από τα εξής:

- **Επίπεδο Πρόσβασης Δικτύου**: Πρόκειται για το κατώτερο επίπεδο και αφορά όλα εκείνα τα πρωτόκολλα που περιγράφουν τις τοπολογίες δικτύων και τις απαιτούμενες διεπαφές για την επικοινωνία δύο σημείων. Επιπλέον περιλαμβάνει όλες τις σχετικές προτυποποιήσεις σχετικά με τους φυσικούς διαύλους επικοινωνίας. Αντιστοιχεί στα δύο πρώτα επίπεδα του μοντέλου OSI.

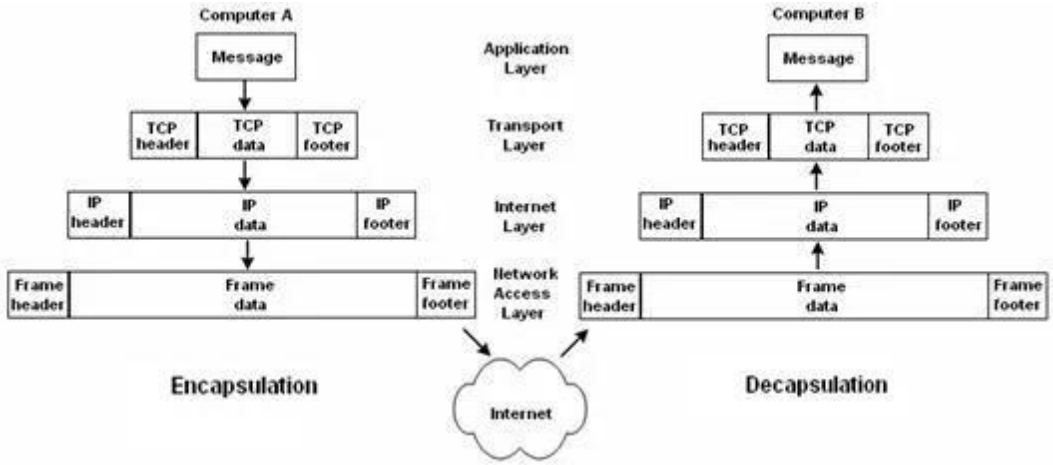
- **Επίπεδο Διαδικτύου**: Αφορά όλα τα πρωτόκολλα που είναι υπεύθυνα για τις μεταδόσεις δεδομένων σε ένα δίκτυο. Αντιστοιχεί στο επίπεδο Δικτύου του μοντέλου OSI. Βασικά πρωτόκολλα είναι τα εξής:
    - **IP**: Πρόκειται για το Internet Protocol που είναι υπεύθυνο για την μεταγωγή IP πακέτων από μια συσκευή αφετηρίας σε μια συσκευή προορισμού. Για να επιτευχθεί αυτό γίνεται χρήση IP διευθύνσεων που βρίσκονται στην κεφαλίδα του πακέτου.
    - **ICMP**: Ακρωνύμια του Internet Control Message Protocol, και πρόκειται για ένα πρωτόκολλο που ενημερώνει τα συμμετέχοντα μέρη για προβλήματα στο δίκτυο.
    - **ARP**: Ακρωνύμια για το Adress Resolution Protocol, και αφορά το πρωτόκολλο υπεύθυνο για την εύρεση διευθύνσεων υλικού (MAC address) για δικτυακές συσκευές με γνωστές IP διευθύνσεις.
- **Επίπεδο Μεταφοράς**: Περιλαμβάνει πρωτόκολλα που σχετίζονται με τον έλεγχο ροής στο δίκτυο. Αντιστοιχεί στο επίπεδο Μεταφοράς του μοντέλου OSI. Τα βασικότερα είναι τα εξής:
    - **TCP (Transmission Control Protocol)**: Παρέχει μεθόδους ώστε να εξασφαλιστεί η αξιόπιστη και χωρίς λάθη επικοινωνία.
    - **UDP (User Datagram Protocol)**: Πρόκειται για ένα πρωτόκολλο ελέγχου ροής το οποίο απαιτεί μικρότερο κόστος σε πόρους και προτιμάται σε περιπτώσεις που δεν απαιτείται η αξιοπιστία που παρέχει το TCP.
- **Επίπεδο Λογισμικού**: Είναι το ανώτερο επίπεδο του μοντέλου και αντιστοιχεί στα τρία τελευταία επίπεδα του μοντέλου OSI, Λογισμικού, Παρουσίασης και Συνόδου. Περιλαμβάνει όλο το λογισμικό σε επίπεδο προγραμμάτων και διεργασιών και την μεταξύ τους επικοινωνία.




*Εικόνα 6 - Αναπαράσταση της δικτυακής κίνησης όπως διατρέχει τα επί μέρους επίπεδα κατά την επικοινωνία δύο εξυπηρετητών.*

### *2.3.1.2 Ενθυλάκωση (Encapsulation) & Απενθυλάκωση (Decapsulation)*

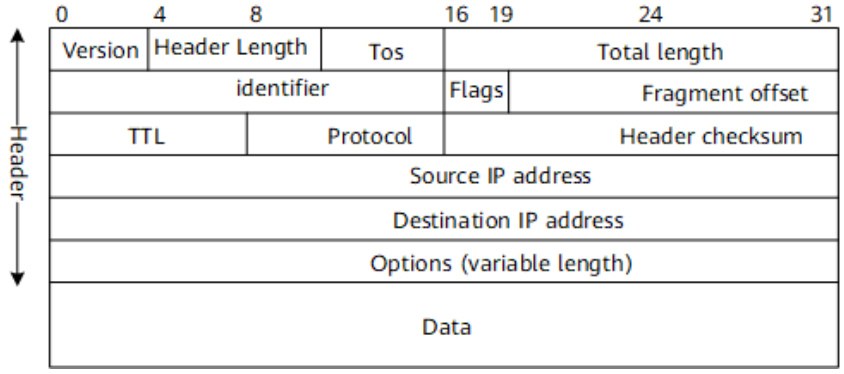
Έχοντας παρουσιάσει συνοπτικά την αρχιτεκτονική της σουίτας πρωτοκόλλων TCP/IP, στην παρούσα ενότητα θα επικεντρωθούμε σε δυο βασικές μεθόδους την ενθυλάκωση και την απενθυλάκωση[40], [41]. Όπως φαίνεται και στην Εικόνα 6, κάθε επίπεδο λαμβάνει πληροφορία από το αμέσως ανώτερο. Σε κάθε βήμα της διαδικασίας αυτής, κάθε επίπεδο χρησιμοποιεί μια δική του μονάδα δεδομένων (Protocol Data Unit – PDU) στην οποία «ενθυλακώνει» την πληροφορία και προσθέτει μια κεφαλίδα με πληροφορίες σχετικές με το συγκεκριμένο επίπεδο. Η ακριβώς ανάποδη διαδικασία ακολουθείται στο άλλο άκρο της επικοινωνίας όπου γίνεται «απενθυλάκωση» από τα κατώτερα επίπεδα προς τα ανώτερα. Η Εικόνα 7 αναπαριστά την διαδικασία αυτή.




*Εικόνα 7 - Διαδικασία ενθυλάκωσης και απενθυλάκωσης*

### *2.3.1.3 Πακέτα IP (IP Packets)*

Τα πακέτα IP είναι τα PDUs του επιπέδου Δικτύου και αποτελούν τον ακρογωνιαίο λίθο κάθε δικτυακής επικοινωνίας. Όλα τα μηνύματα που ανταλλάσσονται μεταξύ δύο σημείων, ανεξάρτητα από το περιεχόμενο ή το πρωτόκολλο που αφορούν, χωρίζονται σε μικρότερα τμήματα πληροφορίας και αποθηκεύονται σε πακέτα. Ο λόγος του συγκεκριμένου κατακερματισμού είναι η δυνατότητα που δίνεται στο δίκτυο και στον αντίστοιχο δικτυακό εξοπλισμό να επεξεργάζεται το κάθε πακέτο ανεξάρτητα από τα υπόλοιπα που συνθέτουν το αρχικό μήνυμα. Με αυτό τον τρόπο είναι εφικτή η πραγματοποίηση παράλληλης επικοινωνίας πολλαπλών σημείων πάνω από το ίδιο μέσο[42].




*Εικόνα 8 - Η μορφή ενός πακέτου IP*

### 2.3.2 Σύνοδος TCP (TCP Session)

Όταν μία συσκευή host χρειάζεται να συνδεθεί με κάποια άλλη χρησιμοποιώντας πρωτόκολλα TCP/IP, ακολουθεί μια διαδικασία εγκαθίδρυσης μιας συνόδου. Πριν την σύνδεση, πραγματοποιείται μια διαδικασία που παρομοιάζεται με χειραψία και ονομάζεται three-way-handshake, κατά την οποία γίνεται η ανταλλαγή ειδικών μηνυμάτων μεταξύ των δύο μερών με σκοπό την εξασφάλιση της συμμετοχής τους στην σύνοδο. Τα πιο σημαντικά είναι τα SYN, για την έναρξη της επικοινωνίας, το FIN, για τον τερματισμό της, και το ACK ως μήνυμα επιβεβαίωσης. Κάθε μήνυμα αναπαρίσταται με μια αντίστοιχη σημαία (flag) η οποία βρίσκεται στο πεδίο TCP ενός πακέτου ενώ κάθε πακέτο ενδέχεται να έχει περισσότερες από μια σημαίες.

Μόλις η επικοινωνία ολοκληρωθεί πραγματοποιείται η ανταλλαγή μηνυμάτων τερματισμού της συνόδου. Μια συσκευή host έχει την δυνατότητα να πραγματοποιεί και να διατηρεί πολλαπλές τέτοιες συνόδους ταυτόχρονα χάρη και στον έλεγχο ροής που παρέχεται από το πρωτόκολλο TCP[43], [44].

# 3 Μεθοδολογία Υλοποίησης: Δημιουργία Σετ Δεδομένων

## 3.1 Εισαγωγή

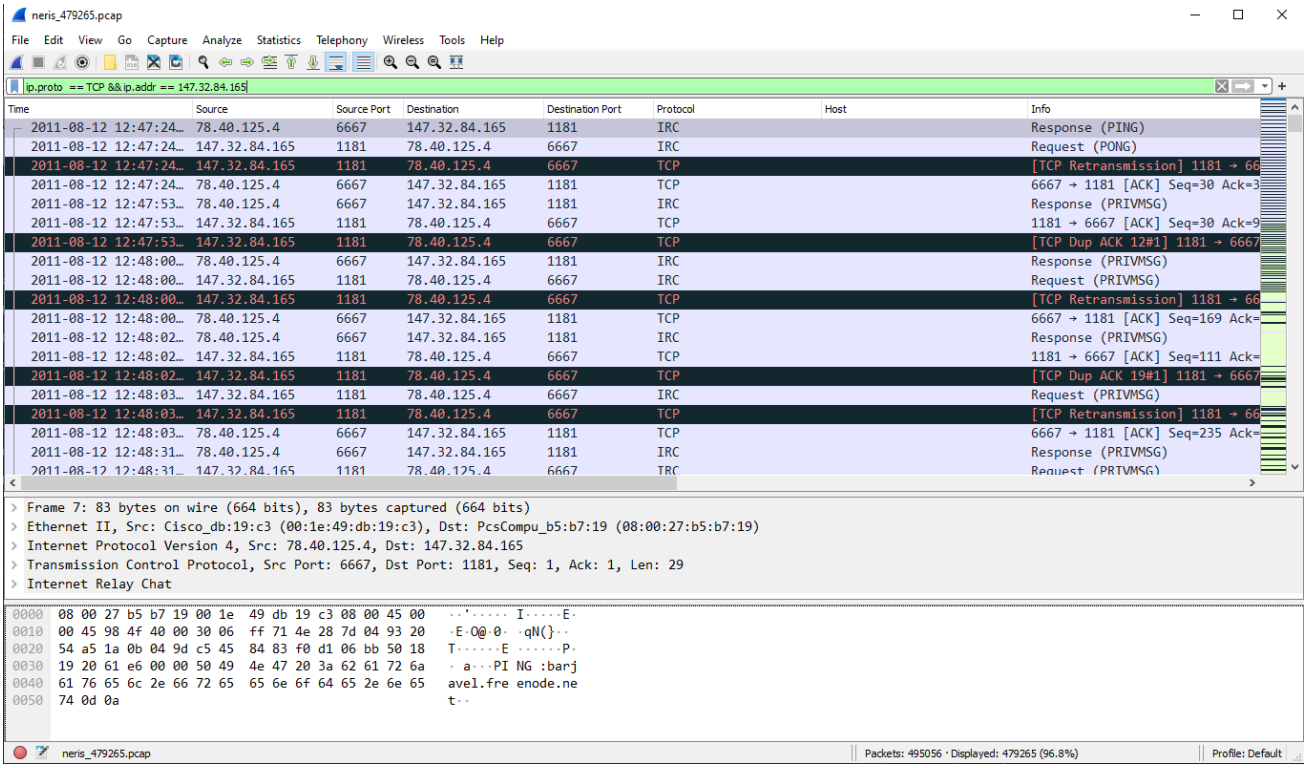
Στο παρόν κεφάλαιο θα ξεκινήσουμε την ανάλυση πάνω στην υλοποίησή μας, ξεκινώντας από το κομμάτι του σετ δεδομένων το οποίο αποτέλεσε τα δεδομένα εισόδου στο μοντέλο μας. Αρχικά θα αναλύσουμε το πώς επιλέξαμε τα αρχικά ακατέργαστα δεδομένα. Στη συνέχεια θα εξηγήσουμε την μεθοδολογία μας για την μορφοποίηση και αποθήκευση των δεδομένων αυτώ. Στο τέλος θα περιγράψουμε τα βήματα μέχρι την δημιουργία του τελικού αρχείου csv.

## 3.2 Ακατέργαστη Πληροφορίας (Raw Data)

Για την δημιουργία του σετ δεδομένων εισόδου, χρησιμοποιήσαμε δικτυακή κίνηση και με σκοπό να καλύψουμε μεγαλύτερο εύρος του χώρου του προβλήματος, συνδυάσαμε τόσο φυσιολογική όσο και κακόβουλη κίνηση. Ενώ για τις ανάγκες της δεύτερης περίπτωση έγινε χρήση υπάρχουσας κίνησης η φυσιολογική επιλέξαμε να παραχθεί από εμάς. Και για τις δύο κατηγορίες κίνησης επιλέξαμε μόνο την κίνηση που αφορούσε στο πρωτόκολλο TCP.

Για τις ανάγκες καταγραφής και αποθήκευσης της φυσιολογικής κίνησης επιλέξαμε το ανοιχτού κώδικα λογισμικό Wireshark[45], το οποίο παράγει ένα αρχείο τύπου pcap[46]. Τέτοιοι τύποι αρχείων περιέχουν πληθώρα μετρικών σχετικών με τα πακέτα IP και τις συνδέσεις που γίνανε ενώ σε αυτά περιλαμβάνεται η πληροφορία για όλα τα επίπεδα του μοντέλου TCP/IP. Η ανάλυση των αρχείων αυτών συμβάλει στην επίλυση προβλημάτων δικτύου αφενός και αφετέρου αποτελεί σημαντική διαδικασία ως μέρος της στρατηγικής κυβερνοασφάλειας ενός Οργανισμού. Σύμφωνα μάλιστα με σχετική έκθεση[47], με την συμμετοχή ιδιωτικών εταιριών από όλο τον κόσμο, η σύλληψη και ανάλυση δικτυακής κίνησης ήταν η πιο κοινή πρακτική που ακολουθούσανε.

Στην παρακάτω εικόνα (Εικόνα 9), παρατίθεται ένα στιγμιότυπο από το αρχείο pcap του Botnet Neris. Το κεντρικό παράθυρο της εφαρμογής χωρίζεται σε τέσσερα διακριτά τμήματα. Το πρώτο αφορά το πεδίο φίλτρων του προγράμματος, όπου μπορούμε να επιλέξουμε την ακριβή πληροφορία που θα εμφανιστεί στο κεντρικό παράθυρο. Εδώ συγκεκριμένα, έχουμε επιλέξει να εμφανιστεί οποιαδήποτε κίνηση περιέχει σαν αφετηρία ή προορισμό την διεύθυνση IP “147.32.84.165” ενώ ζητάμε μόνο την κίνηση που αφορά πακέτα του πρωτοκόλλου TCP. Επιλέγοντας μια οποιαδήποτε γραμμή, στο πιο κάτω τμήμα του παραθύρου εμφανίζεται πληθώρα πληροφοριών που αφορούν την κεφαλίδα του πακέτου, ενώ στο τελευταίο τμήμα εμφανίζεται σε μορφή ASCII τα ωφέλιμο φορτίο(payload) του πακέτου.




*Εικόνα 9 - Στιγμιότυπο από αρχείο pcap με την κακόβουλη κίνηση του Botnet Neris*

### 3.2.1 Επιλογή κακόβουλης κίνησης

Για την κακόβουλη κίνηση δεν χρειάστηκε να δημιουργήσουμε δική μας καθώς υπάρχουν διαθέσιμα dataset που είναι προϊόντα εργαστηριακών ερευνών πάνω στην συμπεριφορά πληθώρας κακόβουλου λογισμικού (botnet, malware, κ.ά.) καθώς και δικτυακή κίνηση από διάφορες εκδηλώσεις όπως hackathons που διοργανώνονται για εκπαιδευτικούς σκοπούς. Επιπλέον οι επιλογές μας βασίστηκαν και στις υπάρχουσες έρευνες που χρησιμοποίησαν κακόβουλη κίνηση για τα δικά τους μοντέλα M.M..

Βασική προϋπόθεση σε κάθε περίπτωση ήταν να υπάρχει τεκμηρίωση του πειράματος όπου να διευκρινίζεται η διεύθυνση IP του επιτιθέμενου ή του μολυσμένου μηχανήματος. Στα περισσότερα από τα προαναφερθέντα εργαστηριακά πειράματα, το διαθέσιμο αρχείο pcap με την καταγεγραμμένη δικτυακή κίνηση περιλάμβανε, πέρα από την κίνηση του κακόβουλου λογισμικού, και φυσιολογική κίνηση. Οπότε ήταν υψηλής σημασίας να γνωρίζουμε πέρα από κάθε αμφιβολία ότι μια επικοινωνία, δηλαδή μια γραμμή στο pcap αρχείο, αντιστοιχεί σε κακόβουλη κίνηση.

Τα βασικά dataset που χρησιμοποιήθηκαν ήταν τα εξής:

- **First**[48]: Σετ δεδομένων για εκπαιδευτικό σκοπό με κακόβουλη κίνηση προερχόμενη από διαφορετικά σενάρια επιθέσεων.
- **CTU-13**[49]: Σετ δεδομένων με κακόβουλη κίνηση προερχόμενη από πολλαπλά διαφορετικά malwares και trojans. Από το συγκεκριμένο σετ χρησιμοποιήσαμε κίνηση από τα Neris και Virut.

### 3.2.3 Επιλογή φυσιολογικής κίνησης

Αρχικά να αναφέρουμε ότι επιλέξαμε η φυσιολογική μας κίνηση να παραχθεί από εμάς ώστε να υπάρχει μεγαλύτερος έλεγχος. Πιο συγκεκριμένα μέσα από διαφορετικά σενάρια χρήσης σκοπό είχαμε να παράγουμε διαφορετικού είδους κίνηση. Τα σενάρια αυτά διαφοροποιούνταν τόσο ως προς το περιβάλλον της κίνησης, σπιτικό και εταιρικό, όσο και ως προς τον ίδιο τον χρήστη που την παρήγαγε εξ ’αρχής. Επιπλέον να σημειώσουμε ότι η φυσιολογική κίνηση που περιέχεται στα pcap της κακόβουλης κίνησης εξαιρέθηκε.

Όπως αναλύσαμε στην υποενότητα 2.2.6.2, ακόμα και αν τηρούνται οι τεχνικές προϋποθέσεις στα δεδομένα εισόδου, αν εκείνα δεν καλύπτουν πλήρως τον χώρο του προβλήματος προς εξέταση, το αποτέλεσμα του μοντέλου θα είναι λάθος. Αυτό συμβαίνει διότι η αρχική γνώση του αλγορίθμου μάθησης είναι ανεπαρκής. Στην περίπτωσή μας, τα δεδομένα εισόδου είναι δικτυακή κίνηση. Αν την κίνηση την παρέχουμε μέσα από ένα μόνο σενάριο χρήσης, το πιο πιθανό είναι ότι το μοντέλο μας να μην μάθει ολοκληρωμένα με αποτέλεσμα προβλέψεις σε νέα δεδομένα (unseen data) να μην έχουνε υψηλή ακρίβεια ή να βγαίνουν λάθος συμπεράσματα.

Για αυτόν ακριβώς τον λόγο, τα αρχεία pcap για την φυσιολογική μας κίνηση, προκύψαν αρχικά από προσωπική μας παραγόμενη κίνηση με καθημερινή οικιακή χρήση, όπως browsing, social media, streaming, καθώς και καθημερινή εταιρική χρήση τόσο on premise όσο και μέσω τηλεργασίας και χρήσης λύσης VPN. Επιπλέον, για τον περιορισμό, αν όχι για την πλήρη αποφυγή, μοτίβων παρόμοιας κίνησης, ζητήσαμε από συναδέλφους να προχωρήσουν στην σύλληψη κίνησης στα δικά τους εταιρικά περιβάλλοντα, θεωρώντας ότι τα προσωπικά θα είχαν πολλές ομοιότητες με την αντίστοιχη δική μας. Στον παρακάτω πίνακα παραθέτουμε στατιστικά σχετικά με την κίνηση ανά σενάριο.

| Dataset (#) | Total TCP Packets | Various Protocols (total packets) | Various Protocols (data size) |
|---|---|---|---|
| 01 | 64684 | HTTP (27193), HTTPS (13521) | HTTP (58,94 MB), HTTPS (33,50) |
| 02 | 51407 | SMB2 (12797), HTTP (1772) HTTPS (11403), POP3/SSL (511), NBSS (6952), LDAP (25), Kerberos (8) | SMB2 (18,15 MB), HTTP (919,33 KB) HTTPS (12,59 MB), POP3/SSL (508,37 KB), NBSS (3,01 MB), LDAP (14,7 KB), Kerberos (10,32 KB) |
| 03 | 74977 | SMB2 (4204), HTTP (5324) HTTPS (26464), POP3/SSL (1234), NBSS (6295), LDAP (417), Kerberos (31) | SMB2 (5,66 MB), HTTP (4,66 MB) HTTPS (17,33 MB), POP3/SSL (1,26 MB), NBSS (1,75 MB), LDAP (400,14 KB), Kerberos (30,37 KB) |
| 04 | IPv4: 104096 | HTTP (216), HTTPS (64161) | HTTP ( 85,54 KB), HTTPS (77,09 MB) |
| | IPv6: 154420 | HTTP (626), HTTPS (97816) | HTTP (272,22 KB), HTTPS (110,60 MB) |
| 05 | 770832 | SMB2 (578156), HTTP (29) HTTPS (13807), POP3/SSL (3281), NBSS (15668), Kerberos (133) | SMB2 (729,37 MB), HTTP (14,22 KB) HTTPS (14 MB), POP3/SSL (3,74 MB), NBSS (6,45 MB), Kerberos (126,82 KB) |

## 3.3 Εξόρυξη Δεδομένων (Data Mining)

Έχοντας ολοκληρώσει την συλλογή της δικτυακής κίνησης, κακόβουλης και μη, σε αυτήν την φάση της υλοποίησής μας θα πρέπει να την αποθηκεύσουμε και να την μορφοποιήσουμε κατάλληλα ώστε να παράγουμε το τελικό αρχείο εισόδου csv. Στην ενότητα αυτή θα αναλύσουμε τον κώδικα σε γλώσσα python με τον οποίο, μέσω της χρήσης συγκεκριμένης βιβλιοθήκης, εξαγάγαμε τα απαραίτητα χαρακτηριστικά του κάθε πακέτου των επί μέρους pcap.

### 3.3.1 Χαρακτηριστικά πακέτων

Όπως αναφέραμε και στην ενότητα 1.2.3, η επιλογή των γνωρισμάτων των παρατηρήσεων μας βασίστηκε και σε υπάρχουσα έρευνα [3]. Παρακάτω παραθέτουμε συνοπτικά τα χαρακτηριστικά αυτά:

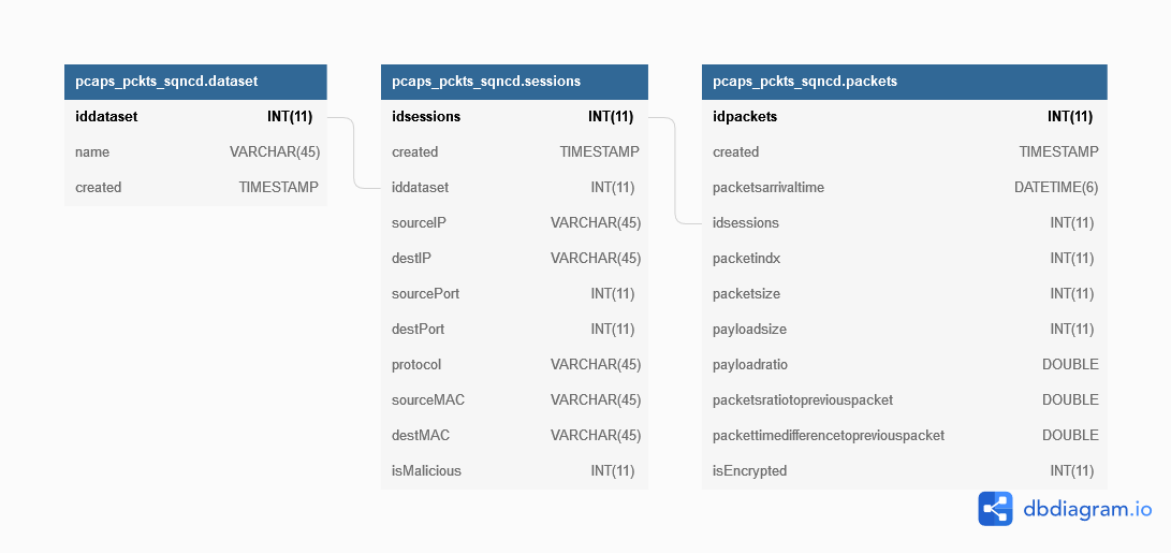
- **Packet Size:** Αφορά το συνολικό μέγεθος ενός IP πακέτου.
- **Payload Size:** Αφορά το μέγεθος του «ωφέλιμου φορτίου». Πρόκειται για μια καλή ένδειξη αν μια κίνηση αποτελεί μέρος επίθεσης.
- **Payload Ratio**: Αφορά την αναλογία του «ωφέλιμου φορτίου» ως προς το συνολικό μέγεθος του πακέτου.
- **Ratio to Previous Packet**: Στα πλαίσια μιας συνόδου TCP (βλ. 2.3.2) τα πακέτα κινούνται σειριακά και συνήθως χαρακτηρίζονται από συγκεκριμένες τάσεις ως προς το μέγεθός τους. Αυτό μπορεί να χρησιμοποιηθεί για τον χαρακτηρισμό κίνησης ως κακόβουλης, απλά συγκρίνοντας δύο πακέτα στην σειρά.
- **Time Difference**: Ξανά στα πλαίσια μια συνόδου TCP και με την ίδια λογική με το ratio to previous packet, μπορεί να εξαχθεί συμπέρασμα για το αν η κίνηση είναι κακόβουλη, από την χρονική διαφορά δύο σειριακών πακέτων. Για το πρώτο πακέτο σε κάθε σύνοδο η τιμή αυτή θα είναι μηδενική.

### 3.3.2 Εξαγωγή και Αποθήκευση της πληροφορίας

Έχοντας παραθέσει τα επιλεγμένα χαρακτηριστικά των πακέτων, στην παρούσα ενότητα θα αναλύσουμε τον κώδικά μας για την εξαγωγή των χαρακτηριστικών αυτών και αποθήκευσή τους σε μια σχεσιακή βάση δεδομένων αφού πρώτα όμως αναφερθούμε στον σχετικό κώδικα για την δημιουργία της βάσης μας.

#### *3.3.2.1 Δημιουργία Σχεσιακής Βάσης*

Όπως αναφέραμε και στο εισαγωγικό κεφάλαιο (ενότητα 1.2.4), δημιουργήσαμε μια σχεσιακή βάση σε MariaDB αποτελούμενη από τρεις συνολικά πίνακες. Το πρόγραμμα για την εξόρυξη και την αποθήκευση της απαραίτητης πληροφορίας, του οποίου την λειτουργία αναλύουμε στις επόμενες ενότητες, δεχόταν κάθε pcap κίνησης ξεχωριστά και τοποθετούσε την πληροφορία αυτή στους πίνακες *sessions* και *packets*. Στην Εικόνα 10 παρουσιάζονται οι εν λόγω πίνακες και οι μεταξύ τους σχέσεις.




*Εικόνα 10 - Το διάγραμμα της βάσης μας με τους τρεις πίνακες και τις σχέσεις τους.*

Πιο συγκεκριμένα έχουμε τους:

pcaps_pckts_sqncd.dataset

Ανάλογα με το dataset που αποθηκεύαμε κάθε φορά προσδίδαμε στο χαρακτηριστικό *dataset.name* το αναγνωριστικό "Normal" ή "Malicious". Αυτός ο διαχωρισμός θα παίξει ρόλο κατά την δημιουργία των επιμέρους csv του σετ δεδομένων μας.

pcaps_pckts_sqncd.sessions

Ο πίνακας προορισμένος για την αποθήκευση διάφορων χαρακτηριστικών της κάθε συνόδου TCP. Αποθηκεύονται πληροφορίες όπως οι IP και MAC διευθύνσεις αφετηρίας και προορισμού το είδος του πρωτοκόλλου, που στην περίπτωσή μας αφορά μόνο το TCP, καθώς και άλλα. Το πιο σημαντικό ωστόσο είναι το πεδίο "isMalicious" το οποίο λαμβάνει τιμές 0 και 1 και χαρακτηρίζει την κίνηση κακόβουλη ή μη και το οποίο θα χρησιμοποιήσουμε στη συνέχεια στα SQL queries για την δημιουργία των επιμέρους csv.

pcaps_pckts_sqncd.packets

Ο πιο σημαντικός πίνακας της βάσης μας καθώς περιλαμβάνει όλα τα γνωρίσματα του μοντέλου μας. Περιέχει τα εξής σημαντικά πεδία:

- **packetsarrivaltime**: Περιέχει το timestamp του πακέτου όπως αυτό παρέχεται από το Wireshark
- **packetindx**: Ένας αύξων αριθμός πακέτου ανά σύνοδο TCP. Στις επόμενες ενότητες θα εξηγήσουμε τον ακριβή ρόλο του.
- **packetsize**: αφορά το μέγεθος του κάθε πακέτου
- **payloadsize**: αφορά το μέγεθος του ωφέλιμου φορτίου του κάθε πακέτου
- **payloadratio**: αποθηκεύει την ποσοστιαία τιμή του πηλίκου μεταξύ των δύο προηγούμενων μεγεθών
- **packetsratiotopreviouspacket**: αφορά την αναλογία μεταξύ των μεγεθών του τρέχοντος πακέτου από το προηγούμενο
- **packettimedifferencetopreviouspacket**: η χρονική διαφορά ανάμεσα από δύο σειριακά πακέτα

- **isEncrypted**: αναγνωριστικό για το αν η κίνηση είναι κρυπτογραφημένη ή όχι

#### *3.3.2.2 Βιβλιοθήκη Scapy*

Στο επίκεντρο του προγράμματος εξόρυξης και αποθήκευσης της απαραίτητης πληροφορίας βρίσκεται η βιβλιοθήκη scapy της python. Στην παρούσα ενότητα θα κάνουμε μια αναφορά στην βιβλιοθήκη και τα σημαντικότερα χαρακτηριστικά της [4].

Ενώ στο πρόγραμμά μας κάνουμε χρήση μόνο βασικές συναρτήσεις της βιβλιοθήκης, στην πραγματικότητα το εγχείρημα γύρω από το scapy αφορά κυρίως ένα ανεξάρτητο πρόγραμμα «πολυεργαλείο». Οι δυνατότητες του προγράμματος καλύπτουν απλή σάρωση και ανάλυση δικτυακής κίνησης μέχρι και δημιουργία κίνησης και την πραγματοποίηση επιθέσεων. Χαρακτηριστικά που συναντώνται σε σουίτες εργαλείων διαφορετικών κατασκευαστών, το scapy τα παρέχει σε ένα μόνο πρόγραμμα ενώ χαρακτηρίζεται από τρομερή ευελιξία καθώς δεν δεσμεύεται από τους παραδοσιακούς κανόνες της θεωρίας Δικτύων ενώ μέσω εύκολου συνδυασμού των έτοιμων βιβλιοθηκών είναι εφικτή η κατασκευή εργαλείων προορισμένων για πιο εξειδικευμένες ενέργειες.

#### *3.3.2.3 Ανάλυση κώδικα python*

Την παρούσα ενότητα θα την αφιερώσουμε στην ανάλυση του κώδικά μας. Με όνομα “Database_Populator”, καθώς στην πράξη “γεμίζει” την βάση δεδομένων *pcaps_pckts_sqncd* που αναφέραμε προηγούμενα, αποτελείται από δύο python αρχεία το *functions.py*, το οποίο εισάγουμε (import) στο κεντρικό μας πρόγραμμα και περιέχει συναρτήσεις που καλούμε, και το *main.py* το οποίο είναι το βασικό. Η χρήση του έξτρα αρχείου, κρίθηκε απαραίτητη για να πετύχουμε καθαρότερο κώδικα και μεγαλύτερη ευελιξία. Ακολουθεί σχετική ανάλυση.

**functions.py**

Στο συγκεκριμένο αρχείο, βρίσκεται ο κώδικα τόσο για την ανάγνωση των αρχείων pcap όσο και για την αναγνώριση της κακόβουλης κίνησης.

- Libraries Imports

```
1  import logging
2
3  logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
4  from scapy.all import rdpcap, types, PcapReader, RawPcapReader
5  from scapy import plist
6  import time
7  import errno
8  import os
9  import fnmatch
10 import hashlib
```

- retrieve_all_files

```
1 def retrieve_all_files(root):
2    fi = []
3    for path, dirs, files in os.walk(root):
4       for name in files:
```

```
5           fi.append(os.path.join(path, name))
6    fi.sort()
7    return fi
```

Λαμβάνει ως είσοδο το path στο οποίο βρίσκονται τα αρχεία pcap και τα επιστρέφει οργανωμένα σε μια δομή δεδομένων λίστας.

- rdpcap_and_close

```
1 def rdpcap_and_close(filename, count=-1):
2    pcap_reader = PcapReader(filename)
3    packets = pcap_reader.read_all(count=count)
4    pcap_reader.close()
5    return packets
```

Πρόκειται για την βασική μας συνάρτηση που αναλαμβάνει να διαβάσει το αρχειο pcap, που λαμβάνει ως είσοδο μέσα από επανάληψη, όπως θα δούμε στην ανάλυση του κυρίως προγράμματος, και επιστρέφει μια λίστα με τα περιεχόμενα πακέτα.

- is_First

```
1  def is_First(session_title):
2
3     if "217.195.49" in session_title:
4        return True
5     elif "68.164.182.11" in session_title:
6        return True
7     elif "193.9.28.35" in session_title:
8        return True
9     elif "148.251.80.172" in session_title:
10       return True
11    elif "68.164.182.11" in session_title:
12       return True
13    elif "216.47.227.188" in session_title:
14       return True
15    elif "209.59.156.160" in session_title:
16       return True
17    elif "103.10.197.187" in session_title:
18       return True
19    else:
20       return False
```

Όπως αναφέραμε στην ενότητα 3.2.2, βασική προϋπόθεση για την επιλογή της κακόβουλης κίνησης ήταν να υπάρχει σχετική τεκμηρίωση με την ακριβή διεύθυνση IP του μολυσμένου μηχανήματος. Ο λόγος είναι εμφανής στην παρούσα αλλά και στις επόμενες συναρτήσεις αναγνώρισης. Βασικό κριτήριο ανάλογα με το dataset ήταν συγκεκριμένες IP που βάσει της εκάστωτε τεκμηρίωσης ανήκαν στα μολυσμένα μηχανήματα του πειράματος. Από εκεί και πέρα μια απλή if συνθήκη αναλαμβάνει να ελέγξει κατά πόσον μία από τις IP βρίσκεται στον τίτλο της συνόδου TCP. Ως σημείο αναφοράς, ένα παράδειγμα τίτλου μιας συνόδου TCP είναι

το «*TCP 192.168.1.53:52385 > 185.141.195.36:80*» όπου διακρίνεται η διεύθυνση και πόρτα αφετηρίας και η διεύθυνση και πόρτα προορισμού. Την ίδια λογική ακολουθούν και υπόλοιπες συναρτήσεις.

- is_Trickbot

```
def is_Trickbot(session_title):
    if "77.236.96.52" in session_title:
        return True
    elif "88.150.140.232" in session_title:
        return True
    elif "23.254.97.211" in session_title:
        return True
    elif "194.87.144.27" in session_title:
        return True
    elif "194.87.232.127" in session_title:
        return True
    elif "94.242.224.226" in session_title:
        return True
    elif "185.158.115.47" in session_title:
        return True
    else:
        return False
```

- is_Neris

```
def is_Neris(session_title):
    # type: (str) -> bool

    if "147.32.84.165" in session_title:
        return True
    else:
        return False
```

**main.py**

Πρόκειται για το βασικό μας πρόγραμμα εξόρυξης δεδομένων το οποίο, αφού πρώτα πραγματοποιήσει επιτυχής σύνδεση με την βάση *pcaps_pckts_sqncd*, ξεκινάει να διαβάζει τα pcaps με την κίνησή μας. Εδώ να σημειώσουμε ότι ενώ η συνάρτηση *retrieve_all_files* επιτρέπει την ανάγνωση πολλαπλών αρχείων, στις περιπτώσεις μεγάλων σε μέγεθος pcap, προτιμήθηκε η εισαγωγή ενός αρχείου την φορά καθώς το κόστος σε πόρους συστήματος ήταν υψηλός.

Πριν περάσουμε στην ανάλυση του κώδικά μας, είναι χρήσιμο σε αυτό το σημείο να εξηγήσουμε τον γενικό μηχανισμό. Η βιβλιοθήκη scapy παρέχει όλες τις απαραίτητες βιβλιοθήκες και συναρτήσεις ώστε να γίνει πλήρης ανάλυση ενός πακέτου λαμβάνοντας υπόψη όλα τα TCP/IP layers. Έπειτα η οργάνωση της πληροφορίας χωρίζεται σε συνόδους οι οποίες περιέχουν πακέτα. Οπότε για το κάθε pcap χρησιμοποιούμε μια συνάρτηση που διατρέχει τα sessions και άλλη μια που διατρέχει τα πακέτα σε κάθε session ενώ μια κεφαλική συνάρτηση διατρέχει τα pcap αρχεία για τις περιπτώσεις περισσότερων του ενός μικρών αρχείων.

- Libraries Imports

```
1  from __future__ import division
2
3  import datetime
4
5  from functions import *
6  from decimal import Decimal, getcontext
7  import gc
8  from mysql.connector import errorcode
9  import mysql.connector as mariadb
10 import time
11 import logging
12 from scapy_ssl_tls.ssl_tls import *
```

- Logging

```
1 logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
```

Με αυτό τον τόπο αποφεύγουμε τις χαμηλού κινδύνου ειδοποιήσεις του scappy runtime.

- Database connection & Exception Handling

```
1  config = {
2      'host': '127.0.0.1',
3      'user': 'ml_user',
4      'password': 'ml_user',
5      'database': 'pcaps_pckts_sqncd'
6  }
7
8  try:
9      conn = mariadb.connect(**config)
10     print("Connection Established")
11 except mariadb.Error as err:
12     if err == errorcode.ER_ACCESS_DENIED_ERROR:
13         print("Something is wrong with the user name or password")
14     elif err.errno == errorcode.ER_BAD_DB_ERROR:
15         print("Database does not exist")
16     else:
17         print(err)
```

Ο κώδικας για την σύνδεση με την βάση μας. Αφού ορίσουμε τα στοιχεία σύνδεσης, όνομα βάσης και στοιχεία αυθεντικοποίησης, πραγματοποιείται διαχείριση εξαίρεσης σε περίπτωση αποτυχίας σύνδεσης ώστε να παραχθεί το κατάλληλο μήνυμα.

- Καταχώρηση στον πίνακα dataset

```
1 else:
2     print('begin')
3     cursor = conn.cursor()
4     cursor.execute(
5         "INSERT INTO dataset (name) values (%s);", ("Normal",))
```

```
6    datasetid = cursor.lastrowid
7    conn.commit()
```

Στο παραπάνω κομμάτι κώδικα, ανάλογα με την κίνηση που αποθηκεύουμε εισάγουμε μια καταχώρηση στον πίνακα dataset με το όνομα “Normal” ή “Malicious”.

- Επανάληψη επί των pcap αρχείων

```
1  rt = '/home/eukokol/Desktop/_norm_traf/containing_folder'
2
3    for fi in retrieve_all_files(rt):
4      print(fi)
5
6      start = time.time()
7
8      a = rdpcap_and_close(fi)
9      end = time.time()
10     print(end - start)
11     sessions = a.sessions()
12     print('Loop All Sessions')
```

Πρόκειται για την επανάληψη επί των pcap αρχείων. Αποθηκεύουμε την διαδρομή του φακέλου που περιέχει τα pcap μας στην μεταβλητή *rt* η οποία στην συνέχεια δίνεται ως είσοδος στην συνάρτηση *retrieve_all_files*. Υπολογίζεται η χρονική διάρκεια της διαδικασίας ανοίγματος του κάθε pcap για στατιστικού λόγους κυρίως ώστε να γνωρίζουμε την σχέση του χρόνου με το μέγεθος του κάθε pcap. Από την λίστα των πακέτων του pcap εξάγουμε τις συνολικές συνόδους και τις αποθηκεύουμε στην μεταβλητή *sessions*.

- Επανάληψη επί των συνόδων

```
1      for session in sessions:
2        if session.title().startswith('Tcp'):
3          if is_Neris(session.title()) or is_First(session.title()) or
4          is_Milicenso(session.title()) or is_Trickbot(session.title()):
5            isMalicious = 1
6          else:
7            isMalicious = 0
8          print('Malicius: ', isMalicious)
9          sid = 0
10         pid = 0
11         previous_packet_size = 0
12         srcIP = ""
13         dstIP = ""
14         srcPort = 0
15         dstPort = 0
16         previous_ratio = 0
17         previous_packet_time = 0
18         packetindex = 0
19
20         total_packet_size = 0
21         total_payload_size = 0
22         total_payload_ratio = 0
23         total_ratio_to_previous = 0
24         total_time_difference = 0
```

```
25
26          srcMAC = sessions[session][0]["Ethernet"].fields["src"]
27          dstMAC = sessions[session][0]["Ethernet"].fields["dst"]
28
29          if sessions[session][0].haslayer("IP"):
30              srcIP = sessions[session][0]["IP"].fields["src"]
31              dstIP = sessions[session][0]["IP"].fields["dst"]
32          else:
33              srcIP = sessions[session][0]["IPv6"].fields["src"]
34              dstIP = sessions[session][0]["IPv6"].fields["dst"]
35
36          srcPort = sessions[session][0]["TCP"].fields["sport"]
37          dstPort = sessions[session][0]["TCP"].fields["dport"]
38
39          protocol = "TCP"
40
41          cursor.execute(
42              "INSERT INTO sessions (iddataset, sourceIP, destIP, sourcePort, destPort,\
43              protocol, sourceMAC, destMAC, isMalicious) \
44              values (%s, %s,%s, %s,%s, %s,%s, %s,%s);",
45              (datasetid, srcIP, dstIP, srcPort, dstPort, "TCP", srcMAC, dstMAC,
46          isMalicious))
47
48          sessionid = cursor.lastrowid
```

- Η πρώτη μας συνθήκη στην αρχή της επανάληψης αφορά στην αποκλειστική επιλογή των συνόδων TCP. Όπως αναφέραμε στην προηγούμενη παράγραφο η συνάρτηση *sessions()*, αφορά συνόδους για όλα τα διαφορετικά πρωτόκολλα. Στα πλαίσια της δικής μας υλοποίησης, ωστόσο, μας ενδιαφέρουν μόνο τα TCP.

- Η επόμενη συνθήκη αφορά την εξακρίβωση αν μια σύνοδος αφορά πακέτα κακόβουλα. Όπως αναφέραμε στην ενότητα 3.2.2, σε μερικά σετ δεδομένων της κακόβουλης κίνησης υπήρχε και φυσιολογική κίνηση. Οπότε στα πλαίσια μιας καταχώρησης Malicious στον πίνακα dataset της βάσης μας, θα έπρεπε να υπήρχε ένα τρόπος να επιλέξουμε μόνο την κακόβουλη κίνηση. Με την μεταβλητή flag *isMalicious* πετυχαίνουμε ακριβώς αυτό.

- Στη συνέχεια γίνεται η αρχικοποίηση μεταβλητών για την αποθήκευση της αντίστοιχης πληροφορίας στην βάση μας και πιο συγκεκριμένα στους πίνακες *sessions* και *packets*. Στο σώμα της συγκεκριμένης επανάληψης, γίνεται η εξαγωγή εκείνης της πληροφορίας που αφορά την σύνοδο TCP, που σημαίνει όλες οι σχετικές διευθύνσεις δικτύου και υλικού καθώς και οι αντίστοιχες πόρτες ενώ γίνεται πρόβλεψη τόσο για IPv4 κίνηση όσο και για IPv6.

- Τέλος, το SQL query αναλαμβάνει την εισαγωγή της πληροφορίας στην βάση.

▪ Επανάληψη επί των πακέτων της κάθε συνόδου

Πριν προχωρήσουμε στην ανάλυση του κώδικά μας , θεωρήσαμε χρήσιμο να παραθέσουμε ένα παράδειγμα στιγμιότυπου από την ανάλυση σε ένα πακέτο:

```
2022-02-11 10:14:49.270738
###[ Ethernet ]###
  dst       = 08:00:27:be:20:60
  src       = 52:54:00:12:35:00
  type      = IPv4
###[ IP ]###
     version   = 4L
     ihl       = 5L
     tos       = 0x0
     len       = 628
     id        = 60454
     flags     =
     frag      = 0L
     ttl       = 255
     proto     = tcp
     chksum    = 0x79db
     src       = 142.250.184.131
     dst       = 10.0.2.4
     \options   \
###[ TCP ]###
        sport     = https
        dport     = 45664
        seq       = 55819
        ack       = 1906590105
        dataofs   = 5L
        reserved  = 0L
        flags     = PA
        window    = 31356
        chksum    = 0x421f
        urgptr    = 0
        options   = []
###[ SSL/TLS ]###
           \records   \
            |###[ SSLv2 Record ]###
            |  length    = 0x4e77
            |  content_type= 125
            |###[ Raw ]###
            |     load      = '\'\x03\x8bk\xcc\x14\x9f\xa3
\xd2\xfad2Y\xcc\x13w\xc7W\x83A\xeb/q]S0\xbb\xbc
\x14\xfd\x06P\xae\xb4\t\xb9\xe9\x89c\
x17\x10\xd6\xefi\x89\x9en{\xed\x0e\x14\xeb\x82k,
Lj\x7f\xcf\xaaR\xe3-\x80~\xab\xaf8\x89\x19i\x8aG\x07
\x98!\x8bZ\x92\xde\x88"\x81\xbcbW\x81F\x8bsHLw\xab
\x08\xd9\xea\xa1\xe1\x04\x1d\x81r+\x82\xaaH\xd5\xa30....'
```

Έχοντας σχηματισμένη την εικόνα του μοντέλου TCP/IP, παρατηρούμε ότι η ανάλυση ξεκινάει από τα κατώτερα επίπεδα, όπου διακρίνονται οι πληροφορίες των φυσικών διευθύνσεων και διευθύνσεων Δικτύου, και καταλήγει στο επίπεδο Μεταφοράς με πληροφορίες για το πρωτόκολλο TCP. Στο τέλος βλέπουμε κεφαλίδα "*SSL/TLS*", που υποδεικνύει ότι το πακέτο είναι κρυπτογραφημένο, και μετά την ετικέτα *"Raw"* βλέπουμε το ωφέλιμο φορτίο του πακέτου (payload) κρυπτογραφημένο. Για οικονομία χώρου δεν παρατίθεται το σύνολο του περιεχομένου του payload.

```
1           packetquery = """INSERT INTO packets(packetsarrivaltime, idsessions,
2           packetindx,packetsize, payloadsize, payloadratio,
3           packetsratiotopreviouspacket, packettimedifferencetopreviouspacket,
```

```
4           isEncrypted)
5           values (%s, %s, %s, %s, %s,%s, %s, %s, %s);"""
6           values = []
7
8           for packet in sessions[session]:
9               packet_size = len(packet)
10
11              packetindex += 1
12
13              isEncrypted = 0
14              counter = 0
15              while True:
16                  layer = packet.getlayer(counter)
17                  if layer is not None:
18                      if "SSL/TLS" in layer.name:
19                          isEncrypted = 1
20                  else:
21                      break
22                  counter += 1
23
24              if packet.haslayer("Raw"):
25                  payload_size = len(packet["Raw"])
26
27                  # Calculate Payload Ratio
28                  payload_ratio = 100 * (payload_size / packet_size)
29              else:
30                  payload_size = 0
31                  payload_ratio = 0
32
33              # Calculate Ratio compared to previous packet
34              if previous_packet_size != 0:
35                  previous_ratio = (packet_size / previous_packet_size) * 100
36                  previous_packet_size = packet_size
37              else:
38                  previous_ratio = "1"
39                  # Set size for next packet
40                  previous_packet_size = packet_size
41
42              # Calculate time between packets
43              if previous_packet_time != 0:
44                  packet_time_difference = Decimal(
45                      packet.time) - Decimal(previous_packet_time)
46              else:
47                  packet_time_difference = 0
48                  previous_packet_time = Decimal(packet.time)
49
50              t = packet.time
51              packet_time_conv = datetime.datetime.fromtimestamp(t).strftime
52              ('%Y-%m-%d %H:%M:%S.%f')
53
54              values.append((packet_time_conv, sessionid, packetindex,
55                  packet_size, payload_size, payload_ratio, previous_ratio,
56                          packet_time_difference, isEncrypted))
```

- SQL αίτημα για τον πίνακα *packets*

  Ξεκινώντας, και λίγο πριν τον κώδικα της επανάληψης, φτιάχνουμε το SQL query μας και αρχικοποιούμε μια λίστα όπου θα αποθηκεύσουμε την πληροφορία μας καθώς θα συλλέγεται από όλα τα πακέτα του κάθε session.

- packetsize, packetindx
  Αρχικά υπολογίζουμε το μέγεθος του πακέτου και αυξάνουμε τον δείκτη packetindex.
- Έλεγχος για την ύπαρξη SSL/TLS πεδίου. Όπως αναφέραμε είναι υπόδειξη κρυπτογραφημένου πακέτου.
- payload_size & payload_ratio
  Έλεγχος αρχικά ύπαρξης της κεφαλίδας *"Raw"*, που υποδεικνύει ύπαρξη payload, και στην συνέχεια υπολογισμού του μεγέθους του payload καθώς και της αναλογίας ως προς το μέγεθος του πακέτου που υπολογίστηκε στο προηγούμενο βήμα.
- packetsratiotopreviouspacket
  Εδώ υπολογίζουμε την αναλογία του μεγέθους του τρέχοντος πακέτου με το προηγούμενο. Απαραίτητος έλεγχος για το αν πρόκειται για το πρώτο πακέτου του session, περίπτωση στην οποία θέτουμε την αναλογία να ισούται με '1'.
- packettimedifferencetopreviouspacket
  Υπολογισμός της χρονικής διαφοράς του τρέχοντος από το προηγούμενο. Και εδώ είναι σημαντικό να ελέγξουμε για το αν πρόκειται για την αρχή της συνόδου, οπότε και θέτουμε την τιμή '0'.

- Commit και έξοδος

```
1            cursor.executemany(packetquery, values)
2            conn.commit()
3
4        sessions.clear()
5        del sessions
6        del a
7        gc.collect()
8        end = time.time()
9        conn.commit()
```

Στο τελικό κομμάτι του κώδικά μας, εκτελείται το Insert για τις πληροφορίες που θα αποθηκευτούν στον πίνακα *packets* ενώ εκτελούνται και οι δύο εντολές για το commit στην βάση μας. Τέλος πραγματοποιείται εκκαθάριση της μνήμης.

### 3.3.3 Μορφοποίηση της πληροφορίας

Έχοντας αποθηκευμένη την ακατέργαστη πληροφορία μας στην βάση δεδομένων, το επόμενο βήμα είναι η εξαγωγή της και η δημιουργία του τελικού σετ δεδομένων. Πριν ωστόσο προχωρήσουμε στα βήματα για την δημιουργία του dataset, είναι υψηλής σημασίας να εξηγήσουμε το βασικό μας κριτήριο πάνω στο οποίο βασίστηκε όλη η μεθοδολογία μας και τις επιλογές που πάρθηκαν.

#### *3.3.3.1 Σημασία Συνόδων TCP*

Όπως αναφέραμε και στην προηγούμενη ενότητα, όπου αναλύσαμε το πρόγραμμα εξόρυξης δεδομένων, τα πακέτα ομαδοποιούνται σε συνόδους TCP. Οι τιμές των γνωρισμάτων που επιλέξαμε για το μοντέλο μας, υπολογίζονται στα στενά πλαίσια της κάθε συνόδου TCP και ανεξάρτητα από τα υπόλοιπα πακέτα του pcap. Πιο συγκεκριμένα, παρατηρούμε το επαναλαμβανόμενο μοτίβο, όπου τα γνωρίσματα

*packetsratiotopreviouspacket* και *packettimedifferencetopreviouspacket* του πρώτου πακέτου κάθε συνόδου είναι 1 και 0 αντίστοιχα.

Είναι εμφανές ότι η παραπάνω συμπεριφορά είναι υψηλής σημασίας και θα πρέπει να χαρακτηρίζει και τα αποτελέσματα των προβλέψεών μας. Συνεπώς, και με σκοπό το μοντέλο μας να εκπαιδευτεί βάσει του παραπάνω μοτίβου, πραγματοποιήσαμε ταξινόμηση στα πακέτα βάσει των πεδίων *idsessions* και *packetindex*. Στην παρακάτω εικόνα φαίνεται ένα στιγμιότυπο από dataset φυσιολογικής κίνησης με δύο διαδοχικές συνόδους TCP, *idsessions* «1003» και «1004», όπου παρατηρούμε την συμπεριφορά που περιγράψαμε πιο πάνω.

| packetsarrivaltime | idsessions | packetindx | payloadsize | payloadratio | packetsratiotopreviouspacket | packettimedifferencetopreviouspacket | isMalicious | isEncrypted |
|---|---|---|---|---|---|---|---|---|
| 2022-02-11 10:14:50.154819 | 1,015 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:14:50.200933 | 1,015 | 2 | 0 | 0 | 72.97297297297297 | 0.046113967892550781 | 0 | 0 |
| 2022-02-11 10:14:50.201346 | 1,015 | 3 | 373 | 87.35362997658079 | 790.7407407407408 | 0.046526908874511719 | 0 | 0 |
| 2022-02-11 10:14:50.301784 | 1,015 | 4 | 0 | 0 | 12.646370023419204 | 0.14696502685546875 | 0 | 0 |
| 2022-02-11 10:15:00.301426 | 1,015 | 5 | 0 | 0 | 100 | 10.146606922149658 | 0 | 0 |
| 2022-02-11 10:15:10.541452 | 1,015 | 6 | 0 | 0 | 100 | 20.386632919311523 | 0 | 0 |
| 2022-02-11 10:15:20.784228 | 1,015 | 7 | 0 | 0 | 100 | 30.629409074783325 | 0 | 0 |
| 2022-02-11 10:15:31.021410 | 1,015 | 8 | 0 | 0 | 100 | 40.866590976715090 | 0 | 0 |
| 2022-02-11 10:15:41.261635 | 1,015 | 9 | 0 | 0 | 100 | 51.1068160533905 | 0 | 0 |
| 2022-02-11 10:15:51.501393 | 1,015 | 10 | 0 | 0 | 100 | 61.346574068069460 | 0 | 0 |
| 2022-02-11 10:16:01.741431 | 1,015 | 11 | 0 | 0 | 100 | 71.586611986160280 | 0 | 0 |
| 2022-02-11 10:16:11.981493 | 1,015 | 12 | 0 | 0 | 100 | 81.826673984527590 | 0 | 0 |
| 2022-02-11 10:16:22.222247 | 1,015 | 13 | 0 | 0 | 100 | 92.067427873611450 | 0 | 0 |
| 2022-02-11 10:16:32.461830 | 1,015 | 14 | 0 | 0 | 100 | 102.30701088905334 | 0 | 0 |
| 2022-02-11 10:16:42.701427 | 1,015 | 15 | 0 | 0 | 100 | 112.54660797111940 | 0 | 0 |
| 2022-02-11 10:16:45.304980 | 1,015 | 16 | 0 | 0 | 100 | 115.15016102790833 | 0 | 0 |
| 2022-02-11 10:16:45.352023 | 1,015 | 17 | 0 | 0 | 100 | 115.19720387458801 | 0 | 0 |
| 2022-02-11 10:11:55.049793 | 1,016 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 2022-02-11 10:11:55.098534 | 1,016 | 2 | 0 | 0 | 58.06451612903226 | 0.048741102218627930 | 0 | 0 |
| 2022-02-11 10:12:47.680289 | 1,016 | 3 | 0 | 0 | 172.22222222222223 | 52.63049602508545 | 0 | 1 |
| 2022-02-11 10:12:47.684475 | 1,016 | 4 | 0 | 0 | 83.87096774193549 | 52.634681940078735 | 0 | 1 |
| 2022-02-11 10:12:47.684541 | 1,016 | 5 | 0 | 0 | 69.23076923076923 | 52.634747982025150 | 0 | 0 |
| 2022-02-11 10:12:47.734433 | 1,016 | 6 | 0 | 0 | 100 | 52.684639930725100 | 0 | 0 |
| 2022-02-11 10:14:03.940078 | 1,017 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:14:03.998301 | 1,017 | 2 | 0 | 0 | 72.97297297297297 | 0.058223009109497070 | 0 | 0 |
| 2022-02-11 10:14:04.004146 | 1,017 | 3 | 107 | 18.739054290718038 | 1,057.4074074074074 | 0.064068078994750980 | 0 | 1 |
| 2022-02-11 10:14:04.096772 | 1,017 | 4 | 0 | 0 | 9.457092819614711 | 0.15669339353942871 | 0 | 0 |
| 2022-02-11 10:14:04.098102 | 1,017 | 5 | 0 | 0 | 100 | 0.15802407264709473 | 0 | 0 |
| 2022-02-11 10:14:04.519325 | 1,017 | 6 | 0 | 0 | 218.5185185185185 | 0.579246997833252 | 0 | 1 |
| 2022-02-11 10:14:04.520252 | 1,017 | 7 | 0 | 0 | 189.83050847457628 | 0.58017396926879880 | 0 | 1 |
| 2022-02-11 10:14:04.520300 | 1,017 | 8 | 0 | 0 | 459.375 | 0.58022189140319820 | 0 | 1 |
| 2022-02-11 10:14:04.526141 | 1,017 | 9 | 0 | 0 | 8.649173955296405 | 0.58606290817260740 | 0 | 1 |
| 2022-02-11 10:14:04.568598 | 1,017 | 10 | 0 | 0 | 60.67415730337079 | 0.62852001190185550 | 0 | 0 |
| 2022-02-11 10:14:04.569084 | 1,017 | 11 | 0 | 0 | 157.40740740740742 | 0.62900590896606450 | 0 | 1 |
| 2022-02-11 10:14:46.536376 | 1,017 | 12 | 0 | 0 | 278.8235294117647 | 42.596297979354860 | 0 | 1 |
| 2022-02-11 10:14:46.650756 | 1,017 | 13 | 0 | 0 | 22.78481012658228 | 42.710677862167360 | 0 | 0 |
| 2022-02-11 10:14:46.657466 | 1,017 | 14 | 0 | 0 | 100 | 42.717387914657590 | 0 | 0 |
| 2022-02-11 10:14:46.674468 | 1,017 | 15 | 0 | 0 | 172.22222222222223 | 42.734390020370480 | 0 | 1 |
| 2022-02-11 10:14:47.061911 | 1,017 | 16 | 0 | 0 | 201.07526881720430 | 43.121833086013794 | 0 | 1 |
| 2022-02-11 10:14:47.066082 | 1,017 | 17 | 0 | 0 | 78.60962566844920 | 43.126003938063660 | 0 | 1 |
| 2022-02-11 10:14:47.079699 | 1,017 | 18 | 0 | 0 | 99.31972789115646 | 43.139621019363400 | 0 | 1 |
| 2022-02-11 10:14:47.107351 | 1,017 | 19 | 0 | 0 | 98.63013698630137 | 43.167273044586180 | 0 | 1 |

*Εικόνα 11 - Στιγμιότυπο από εκτελεσμένο query στη βάση μας, όπου φαίνονται τρεις σύνοδοι TCP ταξινομημένοι σειριακά.*

### *3.3.3.2 Βασικές Επιλογές*

Έχοντας εξηγήσει την λογική πίσω από την διατήρηση της δομής των συνόδων TCP, σε αυτή την υποενότητα θα αναλύσουμε σύντομα σημαντικές επιλογές που πάρθηκαν βάσει αυτής την λογικής.

**Μη επιλογή ολόκληρης συνόδου**

Εφόσον αποφασίσαμε να διατηρήσουμε την δομή των συνόδων TCP, η χρήση οποιασδήποτε τυχαίας μεθόδου επιλογής πακέτων δεν ήταν εφικτή καθώς θα χαλούσε την δομή αυτή. Αυτό σήμαινε ότι, τουλάχιστον η αρχή των περιεχόμενων συνόδων έπρεπε να υπάρχει. Ωστόσο, για την εκπαίδευση του μοντέλου μας δεν ήταν απαραίτητη η διατήρηση ολόκληρης της συνόδου καθώς οι τιμές των γνωρισμάτων μας για το κάθε πακέτο της κάθε συνόδου, εκτός του πρώτου, εξαρτώνται από το αμέσως προηγούμενο. Για να γίνει πιο ξεκάθαρη η ιδέα

αυτή, παρατηρώντας την εικόνα της προηγούμενης ενότητας, αν υποθέσουμε ότι τη σύνοδο με αναγνωριστικό "1017" την αποθηκεύσουμε στο επί μέρους dataset έως το πακέτο με αναγνωριστικό "12", η εκπαίδευση του μοντέλου μας θα παραμείνει ανεπηρέαστη, καθώς οι τιμές των γνωρισμάτων δεν εξαρτώνται από τα πακέτα που δεν θα συμπεριληφθούν.

**Κρυπτογραφημένη κίνηση**

Η διατήρηση της δομής, ωστόσο, επηρέασε και μια άλλη παράμετρο της ακατέργαστης πληροφορίας που ήταν η κρυπτογραφημένη κίνηση. Στην Εικόνα 12 και συγκεκριμένα στην τελευταία στήλη, που υποδεικνύει την ύπαρξη κρυπτογράφησης, παρατηρούμε ότι σε μία σύνοδο TCP δύναται να υπάρχουν τόσο κρυπτογραφημένα όσο και μη πακέτα. Αυτό οφείλεται στα σήματα σηματοδοσίας του πρωτοκόλλου TCP τα οποία δεν είναι κρυπτογραφημένα σε αντίθεση με την πληροφορία που ανταλλάσσεται. Συνεπώς, το κριτήριο της ύπαρξης ή όχι κρυπτογράφησης, δεν χρησιμοποιήθηκε κατά την δημιουργία του σετ.

**Δημιουργία δύο σετ δεδομένων**

Τέλος, θέλοντας να καλύψουμε όσο μεγαλύτερο εύρος από το σενάριο φυσιολογικής κίνησης ήταν εφικτό, αποφασίσαμε να εξάγουμε δύο υποσύνολα πακέτων από το κάθε σενάριο με την προϋπόθεση και τα δύο να ξεκινούν με την αρχή μιας συνόδου TCP. Επιπλέον το πόσο αργότερα μέσα στο dataset μας θα γινόταν η επιλογή του δεύτερου υποσυνόλου ήταν άνευ σημασίας καθώς η χρονική σειρά των πακέτων δεν σχετίζεται με την σειρά των αποθηκευμένων συνόδων. Στην Εικόνα 12, παρατίθεται ένα στιγμιότυπο από dataset φυσιολογικής κίνησης ταξινομημένο βάσει της χρονοσήμανσης του κάθε πακέτου. Διακρίνουμε ότι, το πρώτο χρονικά TCP πακέτο του dataset ανήκει στην σύνοδο με αναγνωριστικό «1233» ενώ το αμέσως επόμενο στην σύνοδο «1028». Αυτό δείχνει ότι οι επιλογή των δύο υποσυνόλων με τον τρόπο που περιγράψαμε δεν δημιουργεί πρόβλημα συγκέντρωσης κίνησης από συγκεκριμένο χρονικό σημείο καταγραφής.

| packetsarrivaltime | idsessions | packetindx | payloadsize | payloadratio | packetsratiotopreviouspacket | packettimedifferencetopreviouspacket | isMalicious | isEncrypted |
|---|---|---|---|---|---|---|---|---|
| 2022-02-11 10:11:08.621428 | 1,233 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:08.621690 | 1,028 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:08.877450 | 1,212 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:08.877602 | 1,176 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:09.389394 | 1,179 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:09.389445 | 1,037 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:09.389590 | 1,026 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:09.389590 | 1,027 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:09.889683 | 1,173 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 2022-02-11 10:11:09.889735 | 1,173 | 2 | 0 | 0 | 69.23076923076923 | 0.0000519752502441406 25 | 0 | 0 |
| 2022-02-11 10:11:09.890119 | 1,286 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:09.922695 | 1,286 | 2 | 0 | 0 | 100 | 0.03257584571838379 | 0 | 0 |
| 2022-02-11 10:11:09.922721 | 1,173 | 3 | 0 | 0 | 100 | 0.03303790092468262 | 0 | 0 |
| 2022-02-11 10:11:10.157657 | 1,129 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:10.157836 | 1,007 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:10.406528 | 1,107 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 2022-02-11 10:11:10.406739 | 1,107 | 2 | 0 | 0 | 13.428120063191153 | 0.0002110004425048828 | 0 | 1 |
| 2022-02-11 10:11:10.406952 | 1,084 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2022-02-11 10:11:10.497064 | 1,084 | 2 | 0 | 0 | 265 | 0.09011220932006836 | 0 | 1 |
| 2022-02-11 10:11:10.497089 | 1,107 | 3 | 0 | 0 | 63.52941176470588 | 0.0905609130859375 | 0 | 0 |
| 2022-02-11 10:11:10.498035 | 1,084 | 3 | 0 | 0 | 58.490566037735846 | 0.09108304977416992 | 0 | 1 |
| 2022-02-11 10:11:10.498061 | 1,107 | 4 | 0 | 0 | 100 | 0.09153294563293457 | 0 | 0 |
| 2022-02-11 10:11:10.500219 | 1,107 | 5 | 0 | 0 | 172.22222222222223 | 0.09369111061096191 | 0 | 1 |

*Εικόνα 12 - Στιγμιότυπο από αποτέλεσμα query στην βάση μας με τα πακέτα ταξινομημένα βάσει όπου διακρίνεται η τυχαία κατανομή τους σε συνόδους*

### 3.3.4 Διαδικασία δημιουργίας τελικού Dataset

Έχοντας αναλύσει τις επιλογές που κάναμε, στην ενότητα αυτή θα προχωρήσουμε στην ανάλυση της διαδικασίας που ακολουθήθηκε.

#### *3.3.4.1 Μέγεθος σετ δεδομένων*

Όπως αναφέραμε, προχωρήσαμε στην χρήση δύο ξεχωριστών σετ δεδομένων εκπαίδευσης τα οποία δημιουργηθήκανε με την ίδια ακριβώς λογική. Εξαιτίας του υπολογιστικού κόστους που προκύπτει από απαραίτητες μετατροπές των δεδομένων πριν τροφοδοτήσουν το μοντέλο μας, δεν ήταν εφικτό να χρησιμοποιήσουμε πολύ μεγάλα αρχικά σετ δεδομένων οπότε καταλήξαμε σε ένα μέγεθος 100.000 πακέτων ανά σετ.

#### *3.3.4.2 Διαδικασία Δημιουργίας του τελικού σετ*

Η δημιουργία των τελικών μας σετ δεδομένων εκπαίδευσης αποτελείται από δύο διακριτές φάσεις, αρχικά την δημιουργία επί μέρους subsets και στην συνέχεια την ένωση αυτών σε δύο τελικά. Με τον τρόπο αυτό πετύχαμε ισόποση κατανομή της πληροφορίας στα τελικά σετ δεδομένων εκπαίδευσης. Ακολουθεί σχετική ανάλυση.

**Φάση 1η**

Έχοντας συνολικά πέντε διαφορετικά dataset φυσιολογικής κίνησης αποθηκευμένα στην βάση μας, το κάθε ένα για διαφορετικό σενάριο κίνησης, και άλλα πέντε κακόβουλης κίνησης, εξαγάγαμε τα πρώτα 10.000 πακέτα από το κάθε dataset για την δημιουργία του πρώτου σετ δεδομένων εκπαίδευσης, και επαναλάβαμε το ίδιο για το δεύτερο σετ με την λογική που περιγράψαμε στην ενότητα 3.3.4.1. Πιο συγκεκριμένα για κάθε ένα dataset εκτελέσαμε δύο queries ελαφρώς διαφορετικά.

Το πρώτο ουσιαστικά εξάγει τα πρώτα 10.000 πακέτα από κάθε dataset έχοντας ταξινομήσει τα πακέτα μας βάσει αναγνωριστικού συνόδου TCP και αναγνωριστικού πακέτου μέσα στην κάθε σύνοδο. Στο παρακάτω παράδειγμα αιτήματος για την δημιουργία του πρώτου υποσυνόλου κακόβουλης κίνησης, βλέπουμε στην πράξη αυτό που περιγράψαμε. Επιλέγονται αρχικά τα γνωρίσματα του μοντέλου μας με επιπλέον πεδίο την χρονοσήμανση του κάθε πακέτου. Ακολουθούν τα απαραίτητα INNER JOINS, ενώ στο τέλος χρησιμοποιούνται ως συνθήκες το αναγνωριστικό του dataset καθώς και η ύπαρξη κακόβουλης κίνησης.

```
SELECT p.packetsarrivaltime, p.packetsize, p.payloadsize, p.payloadratio, p.packetsratiotopreviouspacket, p.packettimedifferencetopreviouspacket, s.isMalicious INTO OUTFILE "/tmp/mal101.csv" FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY "" LINES TERMINATED BY "\n" FROM packets AS p INNER JOIN sessions s ON (p.idsessions = s.idsessions) INNER JOIN dataset d ON (s.iddataset=d.iddataset) WHERE d.iddataset = 12 AND s.isMalicious = 1 ORDER BY s.idsessions,p.packetindx LIMIT 10000;
```

Για το δεύτερο υποσύνολο, και τηρώντας την προϋπόθεση να ξεκινάει και εκείνο με την αρχή μιας συνόδου, αρχικά εντοπίσαμε την σύνοδο στην οποία ανήκε το τελευταίο πακέτο του πρώτου υποσυνόλου καθώς και τις συνόδους ανά dataset και στην συνέχεια με σχετικό αίτημα συμπεριλάβαμε τα πακέτα από την αρχή της επόμενης συνόδου και μέχρι την τελευταία με όρισμα LIMIT 10000. Με αυτό τον τρόπο εξασφαλίζαμε ότι θα ξεκινάει με την αρχή συνόδου ενώ όπως εξηγήσαμε και προηγουμένως, η τελευταία σύνοδος δεν είναι απαραίτητο να είναι ολόκληρη.

Στο παρακάτω αίτημα για την δημιουργίας του δεύτερου υποσυνόλου για το πρώτο dataset φυσιολογικής κίνησης, χρησιμοποιούνται τα BETWEEN…AND της MySQL με ορίσματα τα αναγνωριστικά της αμέσως επόμενης συνόδου από την τελευταία του πρώτου υποσυνόλου και της τελευταίας του dataset.

```
SELECT p.packetsarrivaltime, p.packetsize, p.payloadsize, p.payloadratio, p.packetsratiotopreviouspacket,
p.packettimedifferencetopreviouspacket, s.isMalicious INTO OUTFILE "/tmp/norm102.csv" FIELDS TERMINATED BY ',' OPTIONALLY
ENCLOSED BY '"' LINES TERMINATED BY "\n" FROM packets AS p INNER JOIN sessions s ON (p.idsessions = s.idsessions) INNER
JOIN dataset d ON (s.iddataset=d.iddataset) WHERE d.iddataset = 1 AND s.isMalicious = 0 AND s.idsessions BETWEEN 1020 AND 1290
ORDER BY s.idsessions,p.packetindx LIMIT 10000;
```

**Φάση 2η**

Έχοντας δημιουργήσει τα επιμέρους σετ δεδομένων, το τελευταίο βήμα είναι η σύνθεση του τελικού μας αρχείου csv, δηλαδή εκείνο που θα εισάγουμε στο python project ως σετ δεδομένων στο μοντέλο μας. Ενώ η διαδικασία που ακολουθήθηκε ήταν ένα απλό concatenation, μέσω της εντολής "cat" του Linux, θεωρήσαμε σημαντικό να εισάγουμε στο dataset μας όση περισσότερη τυχαιότητα ήταν εφικτό. Για αυτό τον λόγο, επιλέξαμε να εισάγουμε τα αρχεία εναλλάξ. Στο παρακάτω κώδικα φαίνεται το concatenation για την δημιουργία του πρώτου σετ εκπαίδευσης.

```
cat mal501.csv norm101.csv mal101.csv norm201.csv mal201.csv norm301.csv mal301.csv norm401.csv mal401.csv norm501.csv >
/home/eukokol/Desktop/dtset01.csv
```

# 4 Μεθοδολογία Υλοποίησης: Μοντέλο Μηχανικής Μάθησης

## 4.1 Εισαγωγή

Το παρόν κεφάλαιο αφορά την δεύτερη φάση της υλοποίησής μας και πιο συγκεκριμένα το μοντέλο M.M.. Με την θεωρία πάνω στην Μηχανική Μάθηση καθώς και στον αλγόριθμο Τυχαίου Δάσους να έχει καλυφθεί στο 2ο Κεφάλαιο, στο παρόν θα αναλύσουμε στοιχεία πάνω στον κώδικά μας καθώς και στις επιλογές που έγιναν.

Σε αυτό το σημείο και πριν ξεκινήσουμε, θεωρήσαμε καλό να κάνουμε μια αναφορά στην πρώτη προσπάθεια που έγινε, όπου το μοντέλο μας βασίστηκε στα LSTMs (Long-Short Term Memory) [50]. Πρόκειται για τύπο νευρωνικών δικτύων της κατηγορίας των Recursive Neural Networks. Είναι ιδανικά σε περιπτώσεις σειριακού σετ δεδομένων (sequence data) και αυτός ήταν ο λόγος που τα επιλέξαμε. Ωστόσο, επειδή πρόκειται για ένα αρκετά πολύπλοκο αντικείμενο τόσο ως προς την υλοποίησή τους όσο και ως προς την οργάνωση των δεδομένων εκπαίδευσης που τα τροφοδοτούν, στραφήκαμε στο μοντέλο M.M. Random Forest.

Η δεύτερη φάση της υλοποίησής μας, χωρίζεται σε δύο σκέλη. Στο πρώτο, πραγματοποιούμε μια διαδικασία ελέγχου της εγκυρότητας του μοντέλου μας (validation process) με σκοπό την παραμετροποίηση των παραμέτρων (hyperparameters) του αλγορίθμου μας, που στην προκειμένη είναι ο RandomForestRegressor. Στο δεύτερο σκέλος, έχοντας προσδιορίσει τις κατάλληλες παραμέτρους, προχωράμε σε μια καινούργια πρόβλεψη. Ακολουθούν οι αντίστοιχες ενότητες με την ανάλυσή μας.

## 4.2 Σχετική θεωρία

Όπως αναφέραμε, το μέρος που αφορά την υλοποίηση του μοντέλου μας αποτελείται από δύο διακριτές φάσεις η κάθε μία από τις οποίες αντιστοιχεί διαφορετικό python script. Ωστόσο οι ουσιαστικές διαφορές μεταξύ των δύο script είναι ελάχιστες. Πριν ωστόσο περάσουμε στην ανάλυση του κώδικα, θα εξηγήσουμε την μεθοδολογία στην οποία βασιστήκαμε, αρχικά αναφορικά με την οργάνωση των δεδομένων μας και στην συνέχεια για την διαδικασία ελέγχου εγκυρότητας.

### 4.2.1 Μορφοποίηση Sequential Data

Με σκοπό να χρησιμοποιήσουμε αυτήν την κατηγορία δεδομένων για την τροφοδότηση ενός μοντέλου επιτηρούμενης μάθησης, θα πρέπει να προηγηθεί η οργάνωσή τους σε κατάλληλη μορφή. Η αναγκαία αυτή τροποποίηση είναι ο μόνος τρόπος ώστε τα δεδομένα μας να μπορούν να τροφοδοτούν μοντέλα επιτηρούμενης μάθησης τα οποία αναμένουν τιμές εισόδου (X) και εξόδου (y).

Σε μια κλασική περίπτωση δεδομένων σε σειρά, είτε πρόκειται για χρονική είτε όχι, οι παρατηρήσεις είναι η μία κάτω από την άλλη με βάση μια αυστηρά συγκεκριμένη ταξινόμηση. Για παράδειγμα, στην Εικόνα 13, φαίνεται η μορφή των δεδομένων μας σε σειρά σε μορφή μη κατάλληλη για την τροφοδοσία του μοντέλου μας.

| Arrival_Time | Packet_Size | Payload_Size | Payload_Ratio | Ratio_to_Previous | Time_Difference | Flag |
|---|---|---|---|---|---|---|
| 2011-08-16 05:52:39.471525 | 62 | 0 | 0.0 | 1.000000 | 0.000000 | 1 |
| 2011-08-16 05:52:39.471536 | 62 | 0 | 0.0 | 100.000000 | 0.000011 | 1 |
| 2011-08-16 05:52:40.032196 | 62 | 0 | 0.0 | 100.000000 | 0.560671 | 1 |
| 2011-08-16 05:52:40.032207 | 62 | 0 | 0.0 | 100.000000 | 0.560682 | 1 |
| 2011-08-16 05:52:40.633329 | 62 | 0 | 0.0 | 100.000000 | 1.161804 | 1 |
| 2011-08-16 05:52:40.633339 | 62 | 0 | 0.0 | 100.000000 | 1.161814 | 1 |
| 2011-08-15 23:41:17.760124 | 60 | 0 | 0.0 | 1.000000 | 0.000000 | 1 |
| 2011-08-15 23:41:18.319177 | 60 | 0 | 0.0 | 100.000000 | 0.559053 | 1 |
| 2011-08-15 23:41:18.820336 | 60 | 0 | 0.0 | 100.000000 | 1.060212 | 1 |
| 2011-08-16 04:09:17.150959 | 60 | 0 | 0.0 | 1.000000 | 0.000000 | 1 |
| 2011-08-16 04:09:17.712154 | 60 | 0 | 0.0 | 100.000000 | 0.561195 | 1 |
| 2011-08-16 04:09:18.312542 | 60 | 0 | 0.0 | 100.000000 | 1.161583 | 1 |
| 2011-08-16 04:09:14.847904 | 60 | 0 | 0.0 | 1.000000 | 0.000000 | 1 |
| 2011-08-16 04:09:15.408991 | 60 | 0 | 0.0 | 100.000000 | 0.561087 | 1 |
| 2011-08-16 04:09:16.009499 | 60 | 0 | 0.0 | 100.000000 | 1.161595 | 1 |
| 2011-08-16 04:09:21.757995 | 60 | 0 | 0.0 | 1.000000 | 0.000000 | 1 |
| 2011-08-16 04:09:22.318556 | 60 | 0 | 0.0 | 100.000000 | 0.560561 | 1 |
| 2011-08-16 04:09:22.919566 | 60 | 0 | 0.0 | 100.000000 | 1.161571 | 1 |
| 2011-08-16 04:09:19.454846 | 60 | 0 | 0.0 | 1.000000 | 0.000000 | 1 |
| 2011-08-16 04:09:20.015400 | 60 | 0 | 0.0 | 100.000000 | 0.560554 | 1 |
| 2011-08-16 04:09:20.616316 | 60 | 0 | 0.0 | 100.000000 | 1.161470 | 1 |
| 2011-08-16 00:20:57.072843 | 62 | 0 | 0.0 | 1.000000 | 0.000000 | 1 |
| 2011-08-16 00:20:57.072853 | 62 | 0 | 0.0 | 100.000000 | 0.000010 | 1 |
| 2011-08-16 00:20:57.633347 | 62 | 0 | 0.0 | 100.000000 | 0.560504 | 1 |

*Εικόνα 13 - Μορφή των δεδομένων του dataset μας πριν την μετατροπή τους σε μορφή κατάλληλη για να τροφοδοτήσει το μοντέλο μας*

#### *4.2.1.1 Sliding Window*

Για τον λόγο αυτόν χρησιμοποιήσαμε την τεχνική του Slide Window[51]. Πρόκειται για μία τεχνική, όπου στην θέση του X μπαίνει η τιμή του αμέσως προηγούμενου βήματος ενώ για y έχουμε την τιμή του επόμενου. Με αυτό τον τρόπο ένα dataset σειριακών δεδομένων μπορεί να μετατραπεί κατάλληλα ώστε να μπορέσει το μοντέλο μας να εκπαιδευτεί. Στην Εικόνα 14 φαίνεται το παραπάνω dataset μορφοποιημένο.

| | var1(t-1) | var2(t-1) | var3(t-1) | var4(t-1) | var5(t-1) | var6(t-1) | var1(t) | var2(t) | var3(t) | var4(t) | var5(t) | var6(t) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 62.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 | 62.0 | 0.0 | 0.0 | 100.0 | 0.000011 | 1.0 |
| 2 | 62.0 | 0.0 | 0.0 | 100.0 | 0.000011 | 1.0 | 62.0 | 0.0 | 0.0 | 100.0 | 0.560671 | 1.0 |
| 3 | 62.0 | 0.0 | 0.0 | 100.0 | 0.560671 | 1.0 | 62.0 | 0.0 | 0.0 | 100.0 | 0.560682 | 1.0 |
| 4 | 62.0 | 0.0 | 0.0 | 100.0 | 0.560682 | 1.0 | 62.0 | 0.0 | 0.0 | 100.0 | 1.161804 | 1.0 |
| 5 | 62.0 | 0.0 | 0.0 | 100.0 | 1.161804 | 1.0 | 62.0 | 0.0 | 0.0 | 100.0 | 1.161814 | 1.0 |
| 6 | 62.0 | 0.0 | 0.0 | 100.0 | 1.161814 | 1.0 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 |
| 7 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 0.559053 | 1.0 |
| 8 | 60.0 | 0.0 | 0.0 | 100.0 | 0.559053 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 1.060212 | 1.0 |
| 9 | 60.0 | 0.0 | 0.0 | 100.0 | 1.060212 | 1.0 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 |
| 10 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 0.561195 | 1.0 |
| 11 | 60.0 | 0.0 | 0.0 | 100.0 | 0.561195 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 1.161583 | 1.0 |
| 12 | 60.0 | 0.0 | 0.0 | 100.0 | 1.161583 | 1.0 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 |
| 13 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 0.561087 | 1.0 |
| 14 | 60.0 | 0.0 | 0.0 | 100.0 | 0.561087 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 1.161595 | 1.0 |
| 15 | 60.0 | 0.0 | 0.0 | 100.0 | 1.161595 | 1.0 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 |
| 16 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 0.560561 | 1.0 |
| 17 | 60.0 | 0.0 | 0.0 | 100.0 | 0.560561 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 1.161571 | 1.0 |
| 18 | 60.0 | 0.0 | 0.0 | 100.0 | 1.161571 | 1.0 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 |
| 19 | 60.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 0.560554 | 1.0 |
| 20 | 60.0 | 0.0 | 0.0 | 100.0 | 0.560554 | 1.0 | 60.0 | 0.0 | 0.0 | 100.0 | 1.161470 | 1.0 |

*Εικόνα 14 - Το μορφοποιημένο dataset μας*

Το παραπάνω dataset προέκυψε μέσω της ίδιας συνάρτησης που χρησιμοποιήσαμε και στον κώδικά μας με την διαφορά ότι χρησιμοποιεί μόνο ένα χρονικό βήμα. Χαρακτηριστικά, οι στήλες var1(t-1) μέχρι και την var6(t-1), περιέχουν τις τιμές των γνωρισμάτων την αμέσως προηγούμενης παρατήρησης ενώ οι εναπομείναντες περιέχουν τα γνωρίσματα της επόμενης παρατήρησης. Με αυτό τον τρόπο μπορούμε για το test_X να χρησιμοποιήσουμε τις στήλες της προηγούμενης παρατήρησης και για test_y τις στήλες της επόμενης.

#### *4.2.1.2 series_to_supervised()*

Στην πράξη, βασιστήκαμε στον κώδικα του [52] ο οποίος στον πυρήνα του κάνει χρήση της έτοιμης συνάρτησης που παρέχεται από την βιβλιοθήκη pandas, την shift(), η οποία πετυχαίνει αυτό που εξηγήσαμε στην προηγούμενη ενότητα. Παρακάτω παραθέτουμε τον σχετικό κώδικα ο οποίος χρησιμοποιήθηκε και στις δύο φάσεις υλοποίησης του μοντέλου μας.

```
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols, names = list(), list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j + 1, i)) for j in range(n_vars)]
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j + 1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j + 1, i)) for j in range(n_vars)]
    # put it all together
    agg = concat(cols, axis=1)
    agg.columns = names
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg.values
```

- Παράμετροι
  - **data**: πρόκειται για το σετ δεδομένων προς μορφοποίηση
  - **n_in**: αφορά τον αριθμό των προηγούμενων παρατηρήσεων βάσει των οποίων θα γίνει η πρόβλεψη από τον αλγόριθμό μας. Ουσιαστικά το X. Στην δική μας περίπτωση βασίσαμε τις προβλέψεις μας σε 6 προηγούμενες παρατηρήσεις.
  - **n_out**: αφορά τις παρατηρήσεις που θα έχουν το ρόλο του y. Εδώ επιλέξαμε αρχικά να γίνεται πρόβλεψη της επόμενης παρατήρησης οπότε η παράμετρος ισούταν με 1. Ωστόσο προχωρήσαμε και σε προβλέψεις των 6 επόμενων παρατηρήσεων.
  - **dropnan**: Εξαιτίας του τρόπου που δουλεύει ο αλγόριθμος σε κάποιες στήλες δημιουργούνται κενά καθώς γίνεται μετατόπιση των παρατηρήσεων. Αυτές οι γραμμές αφαιρούνται αυτόματα με την τιμή ίση με *True*.

- Περιγραφή διαδικασίας
  Αρχικά το σετ δεδομένων μετατρέπεται σε δομή τύπου dataframe, όπου τα δεδομένα οργανώνονται σε αριοθμημένες σειρές και στήλες. Στην συνέχεια ακολουθούν δύο επαναλήψεις οι οποίες αναλαμβάνουν να μορφοποιήσουν τα δεδομένα μας. Η πρώτη αναλαμβάνει τα δεδομένα εισόδου, τα X μας, καθώς και τα ονόματα των στηλών, ενώ η δεύτερη τα δεδομένα εξόδου, τα y, με τις αντίστοιχες στήλες. Στο τέλος συνδυάζονται τα αποτελέσματα και επιστέφεται το περιεχόμενο του νέου σετ, *return agg.values* .

### 4.2.2 Walk Forward Validation

Τα δεδομένα του dataset μας αν και δεν θεωρούνται timeseries, καθώς την ταξινόμησή τους δεν την βασίσαμε στην χρονοσήμανση του κάθε πακέτου, αλλά περισσότερο σειριακά δεδομένα (sequential data), χαρακτηρίζονται από μία συνθήκη. Ο τρόπος που είναι οργανωμένα είναι καθοριστικός για το χώρο του προβλήματος. Η συνθήκη αυτή, ωστόσο, δημιουργεί ένα πρόβλημα με μια παραδοσιακή μεθοδολογία για τον έλεγχο της απόδοσης ενός μοντέλου, την k-fold Cross Validation [53].

Πρόκειται για μια στατιστική μέθοδο για τον προσδιορισμό της ικανότητας ενός μοντέλου M.M. Το ‘k’ στο όνομα αναφέρεται σε πόσα μέρη ένα δείγμα δεδομένων θα χωριστεί. Ο σκοπός είναι να ελεγχθεί η ικανότητα πρόβλεψης του μοντέλου σε εντελώς νέα δεδομένα (unseen data) αφού έχει εκπαιδευτεί ήδη σε γνωστά. Ο αλγόριθμος ξεκινάει ανακατεύοντας το δείγμα τυχαία. Στην συνέχεια γίνεται ο διαχωρισμός σε k τμήματα από τα οποία επιλέγεται το πρώτο ως σετ αξιολόγησης και τα υπόλοιπα τμήματα, k-1, συνιστούν το σετ εκπαίδευσης του μοντέλου. Όπως είναι φανερό από την παραπάνω διαδικασία, από το πρώτο βήμα, ένα σειριακά οργανωμένο σετ δεδομένων, χάνει την ιδιότητα του καθώς ο αλγόριθμος εκλαμβάνει την κάθε παρατήρηση ανεξάρτητη από τις υπόλοιπες.

Με σκοπό, λοιπόν, να μην αλλοιωθεί η συνθήκη που διέπει το σετ δεδομένων μας, επιλέξαμε την χρήση της τεχνικής *Walk Forward Validation* για την φάση της αξιολόγησης του μοντέλου. Η συγκεκριμένη μέθοδος χρησιμοποιεί επίσης την τεχνική του κυλιόμενου παραθύρου, όπως ακριβώς και η συνάρτηση *series_to_supervised*[54].

1. Επιλεγεται ένα αρχικό σετ δεδομένων εκπαίδευσης (training set) και ένα σετ δεδομένων ελέγχου (test ή validation set).
2. Το μοντέλο εκπαιδεύεται με το σετ εκπαίδευσης.
3. Το μοντέλο προχωράει σε πρόβλεψη της επόμενης γνωστής παρατήρησης.
4. Η γνωστή παρατήρηση προστίθεται στο αρχικό σετ εκπαίδευσης, το παράθυρο δηλαδή διευρύνεται, και επαναλαμβάνεται το βήμα 2.

Η παραπάνω διαδικασία συνεχίζεται έως ότου το σετ δεδομένων ελέγχου (validation set) εξαντληθεί.

#### *4.2.2.1 Ανάλυση κώδικα python*

Για τον κώδικά μας βασιστήκαμε στο [55], ωστόσο χρειάστηκε να τον προσαρμόσουμε καθώς στην περίπτωσή μας είχαμε πρόβλεψη της τιμής πολλαπλών γνωρισμάτων (multivariate regression). Παρακάτω παραθέτουμε τον σχετικό κώδικα ενώ στην συνέχεια ακολουθεί η ανάλυσή του καθώς και των επί μέρους συναρτήσεων που καλούνται μέσα.

```
1  def walk_forward_validation(data, val_size, lag_obs):
2      predictions = list()
3      n_pred = lag_obs * 6
4      # split dataset
5      train, test = train_test_split(data, val_size)
6      # seed history with training dataset
7      history = [x for x in train]
8      # step over each time-step in the test set
9      index = 0
10     for i in range(len(test)):
11         index += 1
12         # (!!) split test row into input and output columns
13         testX, testy = test[i, :n_pred], test[i, n_pred:]
14         # fit model on history and make a prediction
15         yhat = random_forest_forecast(history, testX, n_pred)
16         # store forecast in list of predictions
17         predictions.append(yhat)
18         # add actual observation to history for the next loop
19         history.append(test[i])
20         # summarize progress
21         print(f'{index})>expected={testy}, predicted={yhat}')
22     # estimate prediction error
23     error = mean_absolute_error(test[:, n_pred:], predictions, multioutput='uniform_average')
24     return error
```

- <u>Παράμετροι</u>
  - data: Αφορά το σετ δεδομένων μας. Όπως θα δούμε στην συνέχεια, τα δεδομένα μας τα περνάμε ως παράμετρο αφού υποστούν κανονικοποίηση (normalization).
  - val_size: Σε αυτή την παράμετρο ορίζουμε το μέγεθος του δείγματος που θα χρησιμοποιήσουμε για τον έλεγχο της απόδοσης του αλγορίθμου μας. Όπως θα δούμε χρησιμοποιείται στη κληθείσα συνάρτηση *train_test_split()*.
  - lag_obs: Η παράμετρος στην οποία αποθηκεύουμε τον αριθμό των προηγούμενων παρατηρήσεων στις οποίες θα βασιστούν οι προβλέψεις μας.

- <u>Περιγραφή διαδικασίας</u>

  Υπολογίζουμε την μεταβλητή n_pred που αποτελεί ουσιαστικά το όριο για τον καθορισμό των X και y. Καλούμε στην συνέχεια την συνάρτηση *train_test_split()* με παραμέτρους το σετ δεδομένων μας και την val_size. Επιστρέφονται τα σετ εκπαίδευσης και ελέγχου και αποθηκεύονται στις *train* και *test* αντίστοιχα. Αποθηκεύουμε στην μεταβλητή *history* το σύνολο του σετ εκπαίδευσης.

  Ακολουθεί η βασική επανάληψη η οποία, διαχωρίζει αρχικά σε X και y το σετ ελέγχου κάνοντας χρήση του ορίου που περιγράψαμε πιο πάνω, αποθηκεύοντας τα αποτελέσματα στις μεταβλητές *testX* και

*testy*. Γίνεται κλήση της συνάρτησης *random_forest_regressor()* με παραμέτρους το history, που περιέχει το train dataset, το testX από το όριο από προηγούμενα βήματα. Το απότέλεσμα επιστρέφεται στην μεταβλητή yhat η οποία σε κάθε επανάληψη γεμίζει την λίστα *predictions*.

Στη συνέχεια βλέπουμε την εντολή *history.append(test[i])*. Αξίζει να σταθούμε στην συγκεκριμένη γραμμή καθώς έτσι επιτυγχάνεται η τεχνική sliding window που περιγράψαμε στην αρχή της ενότητας. Ουσιαστικά μετά από την αρχική τροφοδότηση του μοντέλου μας, πρώτη επανάληψη, σε κάθε επόμενη, στο *history* προστίθεται κάθε φορά η επόμενη παρατήρηση από το *test dataset*.

Στο τέλος κάθε επανάληψης εκτυπώνεται η αναμενόμενη παρατήρηση και εκείνη που προβλέφθηκε ενώ μετά το πέρας των επαναλήψεων υπολογίζεται το Mean Absolute Error[56], μια μετρική υπολογισμού της απόκλισης των προβλέψεών που παρέχεται από την scikit-learn. Η συνάρτηση πριν έξοδο της επιστρέφει την απόκλιση το testy και την λίστα με τις προβλέψεις.

- train_test_split()

```
1 def train_test_split(data, val_size):
2     return data[:-val_size], data[-val_size:]
```

Χωρίζει το dataset σε train και test με την τεχνική list slicing.

- random_forest_forecast

```
1  def random_forest_forecast(train, testX, n_obs):
2      # transform list into array
3      train = asarray(train)
4      # split into input and output columns
5      trainX, trainy = train[:, :n_obs], train[:, n_obs:]
6      # fit model
7      model = RandomForestRegressor(n_estimators=1000, n_jobs=8)
8      model.fit(trainX, trainy)
9      # make a one-step prediction
10     yhat = model.predict([testX])
11     return yhat[0]
```

Είναι η συνάρτηση όπου καλούμε τον αλγόριθμο RandomForestRegressor(). Μετά τις παραμέτρους που ορίζονται στην *walk_forward_validation()*, μετατρέπουμε το training dataset σε πίνακα και τον χωρίζουμε στα *trainX* και *trainy*. Στη συνέχεια εκπαιδεύουμε το μοντέλο μας με την χρήση χιλίων δέντρων απόφασης, ορίζοντας σαν υπερπαράμετρο *n_estimators=1000*, και χρήση οχτώ παράλληλων νημάτων. Στο τέλος κάνουμε πρόβλεψη κάνοντας χρήση του testX. Η πρόβλεψη επιστρέφεται στο κυρίως πρόγραμμα.

- Υπόλοιπος κώδικας

```
1  # load dataset
```

```
filename = 'dataset_12K.csv'
names = ['Arrival_Time', 'Packet_Size', 'Payload_Size', 'Payload_Ratio',
        'Ratio_to_Previous', 'Time_Difference', 'Flag']
dt = read_csv(filename, parse_dates=True, index_col=0, names=names)

values = dt.values

# Scale the data before fitting
values = values.astype('float32')
scaler = StandardScaler()
scaled = scaler.fit_transform(values)

lag_obs = 6
pred = 1
val_size = 50

data = series_to_supervised(scaled, lag_obs, pred)


mae, y, yhat = walk_forward_validation(data, val_size, lag_obs)
print('MAE: %.3f' % mae)
```

Το κυρίως πρόγραμμα, στο οποίο καλούνται οι *series_to_supervised()* και *walk_forward_validation()*. Ξεκινάει διαβάζοντας το βασικό μας σετ δεδομένων ορίζοντας ονόματα στις στήλες με το "Arrival Time" να εξαιρείται και να μπαίνει απλά σαν δείκτης.

Στη συνέχεια, ακολουθείται η διαδικασία κανονικοποίησης. Η κανονικοποίηση είναι μια τεχνική επί των δεδομένων εισόδου καθώς τιμές γνωρισμάτων που παρουσιάζουν ισχυρές διακυμάνσεις μεταξύ τους, όπως για παράδειγμα στον αριθμό των δεκαδικών τους ψηφίων. Η scikit-learn παρέχει πολλές και διαφορετικής νοοτροπίας μεθόδους κανονικοποίησης (scalers). Ωστόσο δεν κάνουν όλες για όλα τα είδη δεδομένων. Στην αρχή χρησιμοποιήσαμε τον MinMaxScaler(), αλλά παρατηρήσαμε ότι υπήρχε απώλεια πληροφορίας καθώς κατά την μετατροπή στην αρχική μορφή υπήρχε αλλοίωση. Οπότε κάναμε χρήση του StandarScaler() [57]. Επομένως ως παράμετρος δεδομένων στις συναρτήσεις που περιγράψαμε προηγουμένως είναι το αρχικό μας σετ μετά την κανονικοποίηση.

Στη συνέχεια ορίζονται οι τιμές για τις μεταβλητές *lag_obs*, *pred*, που είναι ο αριθμός των παρατηρήσεων που θέλουμε να προβλέψουμε, και *val_size*.

### 4.2.3 Πρόβλεψη (Prediction)

Έχοντας στην προηγούμενη φάση καταλήξει στις υπερπαραμέτρους του μοντέλου μας, στο σκέλος αυτό προχωράμε στην πρόβλεψη νέων παρατηρήσεων. Σε αντίθεση με το σκέλος του Walk Forward Validation, εδώ θα τροφοδοτήσουμε το μοντέλο μας με το ολόκληρο το σετ δεδομένων. Η μορφοποίηση των δεδομένων, ωστόσο, σε δεδομένα κατάλληλα για επιτηρούμενη μάθηση θα γίνει και εδώ με την ίδια ακριβώς συνάρτηση, *series_to_supervised()*, που χρησιμοποιήσαμε και στο πρώτο σκέλος.

- Υπόλοιπος κώδικας

```
filename = 'dtset01.csv'
names = ['Arrival_Time', 'Packet_Size', 'Payload_Size', 'Payload_Ratio',
        'Ratio_to_Previous', 'Time_Difference', 'Flag']
dt = read_csv(filename, parse_dates=True, index_col=0, names=names)

values = dt.values

values = values.astype('float32')
scaler = StandardScaler()
scaled = scaler.fit_transform(values)

lag_obs = 6

# reframe data to fit the ML model
data = series_to_supervised(scaled, lag_obs)

# splitting the data into input and output
n_pred = lag_obs * 6
train_X, train_y = data[:, :n_pred], data[:, n_pred:]

# fit model
model = RandomForestRegressor(n_estimators=1000)
model.fit(train_X, train_y)

row = values[-6:].flatten()
# print(asarray([row]))

yhat = model.predict(asarray([row]))
inv_yhat = scaler.inverse_transform(yhat)

print(f'Input: {row})')

print('\n########################################')
print('\n########## Predicted Packet ##########')
print(inv_yhat)
```

Όπως φαίνεται πέρα από την κανονικοποίηση και την κλήση της συνάρτησης *series_to_supervised()*, η διαφορά όπως αναφέραμε είναι η εκπαίδευση του μοντέλου μας με όλο το σετ δεδομένων. Άξια αναφοράς είναι η μέθοδος inverse_transform() η οποία μετατρέπει να κανονικοποιημένα δεδομένα μας στην αρχική τους μορφή, όπου στην προκειμένη περίπτωση αφορά την πρόβλεψή μας.

# 5 Προβλήματα με την μεθοδολογία μας & Αλλαγές

## 5.1 Εισαγωγή

Έχοντας αναλύσει τον κώδικα των δύο μεγάλων script, για το validation και το prediction, καθώς και την αρχική μας μεθοδολογία, στο παρόν κεφάλαιο θα αναλύσουμε τα προβλήματα που παρουσιάστηκαν καθώς και τις λύσεις στις οποίες προχωρήσαμε. Ενώ η γενική θεωρία μας παραμένει η ίδια, χρειάστηκε να προβούμε σε τροποποιήσεις σχετικές τόσο με την διαδικασία επιλογής και δημιουργίας του τελικού σετ δεδομένων όσο και με τον κώδικα του Walk Forward Validation.

## 5.2 Το Πρόβλημα των TCP Acknowledgement (ACK) πακέτων

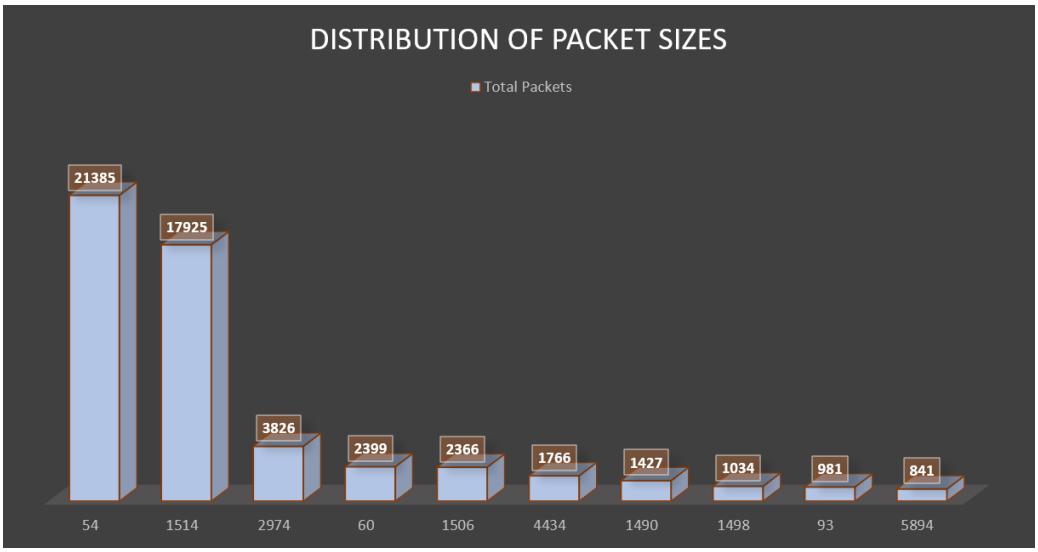
Στα προηγούμενα κεφάλαια, κάναμε αναφορά στα χαρακτηριστικά των πακέτων που επιλέξαμε ως γνωρίσματα, Ενότητα 3.3.1, δύο εκ των οποίων ήταν τα packetsize και payloadsize, ενώ στην Ενότητα 2.3.2 κάναμε μια σύντομη αναφορά στα μηνύματα διαχείρισης των συνόδων TCP. Τέτοια μηνύματα στέλνονται μέσα σε πακέτα τα οποία όμως είναι σταθερού μεγέθους και χωρίς ωφέλιμο φορτίο καθώς περιέχουν απλές τιμές που προσδιορίζουν διαφορετικά σήματα.

Παρατηρήθηκε ένα σταθερό μοτίβο τέτοιων πακέτων, με πλειοψηφία στην μεν φυσιολογική κίνηση με μέγεθος 54 και 60 bytes ενώ στην κακόβουλη με 60 και 62 bytes αντίστοιχα. Με την βοήθεια του Wireshark είναι εύκολο να εντοπίσουμε ότι τα παραπάνω μεγέθη αντιστοιχούν σε συνήθη μεγέθη ACK σημάτων. Η παρακάτω εικόνα δείχνει ένα στιγμιότυπο από το 2ο dataset όπου έχουμε επιλέξει ένα τυχαίο πακέτο σηματοδοσίας.

```
Frame 57: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Dec 21, 2021 16:52:53.410025000 GTB Standard Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1640098373.410025000 seconds
    [Time delta from previous captured frame: 0.000076000 seconds]
    [Time delta from previous displayed frame: 0.000076000 seconds]
    [Time since reference or first frame: 1.530962000 seconds]
    Frame Number: 57
    Frame Length: 54 bytes (432 bits)
    Capture Length: 54 bytes (432 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp]
    [Coloring Rule Name: TCP]
    [Coloring Rule String: tcp]
> Ethernet II, Src: D-LinkIn_01:0c:57 (18:0f:76:01:0c:57), Dst: Cisco_43:67:77 (00:3c:10:43:67:77)
> Internet Protocol Version 4, Src: 106.117.1.74, Dst: 165.225.81.247
Transmission Control Protocol, Src Port: 61392, Dst Port: 8080, Seq: 1, Ack: 1, Len: 0
    Source Port: 61392
    Destination Port: 8080
    [Stream index: 8]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 3719026916
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 2324227279
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 512
    [Calculated window size: 131072]
    [Window size scaling factor: 256]
    Checksum: 0x14f3 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
```

*Εικόνα 15 - Στιγμιότυπο από το Wireshark, όπου διακρίνονται χαρακτηριστικά ενός πακέτου σηματοδοσίας ACK του TCP. Μαρκαρισμένα το μέγεθος του dataframe που περιλαμβάνει την σχετική πληροφορία, και η σχετική σημαία για το ACK.*

Στην Εικόνα 16, παρατηρούμε πως τα πακέτα με μέγεθος 54 bytes, είναι αναλογικά πολύ περισσότερα, ενώ στην Εικόνα 17 φαίνεται ένα στιγμιότυπο μια TCP συνόδου, όπου αναδεικνύεται το πρόβλημα της επαναληψιμότητας τέτοιων πακέτων.




*Εικόνα 16 – Τα πρώτα δέκα μεγέθη πακέτων σετ δεδομένων φυσιολογικής κίνησης, ταξινομημένα βάσει συχνότητας εμφάνισης.*

| packetsarrivaltime | idpackets | idsessions | packetindx | packetsize | payloadsize | payloadratio | packetsratiotopreviouspacket | packettimedifferencetopreviouspacket | isMalicious | isEncrypted |
|---|---|---|---|---|---|---|---|---|---|---|
| 2021-12-21 16:54:57.277030 | 1,113,861 | 1,736 | 172 | 54 | 0 | 0 | 100 | 19.6203510761261 | 0 | 0 |
| 2021-12-21 16:54:57.278745 | 1,113,862 | 1,736 | 173 | 54 | 0 | 0 | 100 | 19.622066020965576 | 0 | 0 |
| 2021-12-21 16:54:57.331707 | 1,113,863 | 1,736 | 174 | 54 | 0 | 0 | 100 | 19.675028085708618 | 0 | 0 |
| 2021-12-21 16:54:57.333225 | 1,113,864 | 1,736 | 175 | 54 | 0 | 0 | 100 | 19.676546096801758 | 0 | 0 |
| 2021-12-21 16:54:57.333480 | 1,113,865 | 1,736 | 176 | 54 | 0 | 0 | 100 | 19.676800966262817 | 0 | 0 |
| 2021-12-21 16:54:57.333963 | 1,113,866 | 1,736 | 177 | 54 | 0 | 0 | 100 | 19.677284002304077 | 0 | 0 |
| 2021-12-21 16:54:57.334181 | 1,113,867 | 1,736 | 178 | 54 | 0 | 0 | 100 | 19.677502155303955 | 0 | 0 |
| 2021-12-21 16:54:57.334600 | 1,113,868 | 1,736 | 179 | 54 | 0 | 0 | 100 | 19.677921056747437 | 0 | 0 |
| 2021-12-21 16:54:57.334953 | 1,113,869 | 1,736 | 180 | 54 | 0 | 0 | 100 | 19.678274154663086 | 0 | 0 |
| 2021-12-21 16:54:57.335332 | 1,113,870 | 1,736 | 181 | 54 | 0 | 0 | 100 | 19.67865300178528 | 0 | 0 |
| 2021-12-21 16:54:57.335728 | 1,113,871 | 1,736 | 182 | 54 | 0 | 0 | 100 | 19.679049015045166 | 0 | 0 |
| 2021-12-21 16:54:57.336159 | 1,113,872 | 1,736 | 183 | 54 | 0 | 0 | 100 | 19.67948007583618 | 0 | 0 |
| 2021-12-21 16:54:57.336458 | 1,113,873 | 1,736 | 184 | 54 | 0 | 0 | 100 | 19.679779052734375 | 0 | 0 |
| 2021-12-21 16:54:57.336898 | 1,113,874 | 1,736 | 185 | 54 | 0 | 0 | 100 | 19.680219173431396 | 0 | 0 |
| 2021-12-21 16:54:57.337294 | 1,113,875 | 1,736 | 186 | 54 | 0 | 0 | 100 | 19.680615186691284 | 0 | 0 |
| 2021-12-21 16:54:57.337552 | 1,113,876 | 1,736 | 187 | 54 | 0 | 0 | 100 | 19.680873155593872 | 0 | 0 |
| 2021-12-21 16:54:57.337987 | 1,113,877 | 1,736 | 188 | 54 | 0 | 0 | 100 | 19.681308031082153 | 0 | 0 |
| 2021-12-21 16:54:57.338425 | 1,113,878 | 1,736 | 189 | 54 | 0 | 0 | 100 | 19.681746006011963 | 0 | 0 |
| 2021-12-21 16:54:57.338700 | 1,113,879 | 1,736 | 190 | 54 | 0 | 0 | 100 | 19.682021141052246 | 0 | 0 |

*Εικόνα 17 - Στιγμιότυπο από σύνοδο TCP όπου φαίνεται επαναλαμβανόμενη πληροφορία.*

Η αρχική μας στρατηγική, όπως την περιγράψαμε στο προηγούμενο κεφάλαιο, για την επιλογή πακέτων από τα επί μέρους σετ δεδομένων, δεν είχε πρόβλεψη για τις προαναφερθείσες περιπτώσεις με αποτέλεσμα μεγάλο ποσοστό της πληροφορίας να είναι ίδια. Η επανάληψη αυτή, στα πλαίσια της διαδικασίας εκπαίδευσης, δεν προσφέρει καμία γνώση στο μοντέλο μας, καθώς πρόκειται για όμοιες παρατηρήσεις, αλλά επιπλέον δεν του επιτρέπει να γενικεύσει στις προβλέψεις του, καθώς οδηγεί στο φαινόμενο του overfitting[58].

Μια λύση, θα ήταν να εξαιρούσαμε όλα τα πακέτα σηματοδοσίας. Έτσι θα συνθέταμε ένα σετ δεδομένων με χρήσιμη πληροφορία για την εκπαίδευση του μοντέλου μας. Το πρόβλημα ωστόσο εντοπίζεται στα ίδια τα γνωρίσματα μας και πιο συγκεκριμένα στα *packetratiotopreviouspacket* και *packettimedifferencetopreviouspacket*, τα οποία εξαρτώνται από το αμέσως προηγούμενο πακέτο και είναι και ό λόγος που διατηρήσαμε την δομή της συνόδου από την αρχή. Μη τηρώντας την δομή αυτή κατά την εκπαίδευση, το μοντέλο ναι μεν «θα μάθει» αλλά δεν θα καταλήγει στα σωστά συμπεράσματα καθώς δεν θα έχει εκπαιδευτεί γύρω από το concept της TCP συνόδου και τις σειριακής κίνησης.

## 5.3 Καθαρισμός της πληροφορίας (Data Cleaning)

Η λύση στην οποία καταλήξαμε ήταν ο καθαρισμός της αρχικής ακατέργαστης πληροφορίας με σκοπό τον περιορισμό αρχικά του ποσοστού των πακέτων που αφορούσαν αποκλειστικά σε TCP σηματοδοσία και στην συνέχεια πακέτων που είχαν την τάση να επαναλαμβάνονται. Από την στιγμή που η επανάληψη αυτή δεν αποτελούσε κάποιο ποιοτικό γνώρισμα που θα είχε σημασία για την εκπαίδευση του μοντέλου, η αλλοίωσή της δεν θα επηρέαζε αρνητικά.

### 5.3.1 Εξακρίβωση του μεγέθους των «προβληματικών» πακέτων

Για κάθε dataset, είτε φυσιολογικής είτε κακόβουλης κίνησης, με σχετικό ερώτημα στην βάση εξαγάγαμε το σύνολο των πακέτων, οργανωμένα ανά μέγεθος πακέτου και ταξινομημένα με φθίνουσα σειρά συχνότητας εμφάνισης του εκάστοτε μεγέθους. Παρακάτω παραθέτουμε παράδειγμα του σχετικού ερωτήματος καθώς και ένα στιγμιότυπο του αποτελέσματος.

```
SELECT p.packetsize, COUNT(p.packetsize) AS frequency
FROM packets AS p INNER JOIN sessions s ON (p.idsessions = s.idsessions) INNER JOIN dataset d ON (s.iddataset=d.iddataset)
WHERE d.iddataset = 1 GROUP BY p.packetsize ORDER BY frequency desc;
```

Με αυτό τον τρόπο, διαπιστώναμε, αρχικά αν στο εν λόγω dataset υπήρχε πρόβλημα επαναλαμβανόμενης πληροφορίας, κάτι που εντοπίσαμε ωστόσο σε όλα τα dataset, σε τι βαθμό καθώς και με ποιο μέγεθος πακέτου. Πέρα από τα TCP διαχειριστικά πακέτα υπήρξε επανάληψη και σε κάποια άλλα μεγέθη πακέτων με ακριβώς ίδιες τιμές packetsize και payloadsize.

### 5.3.2 Απόρριψη συνόδων & περεταίρω βελτιστοποίηση

Όπως αναφέραμε και στην αρχή του κεφαλαίου, συναντήσαμε συνόδους αποτελούμενες εξολοκλήρου από σήματα σηματοδοσίας TCP. Γνωρίζοντας τα συνηθέστερα μεγέθη αυτών των πακέτων ήταν σχετικά εύκολο να τις εξαιρέσουμε απλά επιλέγοντας τις συνόδους που περιείχαν τα υπόλοιπα μεγέθη. Το αποτέλεσμα θα περιείχε μεν πακέτα ACK αλλά θα περιείχε και την χρήσιμη πληροφορία που θέλαμε. Αυτό το πετύχαμε με ερώτημα στην βάση που περιείχε και υποερωτήματα. Ακολουθεί σχετικό παράδειγμα:

```
SELECT p.packetsarrivaltime,p.packetsize,p.payloadsize,p.payloadratio,p.packetsratiotopreviouspacket,p.packettimedifferencetopreviouspacket
FROM packets AS p INNER JOIN sessions s ON (p.idsessions = s.idsessions) INNER JOIN dataset d ON (s.iddataset=d.iddataset) WHERE s.iddataset =
2 AND p.idsessions NOT IN(SELECT s.idsessions FROM packets AS p INNER JOIN sessions s ON (p.idsessions = s.idsessions) INNER JOIN dataset d
ON (s.iddataset=d.iddataset) WHERE s.iddataset = 2 AND p.packetsize IN (54) GROUP BY s.idsessions) AND p.idsessions IN
(SELECT s.idsessions FROM packets AS p INNER JOIN sessions s ON (p.idsessions = s.idsessions)
INNER JOIN dataset d ON (s.iddataset=d.iddataset) WHERE s.iddataset = 2 AND s.isMalicious = 0
AND p.packetsize  IN (1394,60,260,.....,1411,1552,1413,1349) ORDER BY p.idsessions,p.packetindx;
```

Το ερώτημα είναι παρόμοιο με της Ενότητας 3.3.4.2 με την διαφορά ότι έχουν ενσωματωθεί τα δύο υποερωτήματα. Με το πρώτο, απορρίπτουμε εντελώς όλες εκείνες τις συνόδους με συγκεκριμένο μέγεθος πακέτου, εδώ 54 bytes, ενώ με το δεύτερο επιστρέφουμε όλες τις συνόδους που περιέχουν τα μεγέθη στην παρένθεση (για χάρη οικονομίας χώρου παραθέτουμε την αρχή και το τέλος της λίστας). Με τον τρόπο αυτό έχουμε ακριβώς εκείνες τις συνόδους με χρήσιμη πληροφορία. Εδώ να σημειώσουμε ότι ενώ στην φυσιολογική κίνηση κατέστη εύκολο να εξαιρέσουμε όλα τα σήματα ACK με μέγεθος 54 bytes, στην περίπτωση της κακόβουλης, όπου τα αντίστοιχα πακέτα ήταν 60 bytes, κατέστη αδύνατο καθώς το σύνολο των συνόδων TCP περιείχε τέτοια πακέτα. Οπότε και το αντίστοιχο ερώτημα περιείχε μόνο ένα υποερώτημα.

Πέρα, όμως, από τα παραπάνω, περεταίρω βελτιστοποίηση κρίθηκε απαραίτητη και ιδιαίτερα στην περίπτωση της κακόβουλης κίνησης όπου οι επιλογές μας ήταν πιο περιορισμένες. Αρχικά συνδυάζοντας τα δυο προαναφερθέντα ερωτήματα, εξακριβώσαμε  εκ νέου την συχνότητα των πακέτων ανά μέγεθος.

```
SELECT p.packetsize, COUNT(p.packetsize) AS frequency
FROM packets AS p INNER JOIN sessions s ON (p.idsessions = s.idsessions) INNER JOIN dataset d ON (s.iddataset=d.iddataset) WHERE s.iddataset =
6 AND s.isMalicious = 1 AND s.idsessions IN(SELECT s.idsessions FROM packets AS p INNER JOIN sessions s ON (p.idsessions = s.idsessions) INNER
JOIN dataset d ON (s.iddataset=d.iddataset) WHERE s.iddataset = 6 AND s.isMalicious = 1 AND p.packetsize  IN (70,2974,.....1013,4065) GROUP BY
s.idsessions) GROUP BY p.packetsize ORDER BY frequency desc;
```

Με αυτό τον τρόπο μπορέσαμε να διαπιστώσουμε αρχικά αν το πρόβλημα έχει ελαχιστοποιηθεί ενώ ανάλογα με την εναπομείνασα πληροφορία που προέκυπτε από το προηγούμενο ερώτημα, μπορέσαμε να κρίνουμε αν υπάρχουν περιθώρια περεταίρω βελτιστοποίησης.  Με παράμετρο το μέγεθος των πακέτων σηματοδοσίας, εκτελώντας σχετικό ερώτημα εντοπίσαμε εφόσον υπήρχαν συνόδους με συγκέντρωση μεγάλης ποσότητας τέτοιων πακέτων και προχωρούσαμε σε εκ νέου απόρριψη τους.

Η παραπάνω διαδικασία πραγματοποιήθηκε για όλα τα σε δεδομένων, αλλά κυρίως για την κακόβουλη κίνηση καθώς δεν εντοπίσαμε κάποιον πιο πρακτικό τρόπο. Επιπλέον, συμπεριλάβαμε και νέα μεγαλύτερα σετ δεδομένων κακόβουλης κίνησης στα ήδη υπάρχοντα, από τις ίδιες πηγές, αλλά διαπιστώσαμε ότι το πρόβλημα ήταν εντονότερο όσο μεγάλωνε το αρχείο pcap.

## 5.4 Αλλαγές στον κώδικα και στο τελικό σετ εκπαίδευσης

Ως απόρροιας των θεμάτων που περιγράψαμε στην προηγούμενη ενότητα και δεδομένου ότι οι παραπάνω ενέργειες περιορίζουν τον αντίκτυπο του προβλήματος χωρίς να τον εξαλείφουν πλήρως, χρειάστηκε να προχωρήσουμε σε αλλαγές τόσο στον κώδικα του Walk Forward Validation όσο και στην διαδικασία δημιουργίας του τελικού σετ καθώς και στο μέγεθός του.

### 5.4.1 Διάσπαση του Σετ Εκπαίδευσης (Dataset Split)

Όπως αναλύσαμε στο προηγούμενο Κεφάλαιο, Ενότητα 4.2.2, κατά την διαδικασία του validation το μοντέλο εκπαιδεύεται με το μεγαλύτερο μέρους του σετ εκπαίδευσης ενώ ένας αριθμός παρατηρήσεων μένει για την αξιολόγηση της ακρίβειας. Με την ύπαρξη, ωστόσο, ίδιας πληροφορίας το μοντέλο επιδείκνυε τέλεια ακρίβεια, *Mean Absolute Error = 0*, καθώς τύχαινε στο τελευταίο subset, τα δεδομένα εκπαίδευσης να είναι ακριβώς ίδια με τα δεδομένα αξιολόγησης.

Για να αποφύγουμε την παραπάνω περίπτωση, να υπάρχει εξάρτηση από το τελευταίο ουσιαστικά subset του σετ εκπαίδευσης, αποφασίσαμε να διασπάσουμε το συνολικό σετ δεδομένων σε πέντε μέρη και να εκτελέσουμε την διαδικασία του walk forward validation σε κάθε ένα ξεχωριστά ενώ στο τέλος θα εξάγουμε έναν μέσο όρο από τα πέντε Mean Absolute Errors (MAE). Ακολουθεί ο προσαρμοσμένο κώδικας python:

- Dataset split

```
# Split Dataset into 4 parts
def dataset_split(data):
    dt_temp = np.array_split(data, 5)
    dat1 = dt_temp[0]
    dat2 = dt_temp[1]
    dat3 = dt_temp[2]
    dat4 = dt_temp[3]
    dat5 = dt_temp[4]
    return dat1, dat2, dat3, dat4, dat5
```

- Υπόλοιπος Κώδικας

```
# load dataset
filename = 'dtset01.csv'
names = ['Arrival_Time', 'Packet_Size', 'Payload_Size', 'Payload_Ratio',
         'Ratio_to_Previous', 'Time_Difference', 'Flag']
dt = read_csv(filename, parse_dates=True, index_col=0, names=names)

values = dt.values

# Scale the data before fitting
values = values.astype('float32')
scaler = StandardScaler()
```

```
scaled = scaler.fit_transform(values)

lag_obs = 6
pred = 1
val_size = 100

data = series_to_supervised(scaled, lag_obs, pred)

dst1, dst2, dst3, dst4, dst5 = dataset_split(data)

print("First Walk Forward validation commencing")
mae1 = walk_forward_validation(dst1, val_size, lag_obs)
print('MAE1:', mae1)
print("Second Walk Forward validation commencing")
mae2 = walk_forward_validation(dst2, val_size, lag_obs)
print('MAE2:', mae2)
print("Third Walk Forward validation commencing")
mae3 = walk_forward_validation(dst3, val_size, lag_obs)
print('MAE3:', mae3)
print("Forth Walk Forward validation commencing")
mae4 = walk_forward_validation(dst4, val_size, lag_obs)
print('MAE4:', mae4)
print("Fifth Walk Forward validation commencing")
mae5 = walk_forward_validation(dst5, val_size, lag_obs)
print('MAE5:', mae5)

mae_avg = (mae1 + mae2 + mae3 + mae4 + mae5)/5

print('MAE:', mae_avg)
```

### 5.4.2 Σύνθεση & Μέγεθος των τελικών σετ εκπαίδευσης

Ενώ όπως αναφέραμε έγιναν προσπάθειες περιορισμού της επαναλαμβανόμενης πληροφορίας, στις περισσότερες περιπτώσεις είχαμε και απώλεια χρήσιμης πληροφορίας καθώς πολλές μεγάλες σύνοδοι με ACK πακέτα είχαν και πακέτα χρήσιμα για την εκπαίδευση του μοντέλου μας. Επιπλέον, επιβεβαιώσαμε μέσω δημιουργίας καινούργιων σετ δεδομένων κίνησης, ότι το πρόβλημα παραμένει απαράλλαχτο ανεξάρτητα από το σενάριο χρήσης. Βάσει των παραπάνω, αποφασίστηκε ότι το αρχικό μέγεθος σετ εκπαίδευσης, των 100000 πακέτων, δεν ωφελούσε την εκπαίδευση του μοντέλου μας ενώ θεωρητικά θα μπορούσε να την επηρεάσει και αρνητικά προκαλώντας overfitting [58]. Οπότε καταλήξαμε στην δημιουργία δύο σετ δεδομένων εκπαίδευσης των 60000 πακέτων. Το συγκεκριμένο μέγεθος προέκυψε εμπειρικά βάσει των στατιστικών των συνολικών πακέτων χρήσιμης πληροφορίας

Επιπλέον, με σκοπό να επιλύσουμε και το πρόβλημα του Walk Forward Validation, που περιγράψαμε στην προηγούμενη ενότητα, αποφασίσαμε να συνθέσουμε το τελικό σετ εκπαίδευσης με διαφορετικό τρόπο για την κατανόηση του οποίου βοηθάει ο παρακάτω πίνακας.

| | Walk Forward Validation 1st Split | | | | Walk Forward Validation 2nd Split | | | |
|---|---|---|---|---|---|---|---|---|
| | Training | | Validation | | Training | | Validation | |
| | Normal | Malicious | Normal | Malicious | Malicious | Normal | Malicious | Normal |
| Packets Quantity | 5950 | 5950 | 50 | 50 | 5950 | 5950 | 50 | 50 |
| | Walk Forward Validation 3rd Split | | | | Walk Forward Validation 4th Split | | | |
| | Training | | Validation | | Training | | Validation | |
| | Normal | Malicious | Normal | Malicious | Malicious | Normal | Malicious | Normal |
| Packets Quantity | 5950 | 5950 | 50 | 50 | 5950 | 5950 | 50 | 50 |
| | Walk Forward Validation 5th Split | | | |
| | Training | | Validation | |
| | Normal | Malicious | Normal | Malicious |
| Packets Quantity | 5950 | 5950 | 50 | 50 |

Σε κάθε διαχωρισμό (split) του κάθε σετ εκπαίδευσης, κατά την διάρκεια του Walk Forward Validation, ακολουθώντας την παραπάνω στρατηγική κατανομής των δεδομένων μας, είχαμε ως στόχο την ελαχιστοποίηση της πιθανότητας σε ένα subset εκπαίδευσης να έχουμε επαναλαμβανόμενη πληροφορία ενώ ήταν πιο εύκολο να ελέγξουμε και το validation subset. Στο δεύτερο σετ εκπαίδευσης ακολουθήσαμε την αντίστροφη σειρά δημιουργίας από εκείνη του πρώτου.

# 6 Παράθεση Αποτελεσμάτων & Συμπεράσματα

## 6.1 Εισαγωγή

Έχοντας πραγματοποιήσει την βελτιστοποίηση που περιγράψαμε στο προηγούμενο κεφάλαιο και έχοντας δημιουργήσει τα δύο σετ εκπαίδευσης, προχωρήσαμε στην εκτέλεση του Walk Forward Validation αρχικά για να διαπιστώσουμε αν λύθηκε το πρόβλημα που είχαμε με το αρχικό σετ, με το μηδενικό Mean Absolute Error, και στην συνέχεια να εξετάσουμε την απόδοση του μοντέλου μας. Τέλος εκτελώντας τον κώδικα της πρόβλεψης, παραθέτουμε τα αποτελέσματα μας.

## 6.2 Αξιολόγηση (Validation)

Εκτελώντας τον παραλλαγμένο κώδικα του Walk Forward Validation, το σετ εκπαίδευσης διασπάτε σε πέντε μέρη, και πραγματοποιείται η φάση αξιολόγησης της ακρίβειας του μοντέλου για το κάθε ένα ξεχωριστά χρησιμοποιώντας κάθε φορά ένα σετ αξιολόγησης των 100 παρατηρήσεων. Αφού ολοκληρωθεί η αξιολόγηση και για τα πέντε μέρη, εξάγεται ένας μέσος όρος όπως από τα πέντε Mean Absolute Error. Είναι σημαντικό να σημειώσουμε σε αυτό το σημείο ότι ο MAE, προκύπτει συνολικά και από τις έξι προβλέψεις, μία για κάθε γνώρισμα. Αυτό ουσιαστικά σημαίνει ότι η απώλεια ακρίβειας από την πρόβλεψη της τιμής ενός γνωρίσματος θα καλυφθεί από την καλή ακρίβεια στον υπολογισμό ενός από τα υπόλοιπα[56].

Στο παρακάτω στιγμιότυπο, διακρίνονται οι αναμενόμενες τιμές και εκείνες που προβλέφθηκαν για το πρώτο Walk Forward Validation στο πρώτο σετ εκπαίδευσης, ενώ στο τέλος προκύπτει ο MAE που αφορά στο πρώτο διαχωρισμό.

```
82)>expected=[ 2.5458295  2.5833693  1.4596246 -0.2338653 -0.16312361  1.], predicted=[ 1.77459749  1.80975526  1.42201448 -0.04611139 -0.1632069   1.]
83)>expected=[ 2.5458295  2.5833693  1.4596246 -0.1515613 -0.16312361  1.], predicted=[ 2.50507194  2.54246167  1.42712503  0.06366304 -0.16319769  1.]
84)>expected=[ 3.5220146  3.5624669  1.4670231 -0.08330876 -0.1631236   1.], predicted=[ 1.46972962  1.50394819  1.34960028 -0.14994751 -0.16318064  1.]
85)>expected=[ 1.5696447  1.6042715  1.4449626 -0.2542528 -0.1631236  1.    ], predicted=[ 2.66849395  2.70639953  1.44634671 -0.00781381 -0.16316584  1.]
86)>expected=[ 0.59346    0.6251738  1.402022  -0.2533205 -0.1631236  1.    ], predicted=[ 2.33869259  2.37560599  1.44125984 -0.01119363 -0.16317485  1.]
87)>expected=[ 4.4981995  4.5415645  1.4714838  0.6479942  -0.16309257  1.], predicted=[ 1.96830397  2.0040961   1.40887302 -0.05566496 -0.16318131  1.]
88)>expected=[ 0.59346    0.6251738  1.402022  -0.31616926 -0.16309254  1.], predicted=[ 2.06708315  2.10317812  1.40427499 -0.11696544 -0.16315698  1.]
89)>expected=[ 1.5696447  1.6042715  1.4449626  0.04832757 -0.16309254  1.], predicted=[ 2.59538444  2.63307178  1.41724298  0.13731269 -0.16317029  1.]
90)>expected=[ 1.5696447  1.6042715  1.4449626 -0.1515613  -0.16309252  1.], predicted=[ 1.69962784  1.73445327  1.29597933 -0.11438408 -0.16317451  1.]
91)>expected=[ 4.4981995  4.5415645  1.4714838  0.1537162 -0.1630925   1.    ], predicted=[ 2.16832827  2.20473333  1.44036955 -0.00958303 -0.16316897  1.]
92)>expected=[ 0.59346    0.6251738  1.402022  -0.31616926 -0.16309248  1.], predicted=[ 1.79630092  1.83149927  1.3815111  -0.14319966 -0.16316812  1.]
93)>expected=[ 0.59346    0.6251738  1.402022  -0.1515613  -0.16309246  1.], predicted=[ 3.26275922  3.30243794  1.45727714  0.26704762 -0.16314097  1.]
94)>expected=[ 0.59346    0.6251738  1.402022  -0.1515613  -0.16306159  1.], predicted=[ 1.70143579  1.73638463  1.41153313 -0.08451197 -0.16315251  1.]
95)>expected=[ 3.5220146  3.5624669  1.4670231  0.44810534 -0.16306157  1.], predicted=[ 2.13288475  2.16913976  1.41625565 -0.01311688 -0.16313439  1.]
96)>expected=[-0.37737584 -0.35392392 -0.96289146 -0.35677055 -0.16362579  1.], predicted=[ 1.38387215  1.41792038  1.36620478 -0.18290452 -0.16315363  1.]
97)>expected=[-0.3787131  -0.35392392 -0.96289146 -0.15824783 -0.16361283  1.], predicted=[ 2.42459156  2.4557981   0.3890703  10.22987666 -0.16338295  0.578]
```

```
98)>expected=[-0.361329  -0.33246425 -0.05037695 -0.06173911 -0.16361235  1.], predicted=[ 0.45312742  0.39374125  0.58221063  3.61041315 -0.16340855  0.702]
99)>expected=[-0.3787131  -0.35392392 -0.96289146 -0.21422797 -0.1636123   1.], predicted=[-0.10210893 -0.08102927  0.11475529  0.45919781 -0.16304172  1. ]
100)>expected=[-0.37737584 -0.35392392 -0.96289146 -0.35677055 -0.16362579  1.], predicted=[-0.19995627 -0.17950369 -0.11111378  0.55440552 -0.16287152  1.]

MAE: 0.5992147235016293
```

Παρατηρήσαμε ότι το πρόβλημα που υπήρχε στην αρχή δεν παρατηρείται πλέον με το βελτιωμένο σετ εκπαίδευσης αλλά και με την αλλαγή στον κώδικα. Στον παρακάτω πίνακα παρατίθενται τα επί μέρους MAE καθώς και το συνολικό για τα 2 σετ εκπαίδευσης:

| | Σετ Εκπαίδευσης 01 | Σετ Εκπαίδευσης 01 |
|---|---|---|
| MAE_01 | 0.5992147 | 0.2946462 |
| MAE_02 | 0.2006734 | 0.0536266 |
| MAE_03 | 0.1214536 | 0.0371975 |
| MAE_04 | 0.2055303 | 0.1311659 |
| MAE_05 | 0.1150761 | 0.0953200 |
| **MAE** | **0.2483896** | **0.1223912** |

## 6.3 Πρόβλεψη (Prediction)

Έχοντας αξιολογήσει την απόδοση του μοντέλου μας, εκπαιδεύουμε το μοντέλο μας με το συνολικό μέγεθος των σετ εκπαίδευσης και χρησιμοποιώντας σαν είσοδο τα τελευταία έξι πακέτα γίνεται πρόβλεψη του επόμενου. Παρακάτω παραθέτουμε την έξοδο για το κάθε σετ εκπαίδευσης.

- Πρόβλεψη με το 1ο Σετ Εκπαίδευσης

```
Input:
[ 66.   0.  0.      100.      0.47971106   0.
  74.   0.  0.        1.      0.           0.
 124.  58. 46.774193  167.56757   0.0489099    0.
  66.   0.  0.       53.225807  0.10672903   0.
 157.  91. 57.961784  237.87878   0.10680699   0.
  70.   4.  5.714286   44.585987  0.15896082   0.])

########## Predicted Packet ############
[[12065.58749452 11993.24086325    94.73807807   274.31279958
    4.07568482     0.00411765]]
```

- Πρόβλεψη με το 2ο Σετ Εκπαίδευσης

```
Input:
[111.       57.       51.351353   33.134327    1.995759    1.
 111.       57.       51.351353  100.         1.995765    1.
```

```
 70.    16.    22.857143  63.063065   2.0225182  1.
 70.    16.    22.857143 100.       2.022524   1.
 70.    16.    22.857143 100.       2.026598   1.
 70.    16.    22.857143 100.       2.026603   1. ])

########## Predicted Packet ##########
[[214528.99340522   14.34800085  3478.35707476  3419.91400343
   86.94437102   71.23904303]]
```

## 6.4 Παρατηρήσεις & Συμπεράσματα

Κάνοντας μια ανασκόπηση, στην παρούσα Διπλωματική εργασία, παρουσιάσαμε ένα μοντέλο Μηχανικής Μάθησης για την δημιουργία ψευδούς δικτυακής κίνησης μέσω πρόβλεψης του επόμενου πακέτου, δεδομένης ως εισόδου προηγούμενες παρατηρήσεις. Επιλέξαμε να εκπαιδεύσουμε το μοντέλο μας με υπάρχουσα δικτυακή κίνηση τόσο φυσιολογική, παραγμένη από εμάς μέσα από διαφορετικά σενάρια χρήσης, όσο και κακόβουλη από υπάρχοντα datasets ελεύθερα διαθέσιμα στο Διαδίκτυο.

Αναφορικά με το σετ εκπαίδευσης, το χειριστήκαμε ως timeseries/sequential dataset το οποίο προϋπέθετε ότι θα έπρεπε να διατηρήσουμε την σειρά των παρατηρήσεων άθικτη. Αυτό, ωστόσο, δημιούργησε και το πρώτο πρόβλημα. Όπως αναλύσαμε στο 5ο Κεφάλαιο, λόγω της φύσης της αρχικής πληροφορίας υπήρχε πληθώρα επαναλαμβανόμενων παρατηρήσεων. Εξαιτίας της πρώτης προϋπόθεσης, δεν υπήρχε εύκολος τρόπος ολοκληρωτικού καθαρισμού της πληροφορίας παρά μόνο επιλεκτικής εξαίρεσής της. Αν και το πρόβλημα περιορίστηκε, δεν επιλύθηκε.

Αυτό φάνηκε κατά την φάση της αξιολόγησης του μοντέλου μας, όπου εφαρμόστηκε η τεχνική του «Κυλιόμενου παραθύρου», στην οποία οι επαναλαμβανόμενες παρατηρήσεις φάνηκε να επηρέασαν το Mean Absolute Error εμφανίζοντας αρκετά αισιόδοξα αποτελέσματα. Και αυτό το συμπεράναμε από την φάση της πρόβλεψης.

Κατά την πρόβλεψη, όπου χρησιμοποιήσαμε και τα δύο σετ εκπαίδευσης, παρατηρείται αρκετή απόκλιση ως προς τα μεγέθη των τιμών. Και στα δύο σετ και όσες φορές και να εκτελέσαμε τον κώδικα για την πρόβλεψη, οι τιμές ήταν αδικαιολόγητα ψηλές. Μια βασική υπόθεση έχει να κάνει με την διαδικασία της κανονικοποίησης (normalization). Με μία γρήγορη αναζήτηση διαπιστώσαμε ότι για την περίπτωση των Random Forest regressors υπάρχει μια διαφωνία [59] για το αν χρειάζεται ή όχι κανονικοποίηση. Σύμφωνα με τα περισσότερα ευρήματα, δεν θα έπρεπε να υπάρχουν ιδιαίτερες αποκλίσεις στα αποτελέσματα μεταξύ εφαρμογής ή μη κανονικοποίησης στα δεδομένα εισόδου. Εντούτοις, μετά την εκτέλεση του Walk Forward Validation με το πρώτο σετ εκπαίδευσης χωρίς να κανονικοποιήσουμε τις τιμές των γνωρισμάτων, διαπιστώσαμε ότι ο MAE ξεπερνούσε τις 300 μονάδες το οποίο έδειχνε μεγάλη απόκλιση από τις αρχικές τιμές αλλά επιπλέον και εξαιρετικά κακή ακρίβεια για το μοντέλο μας.

Εν κατακλείδι, ο αλγόριθμος του Random Forest για regression προβλήματα, και ιδιαίτερα για time series/sequential σετ εκπαίδευσης, αποδείχτηκε ότι έχει αρκετούς περιορισμούς. Αρχικά, δείχνει να έχει μεγάλη ευαισθησία ως προς το σετ εκπαίδευσης όπως φάνηκε και από την παραπάνω συμπεριφορά. Ένας άλλος περιορισμός, είναι η πρόβλεψη πολλαπλών γνωρισμάτων (multivariate multioutput regression). Στην υπάρχουσα βιβλιογραφία, δεν καταφέραμε να εντοπίσουμε ούτε μια περίπτωση πολλαπλής πρόβλεψης σε time series προβλήματα, καθώς στην πλειοψηφία των παραδειγμάτων, γινόταν πρόβλεψη ενός γνωρίσματος μιας παρατήρησης βάσει των τιμών των υπόλοιπων γνωρισμάτων. Για την επίλυση προβλημάτων σαν αυτό της παρούσας μελέτης, όπου συνδυάζεται time series/sequential dataset και ανάγκη πρόβλεψης μελλοντικών τιμών πολλαπλών γνωρισμάτων, η επιλογή αλγορίθμων deep learning, όπως είναι τα LSTM [50], όπου παρέχεται

εγγενής υποστήριξη για multi output regression [60], ενδεχομένως να συμβάλλει στην καλύτερη αντιμετώπιση των προαναφερθέντων προβλημάτων.

# Αναφορές


[1] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, Jul. 2019, doi: 10.1186/s42400-019-0038-7.

[2] F. Strasak, "Detection of HTTPS Malware Traffic," p. 49.

[3] G. Stergiopoulos, A. Talavari, E. Bitsikas, and D. Gritzalis, "Automatic Detection of Various Malicious Traffic Using Side Channel Features on TCP Packets: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I," 2018, pp. 346–362. doi: 10.1007/978-3-319-99073-6_17.

[4] P. B. and the S. community, "Scapy," *Scapy*. https://secdev.github.io/

[5] "scikit-learn: machine learning in Python — scikit-learn 1.1.1 documentation." https://scikit-learn.org/stable/

[6] "MariaDB Foundation," *MariaDB.org*. https://mariadb.org/

[7] "Machine Learning - Μηχανική μάθηση - τι είναι;," *CSC - Computer Science Center*, Nov. 18, 2018. https://www.csc.com.gr/machine-learning-μηχανική-μάθηση-τι-είναι/ (accessed May 11, 2022).

[8] "Machine learning," *Wikipedia*. May 03, 2022.

[9] J. Brownlee, "What is Machine Learning?," *Machine Learning Mastery*, Nov. 16, 2013. https://machinelearningmastery.com/what-is-machine-learning/ (accessed May 08, 2022).

[10] A. L. Samuel, "Some studies in machine learning using the game of checkers," p. 13.

[11] "Machine Learning textbook." http://www.cs.cmu.edu/~tom/mlbook.html (accessed May 09, 2022).

[12] T. Hastie, R. Tibshiriani, and J. Friedman, "The Elements of Statistical Learning," p. 764.

[13] "Supervised learning," *Wikipedia*. Feb. 24, 2022. Accessed: May 29, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Supervised_learning&oldid=1073728937

[14] "What is Supervised Learning?," Jun. 30, 2021. https://www.ibm.com/cloud/learn/supervised-learning (accessed May 29, 2022).

[15] "ΚΕΦΑΛΑΙΟ 4 - Μηχανική Μάθηση." http://repfiles.kallipos.gr/html_books/93/04a-main.html (accessed Jun. 01, 2022).

[16] "What is Unsupervised Learning?," Mar. 31, 2022. https://www.ibm.com/cloud/learn/unsupervised-learning (accessed May 29, 2022).

[17] "Unsupervised learning," *Wikipedia*. Apr. 21, 2022. Accessed: May 29, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Unsupervised_learning&oldid=1083933993

[18] "Semi-supervised learning," *Wikipedia*. Oct. 10, 2021. Accessed: May 29, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Semi-supervised_learning&oldid=1049180726

[19] "Reinforcement learning," *Wikipedia*. May 13, 2022. Accessed: May 14, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=1087640491

[20] J. Brownlee, "4 Types of Classification Tasks in Machine Learning," *Machine Learning Mastery*, Apr. 07, 2020. https://machinelearningmastery.com/types-of-classification-in-machine-learning/ (accessed May 12, 2022).

[21] "Bernoulli distribution," *Wikipedia*. May 05, 2022. Accessed: May 29, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Bernoulli_distribution&oldid=1086372421

[22] "Categorical distribution," *Wikipedia*. Jun. 07, 2021. Accessed: May 29, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Categorical_distribution&oldid=1027330818

[23] "Understanding Regression In Machine Learning | Built In." https://builtin.com/data-science/regression-machine-learning (accessed May 14, 2022).

[24] A. A. Khan, "101 of Building Blocks of Machine Learning Algorithm," *Medium*, Oct. 12, 2020. https://blog.devgenius.io/101-of-building-blocks-of-machine-learning-algorithm-19a28e372db (accessed May 29, 2022).

[25] "Six Elements Of Machine Learning - A Beginner's Guide," *DEV Community*. https://dev.to/jamesshah/six-elements-of-machine-learning-a-beginner-s-guide-4i35 (accessed May 29, 2022).

[26] "Decision Tree Algorithm, Explained," *KDnuggets*. https://www.kdnuggets.com/decision-tree-algorithm-explained.html/ (accessed May 17, 2022).

[27] "Decision tree," *Wikipedia*. Apr. 15, 2022. Accessed: May 17, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Decision_tree&oldid=1082910061

[28] C. mvs, "How exactly Decision Trees are built with complete example.," *Analytics Vidhya*, Jan. 10, 2022. https://medium.com/analytics-vidhya/how-exactly-decision-trees-are-built-with-complete-example-dbda4a34cf1d (accessed May 29, 2022).
[29] "Ensemble learning," *Wikipedia*. Jan. 22, 2022. Accessed: May 17, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Ensemble_learning&oldid=1067251577
[30] L. Breiman, "Bagging predictors," *Mach Learn*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/BF00058655.
[31] "Bootstrap aggregating," *Wikipedia*. Apr. 22, 2022. Accessed: May 18, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Bootstrap_aggregating&oldid=1084053196
[32] "Bagging (Bootstrap Aggregation)," *Corporate Finance Institute*. https://corporatefinanceinstitute.com/resources/knowledge/other/bagging-bootstrap-aggregation/ (accessed May 18, 2022).
[33] "Random forest," *Wikipedia*. May 04, 2022. Accessed: May 17, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=1086143689
[34] "Introduction to Random Forest in Machine Learning," *Engineering Education (EngEd) Program | Section*. https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/ (accessed May 19, 2022).
[35] J. Brownlee, "Why Data Preparation Is So Important in Machine Learning," *Machine Learning Mastery*, Jun. 14, 2020. https://machinelearningmastery.com/data-preparation-is-important/ (accessed Apr. 22, 2022).
[36] J. Brownlee, "How to Calculate Correlation Between Variables in Python," *Machine Learning Mastery*, Apr. 26, 2018. https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/ (accessed May 22, 2022).
[37] "Garbage in, garbage out," *Wikipedia*. May 09, 2022. Accessed: May 22, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Garbage_in,_garbage_out&oldid=1086949507
[38] "Internet protocol suite," *Wikipedia*. May 06, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Internet_protocol_suite&oldid=1086546522
[39] "OSI model," *Wikipedia*. Apr. 29, 2022. Accessed: May 03, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=OSI_model&oldid=1085309819
[40] Editor, "Encapsulation," *Network Encyclopedia*, Aug. 20, 2019. https://networkencyclopedia.com/encapsulation/ (accessed May 23, 2022).
[41] "Encapsulation (networking)," *Wikipedia*. Apr. 27, 2022. Accessed: May 23, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Encapsulation_(networking)&oldid=1084904962
[42] "IPv4," *Wikipedia*. May 28, 2022. Accessed: May 29, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=IPv4&oldid=1090281220
[43] "Progress KB - How does a TCP session work," *Progress Software Knowledgebase*. https://knowledgebase.progress.com/articles/Article/P73431 (accessed May 23, 2022).
[44] "Understanding TCP Sequence and Acknowledgment Numbers - PacketLife.net." https://packetlife.net/blog/2010/jun/7/understanding-tcp-sequence-acknowledgment-numbers/ (accessed Jul. 24, 2022).
[45] "Wireshark · Go Deep." https://www.wireshark.org/ (accessed May 29, 2022).
[46] "What Is Packet Capture (PCAP)? - IT Glossary | SolarWinds." https://www.solarwinds.com/resources/it-glossary/pcap (accessed Jun. 23, 2022).
[47] D. Monahan, "Report Summary: Unlocking High Fidelity Security 2019," p. 21, 2019.
[48] "FIRST.org / 27th Annual FIRST Conference / Program." https://www.first.org/conference/2015/program#phands-on-network-forensics (accessed May 24, 2022).
[49] "The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background traffic.," *Stratosphere IPS*. https://www.stratosphereips.org/datasets-ctu13 (accessed May 24, 2022).
[50] "Long short-term memory - Wikipedia." https://en.wikipedia.org/wiki/Long_short-term_memory (accessed May 29, 2022).
[51] J. Brownlee, "Time Series Forecasting as Supervised Learning," *Machine Learning Mastery*, Dec. 04, 2016. https://machinelearningmastery.com/time-series-forecasting-supervised-learning/ (accessed Jun. 01, 2022).
[52] J. Brownlee, "How to Convert a Time Series to a Supervised Learning Problem in Python," *Machine Learning Mastery*, May 07, 2017. https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/ (accessed Jun. 01, 2022).
[53] J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation," *Machine Learning Mastery*, May 22, 2018. https://machinelearningmastery.com/k-fold-cross-validation/ (accessed May 30, 2022).

[54] J. Brownlee, "How To Backtest Machine Learning Models for Time Series Forecasting," *Machine Learning Mastery*, Dec. 18, 2016. https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/ (accessed Jun. 01, 2022).
[55] J. Brownlee, "Random Forest for Time Series Forecasting," *Machine Learning Mastery*, Nov. 01, 2020. https://machinelearningmastery.com/random-forest-for-time-series-forecasting/ (accessed Jun. 01, 2022).
[56] "3.3. Metrics and scoring: quantifying the quality of predictions," *scikit-learn*. https://scikit-learn/stable/modules/model_evaluation.html (accessed Aug. 05, 2022).
[57] "sklearn.preprocessing.StandardScaler," *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (accessed Jun. 02, 2022).
[58] J. Brownlee, "Overfitting and Underfitting With Machine Learning Algorithms," *Machine Learning Mastery*, Mar. 20, 2016. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ (accessed Jul. 28, 2022).
[59] J. Fernandez, "How data normalization affects your Random Forest algorithm," *Medium*, Jun. 20, 2022. https://towardsdatascience.com/how-data-normalization-affects-your-random-forest-algorithm-fbc6753b4ddf (accessed Aug. 12, 2022).
[60] J. Brownlee, "How to Develop Multi-Output Regression Models with Python," *Machine Learning Mastery*, Mar. 26, 2020. https://machinelearningmastery.com/multi-output-regression-models-with-python/ (accessed Aug. 13, 2022).