ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

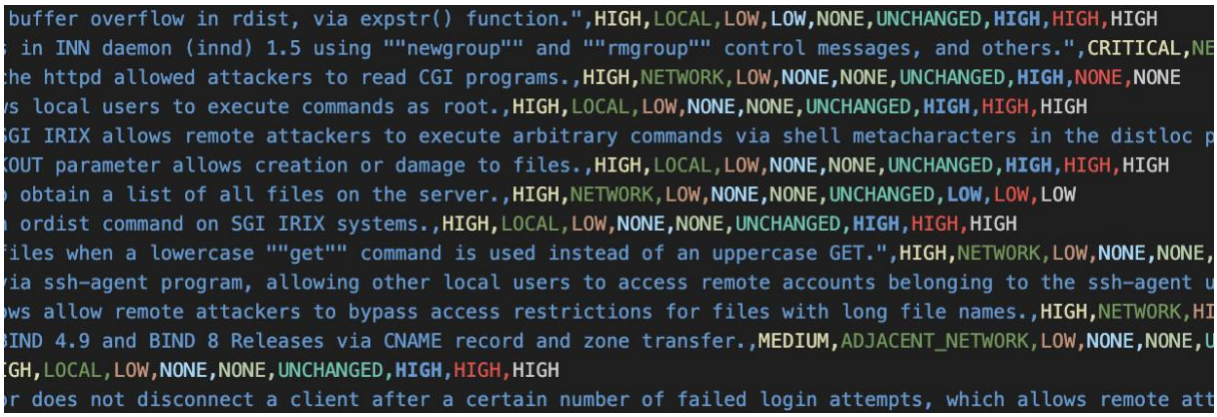
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Μοντέλων Πρόβλεψης Κινδύνου Ευπαθειών χρησιμοποιώντας Τεχνικές Μηχανικής Μάθησης

Κωδικός: 25180



Φοιτητής
Κωνσταντίνος Δημητρίου
Αρ. Μητρώου: 2020202

Επιβλέπων
Χαράλαμπος Μπράτσας
Επίκουρος Καθηγητής

Ημερομηνία 10 Σεπτεμβρίου 2025

Ανάπτυξη Μοντέλων Πρόβλεψης Κινδύνου Ευπαθειών χρησιμοποιώντας Τεχνικές Μηχανικής Μάθησης

Κωδικός Δ.Ε. : 25180

Κωνσταντίνος Δημητρίου
Χαράλαμπος Μπράτσας

13-03-2025
10-09-2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιέρωση»

- *Στη σύντροφό μου, για την κατανόηση, την υποστήριξη, την ατέλειωτη πίστη στις ικανότητες μου, την υπομονή και την εμψύχωση στις πιο απαιτητικές στιγμές.*
- *Στην οικογένειά μου, για την αδιάκοπη στήριξη, την πίστη τους σε εμένα και την καθοδήγησή τους σε κάθε βήμα της ζωής μου.*
- *Στον προπονητή μου, που με δίδαξε πειθαρχία, επιμονή και δύναμη — αξίες που μεταφέρθηκαν από το γυμναστήριο στην ακαδημαϊκή μου πορεία.*
- *Και στους φίλους και συμφοιτητές μου, που με στήριξαν με συναδελφικότητα και χιούμορ μέχρι το τέλος αυτής της διαδρομής.*

Πρόλογος

Η παρούσα διπλωματική εργασία αποτελεί το επιστέγασμα των σπουδών μου στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων. Ο λόγος που επέλεξα το συγκεκριμένο θέμα σχετίζεται άμεσα με το έντονο ενδιαφέρον μου για την ασφάλεια πληροφοριακών συστημάτων, τη μηχανική μάθηση και την επεξεργασία φυσικής γλώσσας (NLP). Η πρόκληση της ανάλυσης και πρόβλεψης του κινδύνου ευπαθειών, με τη χρήση προηγμένων τεχνικών τεχνητής νοημοσύνης, αποτέλεσε για μένα μια εξαιρετική ευκαιρία να εμβαθύνω σε ένα σύγχρονο και ραγδαία εξελισσόμενο πεδίο.

Μέσα από αυτήν τη διαδικασία, απέκτησα πολύτιμες τεχνικές γνώσεις, ανέπτυξα την ερευνητική μου ικανότητα και ενίσχυσα την αναλυτική μου σκέψη. Πιστεύω ότι η εργασία αυτή θα αποτελέσει ισχυρό θεμέλιο για τα επόμενα ακαδημαϊκά ή επαγγελματικά μου βήματα στον τομέα της κυβερνοασφάλειας και της τεχνητής νοημοσύνης.

Περίληψη

Η ραγδαία αύξηση των ψηφιακών απειλών και των ευπαθειών σε πληροφοριακά συστήματα έχει καταστήσει κρίσιμη την ανάπτυξη μεθόδων πρόβλεψης και αξιολόγησης κινδύνου. Στο πλαίσιο της παρούσας διπλωματικής εργασίας αναπτύχθηκε και αξιολογήθηκε ένα σύστημα πρόβλεψης βασικών μετρικών του CVSS v3.1, με βάση τις περιγραφές ευπαθειών που δημοσιεύονται στη βάση δεδομένων CVE (Common Vulnerabilities and Exposures). Η μεθοδολογία στηρίχθηκε σε τεχνικές Επεξεργασίας Φυσικής Γλώσσας (NLP) για τη μετατροπή των περιγραφών σε αριθμητικές αναπαραστάσεις, οι οποίες αποτέλεσαν είσοδο για αλγορίθμους μηχανικής μάθησης.

Η πειραματική διαδικασία οργανώθηκε σε έξι διακριτά σενάρια (E1–E6), με στόχο τη συστηματική διερεύνηση διαφορετικών παραμέτρων. Εξετάστηκαν παραδοσιακές αναπαραστάσεις κειμένου (TF-IDF), προεκπαιδευμένα μοντέλα embeddings (Sentence-BERT) και υβριδικοί συνδυασμοί τους, με κοινό ταξινομητή τον XGBoost σε σχήμα multi-output classification. Η αξιολόγηση πραγματοποιήθηκε με πολλαπλές μετρικές (Accuracy, Precision, Recall, Macro-F1), καθώς και με bootstrap resampling για εκτίμηση διαστημάτων εμπιστοσύνης.

Τα αποτελέσματα έδειξαν ότι η απλή στατιστική προσέγγιση TF-IDF υπερέχει σταθερά σε τεχνικά και λεξικοκεντρικά πεδία, ενώ τα SBERT embeddings προσφέρουν οφέλη σε πιο σημασιολογικά απαιτητικές μεταβλητές, χωρίς όμως να ξεπερνούν συνολικά το TF-IDF. Μέσα από τη σταδιακή βελτιστοποίηση, αναπτύχθηκε το τελικό μοντέλο TF-IDF v3 Optimized, το οποίο συνδύασε μεγαλύτερο λεξιλόγιο, sample weights και stratified splits, επιτυγχάνοντας ακρίβεια άνω του 90% σε αρκετές μεταβλητές.

Η εργασία καταδεικνύει ότι τα εργαλεία μηχανικής μάθησης μπορούν να ενισχύσουν ουσιαστικά τη διαδικασία διαχείρισης ευπαθειών, προσφέροντας αυτοματοποιημένη, αξιόπιστη και επεκτάσιμη υποστήριξη σε σενάρια κυβερνοασφάλειας.

Development of Vulnerability Risk Prediction Models using Machine Learning Techniques

Konstantinos Dimitriou

Abstract

The rapid growth of digital threats and software vulnerabilities has made automated risk assessment methods increasingly essential. This thesis focuses on the development and evaluation of machine learning models for predicting the core metrics of CVSS v3.1, using textual descriptions of vulnerabilities published in the CVE (Common Vulnerabilities and Exposures) database. The approach leverages Natural Language Processing (NLP) techniques to transform unstructured vulnerability descriptions into structured numerical representations suitable for classification models.

The experimental process was organized into six scenarios (E1–E6), designed to systematically investigate the impact of various factors such as text preprocessing, n-gram selection, semantic embeddings, and hybrid representations. Traditional statistical methods (TF-IDF), modern neural approaches (Sentence-BERT), and combinations of the two were explored, with XGBoost employed as the core multi-output classifier across all experiments. Evaluation relied on multiple metrics (Accuracy, Precision, Recall, Macro-F1) as well as bootstrap resampling for robust confidence interval estimation.

The results highlighted three key findings. First, TF-IDF consistently outperformed SBERT embeddings in technical, lexicon-driven dimensions (e.g., attackVector), demonstrating the strength of statistical term frequency methods in structured technical text. Second, semantic embeddings provided advantages in abstract categories requiring contextual understanding, but overall lagged behind TF-IDF on this dataset. Third, the combination of iterative experimentation and optimization led to the development of the final TF-IDF v3 Optimized model, which employed a larger vocabulary, stratified sampling, and class weighting. This model achieved strong generalization, with accuracy exceeding 90% in several CVSS metrics.

The thesis demonstrates that even relatively simple statistical NLP approaches, when combined with robust classifiers and carefully designed experimental protocols, can deliver reliable and scalable support tools for vulnerability assessment. Such models have the potential to significantly enhance cybersecurity decision-making by providing automated, accurate, and interpretable predictions.

Ευχαριστίες

Η ολοκλήρωση του πτυχίου μου αποτελεί ένα σημαντικό βήμα για την διαμόρφωση του μέλλοντος μου, τον δρόμο που θα ακολουθήσω, αλλά και μία ώθηση για την πραγματοποίηση των ονείρων μου. Τίποτα απ' όλα αυτά δεν θα ήταν εφικτό χωρίς την υποστήριξη, αγάπη και θαλπωρή της οικογένειας μου. Θα ήθελα σε αυτό το σημείο να ευχαριστήσω τους γονείς μου, ***Έλενα*** και ***Δημήτρη*** για όσα μου προσφέρουν τόσα χρόνια, χωρίς να ζητήσουν αντάλλαγμα, ούτε καν την αγάπη μου. Να ξέρετε πως την έχετε, μαζί με την ατέρμονη ευγνωμοσύνη μου για τον άνθρωπο και άντρα που με βοηθήσατε να γίνω. Θα ήθελα να ευχαριστήσω ξανά τον προπονητή μου, ο οποίος ποτέ δεν έχε πάψει να με στηρίζει και να με ωθεί στο να γίνω καλύτερος, και που ποτέ δεν έχει αμφιβάλλει για τις ικανότητες μου τόσο στον αθλητισμό όσο και στο ακαδημαϊκό κομμάτι. ***Σάββα,*** σε ευχαριστώ.

Τέλος, το πιο σημαντικό άτομο της ζωής μου, τον έρωτα μου, την μέλλουσα σύζυγό μου, ***Εμμανουέλα.*** Τα τελευταία 3 χρόνια έχουν υπάρξει τα πιο ευτυχισμένα χρόνια της ζωής μου, γεμάτα φως, αγάπη, γέλιο, κατανόηση, υποστήριξη και ζεστασιά. Ό,τι κάνω, το κάνω για εμάς. Για το κοινό μας μέλλον, την ζωή που είναι δική μας να απολαύσουμε, τα μέρη που έχουμε να εξερευνήσουμε μαζί, τον κόσμο που έχουμε να δούμε, τις κουζίνες που έχουμε να δοκιμάσουμε, τις φωτογραφίες που θα διακοσμούν το σπίτι μας και την κάθε στιγμή που θα προσπαθώ να σε κάνω όσο ευτυχισμένο με κάνεις εσύ. Σε αγαπώ. Πιο πολύ από το γυμναστήριο. Μέσα από τα βάθη της καρδιάς μου- Σε ευχαριστώ.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
BERT	Bidirectional encoder representations from transformers
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
FIRST	Forum of Incident Response and Security Teams
NLP	Natural Language processing
NVD	National Vulnerability Database
SBERT	Sentence Bidirectional encoder representations from transformers
TF-IDF	Term Frequency-Inverse Document Frequency
XGBoost	eXtreme Gradient Boosting

Κεφάλαιο 1ο: Εισαγωγή

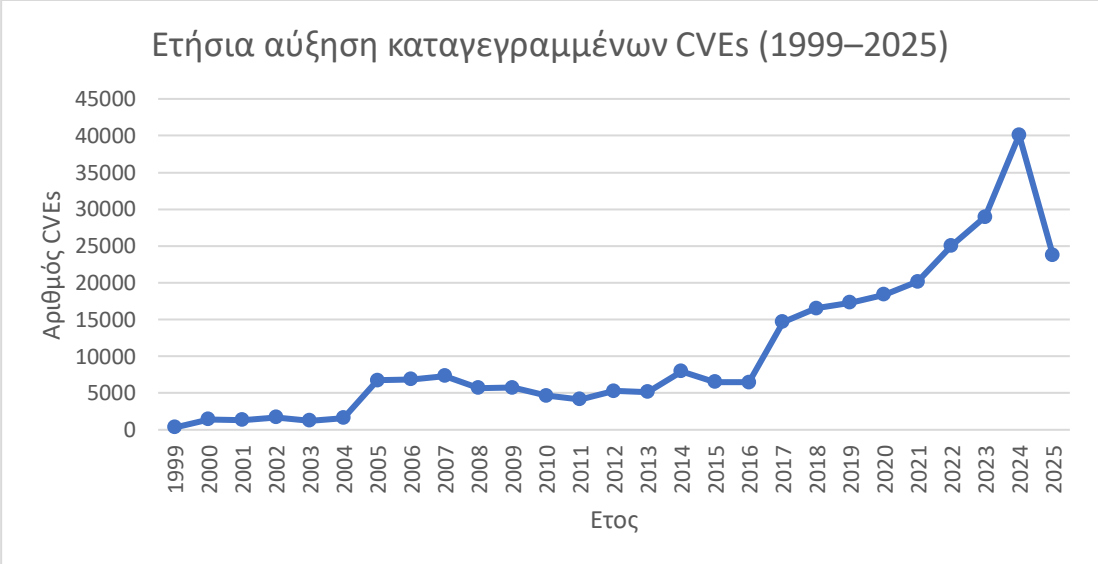
1.1 Υπόβαθρο και Σημασία του Προβλήματος

Η ταχεία ανάπτυξη και διάδοση λογισμικού τις τελευταίες δεκαετίες έχει οδηγήσει σε δραματική αύξηση των ευπαθειών ασφαλείας, οι οποίες αποτελούν κρίσιμη απειλή για οργανισμούς, επιχειρήσεις και κυβερνήσεις. Σύμφωνα με τα στατιστικά στοιχεία του Common Vulnerabilities and Exposures (CVE), ο αριθμός των εγγραφών αυξάνεται κάθε χρόνο με εκθετικούς ρυθμούς, γεγονός που καθιστά αναγκαία την ύπαρξη συστηματικών μεθόδων καταγραφής και αξιολόγησης [1]. Το CVE, σε συνδυασμό με τη National Vulnerability Database (NVD) [2], λειτουργεί ως παγκόσμια βάση αναφοράς, παρέχοντας στους ειδικούς ασφαλείας δομημένη πληροφόρηση για τον εντοπισμό, την κατηγοριοποίηση και τη διαχείριση ευπαθειών.

Για την εκτίμηση της σοβαρότητας μιας ευπάθειας, το πλέον διαδεδομένο πρότυπο είναι το Common Vulnerability Scoring System (CVSS), το οποίο αποδίδει ποσοτικά και ποιοτικά χαρακτηριστικά κινδύνου [3]. Η αξιολόγηση αυτή είναι κρίσιμη για την κατανομή των ενεργειών επιδιόρθωσης (patching) με σειρά προτεραιότητας και τη λήψη αποφάσεων σε επίπεδο κυβερνοασφάλειας. Ωστόσο, η διαδικασία είναι σε μεγάλο βαθμό χειροκίνητη και εξαρτάται από την εμπειρία των αναλυτών, με αποτέλεσμα καθυστερήσεις και ανακρίβειες [4], [5].

Η σημασία του προβλήματος είναι επομένως διπλή: αφενός, ο ολοένα αυξανόμενος αριθμός ευπαθειών καθιστά την παραδοσιακή διαδικασία αξιολόγησης ανεπαρκή. Αφετέρου, η ανακριβής ή καθυστερημένη βαθμολόγηση μπορεί να αφήσει κρίσιμες ευπάθειες εκμεταλλεύσιμες από κακόβουλους χρήστες, δημιουργώντας σοβαρούς κινδύνους για την ασφάλεια πληροφοριακών συστημάτων.

Το Σχήμα 1.1 παρουσιάζει την ετήσια αύξηση των καταγεγραμμένων CVEs από το 1999 έως το 2025 (πρώτο εξάμηνο του 2025), βάσει δεδομένων της NVD, αποτυπώνοντας με σαφήνεια την εκθετική κλιμάκωση του προβλήματος.



Σχήμα 1.1: Ετήσια αύξηση CVEs

1.2 Κίνητρο της Έρευνας

Η συνεχής εξάρτηση από το CVSS για την αξιολόγηση ευπαθειών έχει επανειλημμένα επικριθεί για ζητήματα ακρίβειας και συνέπειας, ιδιαίτερα στις εκδόσεις v2 και v3 [6], [10]. Αν και το CVSS αποτελεί το πιο διαδεδομένο πρότυπο βαθμολόγησης, συχνά αποτυγχάνει να αποτυπώσει τις πραγματικές συνθήκες εκμετάλλευσης μιας ευπάθειας [7]. Η χειροκίνητη φύση της διαδικασίας καθιστά την αξιολόγηση χρονοβόρα και επιρρεπή σε σφάλματα, ιδιαίτερα σε περιβάλλοντα όπου δημοσιεύονται καθημερινά χιλιάδες νέες εγγραφές CVE [1], [3].

Η ανάγκη για αυτοματοποιημένες μεθόδους πρόβλεψης είναι συνεπώς προφανής. Η πρόοδος στη Μηχανική Μάθηση και στην Επεξεργασία Φυσικής Γλώσσας (NLP) προσφέρει νέες δυνατότητες αξιοποίησης των περιγραφών CVE, οι οποίες περιέχουν κρίσιμες πληροφορίες που δεν κωδικοποιούνται πάντοτε στα τυποποιημένα πεδία του CVSS.

Παράλληλα, στη βιβλιογραφία παρατηρείται έλλειψη συστηματικών συγκρίσεων ανάμεσα σε παραδοσιακές στατιστικές μεθόδους (όπως TF-IDF) και σύγχρονες σημασιολογικές αναπαραστάσεις (BERT/SBERT), καθώς και στον τρόπο που αυτές επηρεάζουν την απόδοση ισχυρών αλγορίθμων όπως το XGBoost. Το κενό αυτό αποτελεί το βασικό κίνητρο της παρούσας εργασίας, η οποία διερευνά την αποτελεσματικότητα διαφορετικών προσεγγίσεων αναπαράστασης κειμένου και την ενσωμάτωσή τους σε μοντέλα μηχανικής μάθησης, με στόχο την ακριβή και αυτοματοποιημένη πρόβλεψη μεταβλητών CVSS.

1.3 Σκοπός και Στόχοι της Διπλωματικής Εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη και αξιολόγηση ενός συστήματος πρόβλεψης βασικών μεταβλητών του CVSS v3.1 με βάση τις περιγραφές ευπαθειών που δημοσιεύονται στη βάση NVD. Η εργασία αξιοποιεί τεχνικές Επεξεργασίας Φυσικής Γλώσσας (NLP) και μηχανικής μάθησης, εστιάζοντας στη συγκριτική ανάλυση διαφορετικών αναπαραστάσεων κειμένου και στην αποτίμηση της απόδοσής τους σε πραγματικά δεδομένα.

Οι επιμέρους στόχοι της εργασίας ήταν οι εξής:

- Συλλογή και Οργάνωση Δεδομένων: Ανάκτηση όλων των CVE εγγραφών από τη βάση NVD, με έμφαση στις καταχωρήσεις που διαθέτουν πλήρη μεταδεδομένα CVSS v3.1.
- Δημιουργία Dataset: Σύνθεση ενός ενιαίου συνόλου δεδομένων που περιλαμβάνει το CVE-ID, την περιγραφή της ευπάθειας και τις οκτώ βασικές μεταβλητές του CVSS (baseSeverity, attackVector, attackComplexity, privilegesRequired, userInteraction, scope, confidentialityImpact, integrityImpact, availabilityImpact).
- Προεπεξεργασία Κειμένου: Εφαρμογή βασικών τεχνικών, όπως λημματοποίηση (lemmatization), για τη βελτίωση της ποιότητας του λεξιλογίου.
- Αναπαράσταση Κειμένου: Συστηματική διερεύνηση δύο προσεγγίσεων – TF-IDF και Sentence-BERT – καθώς και της υβριδικής τους εκδοχής, ώστε να μελετηθεί η συμβολή στατιστικών και σημασιολογικών χαρακτηριστικών.
- Εκπαίδευση Μοντέλων: Χρήση του αλγορίθμου XGBoost σε σχήμα multi-output classification, με στόχο την πρόβλεψη όλων των CVSS μεταβλητών.
- Αξιολόγηση Απόδοσης: Αποτίμηση της ποιότητας των προβλέψεων με μετρικές Accuracy και Macro-F1, καθώς και με ανάλυση διαστημάτων εμπιστοσύνης μέσω bootstrap, ώστε να διασφαλιστεί η στατιστική εγκυρότητα.
- Πρακτική Υλοποίηση: Ανάπτυξη proof-of-concept web εφαρμογής με Flask, η οποία επιτρέπει την εισαγωγή νέων CVEs και την αυτόματη πρόβλεψη των CVSS μεταβλητών από το τελικό μοντέλο.

Μέσα από αυτούς τους στόχους, η εργασία συνεισφέρει τόσο σε θεωρητικό επίπεδο, μέσω της συγκριτικής αξιολόγησης διαφορετικών προσεγγίσεων NLP, όσο και σε πρακτικό, παρέχοντας ένα λειτουργικό υπόδειγμα που δείχνει πώς τα αποτελέσματα μπορούν να μετασχηματιστούν σε εργαλείο για την υποστήριξη της διαχείρισης ευπαθειών.

1.4 Συνεισφορά της Εργασίας

Η παρούσα διπλωματική εργασία συνεισφέρει τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο στο πεδίο της αυτοματοποιημένης αξιολόγησης ευπαθειών λογισμικού. Σε θεωρητικό επίπεδο, η εργασία πραγματοποιεί μια συστηματική συγκριτική ανάλυση ανάμεσα σε παραδοσιακές μεθόδους αναπαράστασης κειμένου (TF-IDF) και σύγχρονες σημασιολογικές αναπαραστάσεις (Sentence-BERT), καθώς και στον συνδυασμό τους. Αν και στη βιβλιογραφία υπάρχουν μελέτες που αξιοποιούν είτε TF-IDF [14], είτε embeddings βασισμένα σε BERT [12], η συνδυαστική αξιολόγηση των δύο σε συνδυασμό με το XGBoost αποτελεί καινοτόμο προσέγγιση που αναδεικνύει τα πλεονεκτήματα της συμπληρωματικότητας στατιστικής και σημασιολογικής πληροφορίας.

Σε τεχνικό επίπεδο, η εργασία προτείνει και υλοποιεί ένα ενιαίο pipeline, το οποίο περιλαμβάνει: συλλογή και οργάνωση δεδομένων από τη βάση NVD, προεπεξεργασία των περιγραφών CVE με lemmatization και stop-word removal, εξαγωγή χαρακτηριστικών με TF-IDF και Sentence-BERT, εκπαίδευση πολυεξαγωγικού ταξινομητή (multi-output classifier) με XGBoost.

Η πρακτική συνεισφορά της μελέτης έγκειται στην ανάπτυξη μιας web εφαρμογής με Flask, η οποία λειτουργεί ως proof-of-concept και επιτρέπει σε ερευνητές και επαγγελματίες κυβερνοασφάλειας να εισάγουν περιγραφές ευπαθειών και να λαμβάνουν αυτόματα προβλέψεις για τις αντίστοιχες μεταβλητές CVSS. Με αυτόν τον τρόπο, η εργασία δεν περιορίζεται μόνο σε θεωρητικό επίπεδο αλλά αποδεικνύει την εφικτότητα μετατροπής των ερευνητικών αποτελεσμάτων σε εργαλείο πρακτικής αξίας, το οποίο στο μέλλον θα μπορούσε να εξελιχθεί σε παραγωγικό σύστημα για τη διαχείριση ευπαθειών.

Τέλος, σε ερευνητικό επίπεδο, η εργασία αναδεικνύει την ανάγκη για περαιτέρω διερεύνηση πιο εξελιγμένων μοντέλων, όπως fine-tuned εκδοχές του BERT ή συνδυαστικές αρχιτεκτονικές deep learning, συμβάλλοντας στη διαμόρφωση ενός πλαισίου πάνω στο οποίο μπορούν να βασιστούν μελλοντικές μελέτες.

Η εργασία οργανώνεται σε έξι κεφάλαια, ξεκινώντας με το παρόν, εισαγωγικό Κεφάλαιο 1, όπου παρουσιάζονται το υπόβαθρο και η σημασία του προβλήματος, το κίνητρο της έρευνας, οι στόχοι και η συνεισφορά της μελέτης. Στο Κεφάλαιο 2 πραγματοποιείται η βιβλιογραφική ανασκόπηση: αναλύονται οι βάσεις δεδομένων CVE/NVD, το σύστημα CVSS, οι μέθοδοι αναπαράστασης κειμένου (παραδοσιακές και σύγχρονες), καθώς και τα κυριότερα μοντέλα μηχανικής μάθησης που έχουν εφαρμοστεί στο πεδίο. Το Κεφάλαιο 3 περιγράφει τη μεθοδολογία της εργασίας, από τη συλλογή και προετοιμασία των δεδομένων έως τη διαμόρφωση του πειραματικού πλαισίου και τον καθορισμό των κριτηρίων αξιολόγησης. Στο Κεφάλαιο 4 παρουσιάζεται η υλοποίηση και η ανάλυση των πειραμάτων (E1–E6), όπου εξετάζονται διαφορετικές αναπαραστάσεις κειμένου και συγκρίνονται οι επιδόσεις τους. Στο Κεφάλαιο 5 αναλύεται η ανάπτυξη του τελικού βελτιστοποιημένου μοντέλου και η ενσωμάτωσή του σε εφαρμογή Flask, η οποία λειτουργεί ως proof-of-concept εργαλείο. Τέλος, στο Κεφάλαιο 6 συνοψίζονται τα βασικά συμπεράσματα, αναλύεται η επιστημονική και πρακτική συμβολή της εργασίας, καταγράφονται οι περιορισμοί της και προτείνονται κατευθύνσεις για μελλοντική έρευνα.

Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση

2.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο και η σχετική ερευνητική πρόοδος στον τομέα της αυτόματης αξιολόγησης κινδύνου ευπαθειών λογισμικού με τη χρήση τεχνικών μηχανικής μάθησης και επεξεργασίας φυσικής γλώσσας (NLP). Αρχικά αναλύεται το πλαίσιο των ευπαθειών και οι διεθνείς βάσεις δεδομένων CVE και NVD, οι οποίες αποτελούν τον πυρήνα της τυποποιημένης καταγραφής και διάδοσης πληροφοριών ασφαλείας. Παράλληλα εξετάζεται το σύστημα CVSS, το οποίο έχει καθιερωθεί ως διεθνές πρότυπο για την ποσοτικοποίηση της σοβαρότητας και τη συγκρισιμότητα των ευπαθειών

Στη συνέχεια παρουσιάζονται οι βασικές τεχνικές επεξεργασίας κειμένου που εφαρμόζονται στις περιγραφές CVE, καθώς και οι κυριότερες μεθόδοι αναπαράστασης χαρακτηριστικών, από τις παραδοσιακές προσεγγίσεις (Bag-of-Words, TF-IDF) έως τις σύγχρονες μορφές embeddings (BERT, Sentence-BERT). Εξετάζονται επίσης τα μοντέλα μηχανικής μάθησης που έχουν αξιοποιηθεί στη βιβλιογραφία, με έμφαση στις δενδροβασισμένες μεθόδους (Decision Trees, Random Forests, XGBoost) και στα BERT-based νευρωνικά δίκτυα, τα οποία αντιπροσωπεύουν τις κυρίαρχες κατευθύνσεις του πεδίου

Στη συνέχεια παρουσιάζονται οι κύριες τεχνικές επεξεργασίας κειμένου που εφαρμόζονται στις περιγραφές CVE, ώστε να μετατραπούν σε χρήσιμα χαρακτηριστικά για μοντέλα μηχανικής μάθησης. Οι παραδοσιακές προσεγγίσεις, όπως το Bag-of-Words και το TF–IDF, αν και απλές, παραμένουν ισχυρές αφετηρίες (baseline) με χαμηλό υπολογιστικό κόστος και συχνά ικανοποιητική απόδοση σε τεχνικά κείμενα [12], [18]. Με την πάροδο του χρόνου, αναπτύχθηκαν πιο εξελιγμένες μέθοδοι όπως τα Word2Vec, fastText και Doc2Vec, οι οποίες επιχειρούν να αποτυπώσουν τις σχέσεις ανάμεσα σε λέξεις και προτάσεις μέσω διανυσματικών αναπαραστάσεων.

Η μεγαλύτερη πρόοδος, ωστόσο, προήλθε από τα contextual embeddings. Το BERT [19] και οι παραλλαγές του, όπως το Sentence-BERT [20], μπορούν να κατανοήσουν το νόημα των λέξεων μέσα στο συμφραζόμενο, προσφέροντας σημαντικά καλύτερη απόδοση στην ταξινόμηση περιγραφών CVE [8], [14]. Παράλληλα, νευρωνικά μοντέλα όπως τα CNNs αξιοποιήθηκαν για την ανίχνευση τοπικών μοτίβων [13], ενώ πιο πρόσφατα τα μεγάλα γλωσσικά μοντέλα (LLMs) χρησιμοποιούνται για τη δημιουργία embeddings που βελτιώνουν την πρόβλεψη σε πιο σύνθετες διαστάσεις, όπως Confidentiality, Integrity και Availability [16].

Η παρούσα εργασία επικεντρώνεται σε μια συγκριτική μελέτη ανάμεσα στο TF–IDF και τα contextual embeddings του Sentence-BERT (all-mpnet-base-v2) [21]. Ο συνδυασμός τους με τον αλγόριθμο XGBoost επιλέχθηκε προκειμένου να εξεταστεί αν η παράλληλη χρήση στατιστικών και σημασιολογικών χαρακτηριστικών μπορεί να οδηγήσει σε πιο ακριβή και αξιόπιστη πρόβλεψη μεταβλητών του CVSS.

Τέλος, γίνεται ανασκόπηση των ερευνητικών εργασιών που έχουν εφαρμοστεί μέχρι σήμερα, ώστε να αναδειχθούν οι υφιστάμενες προσεγγίσεις, τα πλεονεκτήματα και οι περιορισμοί τους, καθώς και το επιστημονικό κενό που καλύπτει η παρούσα εργασία. Η ανάλυση αυτή θεμελιώνει τη μεθοδολογική προσέγγιση του επόμενου κεφαλαίου, όπου προτείνεται και αξιολογείται ένα υβριδικό σχήμα που συνδυάζει διαφορετικές τεχνικές αναπαράστασης κειμένου με μοντέλα μηχανικής μάθησης για την πρόβλεψη κρίσιμων μεταβλητών του CVSS

2.2 Η Βάση Δεδομένων Ευπαθειών CVE/NVD

Το Common Vulnerabilities and Exposures (CVE) αποτελεί μια διεθνή πρωτοβουλία για την τυποποιημένη καταγραφή και αναγνώριση ευπαθειών λογισμικού και υλικού [1]. Κάθε εγγραφή αντιστοιχεί σε μια συγκεκριμένη και δημόσια γνωστή ευπάθεια, η οποία συνοδεύεται από ένα μοναδικό αναγνωριστικό της μορφής **CVE-Έτος-Αύξων Αριθμός (π.χ. CVE-2025-12345).** Η χρήση αυτού του τυποποιημένου κωδικού εξασφαλίζει τη συνέπεια στον τρόπο αναφοράς των ευπαθειών και διευκολύνει την κοινή «γλώσσα» επικοινωνίας ανάμεσα σε κατασκευαστές, ερευνητές και οργανισμούς ασφάλειας. Το πρόγραμμα εγκαινιάστηκε το 1999 από τη MITRE Corporation, με την υποστήριξη του U.S. Department of Homeland Security (DHS) [2], και έκτοτε εξελίχθηκε σε σημείο αναφοράς με τη συμμετοχή εξουσιοδοτημένων φορέων καταχώρισης (CVE Numbering Authorities – CNA).

Το CVE λειτουργεί σε στενή σύνδεση με το National Vulnerability Database (NVD) [3], η οποία επεκτείνει κάθε εγγραφή με πρόσθετα Metadata και αξιολογικές πληροφορίες. Ειδικότερα, το NVD ενσωματώνει το Common Vulnerability Scoring System (CVSS) [4], παρέχοντας αποτίμηση της σοβαρότητας, εμπλουτίζει τις καταγραφές με λεπτομέρειες για τον τύπο της επίθεσης, τις απαιτήσεις πρόσβασης, την ανάγκη για αλληλεπίδραση χρήστη, καθώς και τις πιθανές επιπτώσεις σε εμπιστευτικότητα, ακεραιότητα και διαθεσιμότητα. Παράλληλα, συνδέει κάθε εγγραφή με σχετικές αναφορές, ενημερωτικά δελτία, patches ή άλλες πηγές.

Η τεχνική μορφοποίηση των δεδομένων στηρίζεται πλέον στο CVE JSON 5.0/5.1 schema [5], το οποίο εξασφαλίζει δομημένη και μηχανικά αναγνώσιμη μορφή. Αυτό επιτρέπει την αυτοματοποιημένη συλλογή και ανάλυση των δεδομένων και έχει καταστήσει το CVE/NVD μια από τις σημαντικότερες βάσεις για την εκπαίδευση και αξιολόγηση αλγορίθμων μηχανικής μάθησης. Η αξιοποίηση της βάσης δεν περιορίζεται μόνο στη διαχείριση κινδύνων από την πλευρά των οργανισμών· χρησιμοποιείται ευρέως στην έρευνα για την ανάπτυξη μοντέλων πρόβλεψης, καθώς οι περιγραφές των CVE συνιστούν μια ανεξάντλητη πηγή πληροφοριών που δεν αποτυπώνονται πλήρως στις τυποποιημένες μεταβλητές του CVSS.

Ερευνητικές εργασίες όπως των Khazaei et al. (2016) [11] και Aivatoglou et al. (2021) [9] αξιοποίησαν συλλογές από περιγραφές CVE ώστε να αναπτύξουν μοντέλα βασισμένα σε τεχνικές εξόρυξης κειμένου και δέντρα απόφασης για την αυτόματη ταξινόμηση ευπαθειών, ενώ νεότερες μελέτες όπως των Shahid και Debar (2022) [14] προχώρησαν σε πιο εξελιγμένες προσεγγίσεις χρησιμοποιώντας embeddings από BERT-based μοντέλα για την πρόβλεψη μεταβλητών CVSS με μεγαλύτερη ακρίβεια. Η εξέλιξη αυτή καταδεικνύει ότι η σημασία των δεδομένων CVE/NVD υπερβαίνει την απλή καταγραφή και επεκτείνεται στη δημιουργία ενός πλαισίου πειραματισμού και αξιολόγησης που στηρίζει την ανάπτυξη αλγορίθμων πρόβλεψης.

Η κατανόηση της δομής και της λειτουργίας των CVE/NVD επομένως δεν αποτελεί μόνο προϋπόθεση για την ορθή διαχείριση κινδύνων από την πλευρά της βιομηχανίας, αλλά και θεμελιώδη βάση για την ακαδημαϊκή έρευνα. Η παρούσα εργασία εδράζεται σε αυτό το οικοσύστημα δεδομένων, επιδιώκοντας να το αξιοποιήσει τόσο σε επίπεδο αναπαράστασης κειμένου όσο και σε επίπεδο αξιολόγησης μοντέλων πρόβλεψης κινδύνου, ώστε να συμβάλει στη βελτίωση της ακρίβειας και της αξιοπιστίας στην αυτόματη πρόβλεψη CVSS μεταβλητών.

2.3 Το Σύστημα Βαθμολόγησης CVSS

Το Common Vulnerability Scoring System (CVSS) αποτελεί το διεθνώς αναγνωρισμένο πρότυπο για την αποτίμηση της σοβαρότητας των ευπαθειών ασφαλείας και αναπτύχθηκε από τον οργανισμό Forum of Incident Response and Security Teams (FIRST) [4]. Στόχος του συστήματος είναι η παροχή μιας κοινής, ποσοτικοποιημένης κλίμακας αξιολόγησης, η οποία κυμαίνεται από 0.0 έως 10.0, ώστε να καθίσταται εφικτή η αντικειμενική εκτίμηση του επιπέδου κινδύνου και η υποστήριξη αποφάσεων που σχετίζονται με την προτεραιοποίηση διορθώσεων, τη διαχείριση πόρων και την ανάλυση απειλών. Το CVSS χρησιμοποιείται ευρέως τόσο στη βιομηχανία όσο και στην ερευνητική κοινότητα, καθώς προσφέρει ένα κοινό σημείο αναφοράς για την ταξινόμηση ευπαθειών και την ανάπτυξη αυτοματοποιημένων μεθόδων πρόβλεψης κινδύνου [3], [4], [7].

Η δομή του CVSS βασίζεται σε τρεις συμπληρωματικές διαστάσεις. Το Base Score αποτιμά τη σοβαρότητα της ευπάθειας ανεξάρτητα από συγκεκριμένες συνθήκες περιβάλλοντος και αποτελεί το πιο διαδεδομένο και ερευνητικά αξιοποιούμενο τμήμα του συστήματος. Το Temporal Score προσαρμόζει την αξιολόγηση βάσει παραγόντων όπως η διαθεσιμότητα exploit ή patches, ενώ το Environmental Score επιτρέπει την προσαρμογή στις ανάγκες ενός συγκεκριμένου οργανισμού ή υποδομής [4]. Ειδικότερα, το Base Score προκύπτει από μια σειρά μεταβλητών όπως το Attack Vector, το Attack Complexity, τα Privileges Required, η ανάγκη User Interaction, το Scope και οι επιπτώσεις σε Confidentiality, Integrity και Availability. Ανάλογα με την τελική τιμή, κάθε ευπάθεια ταξινομείται σε ποιοτικές κατηγορίες, από None (0.0) έως Critical (9.0–10.0), διευκολύνοντας την πρακτική χρήση της κλίμακας στην καθημερινή διαχείριση κινδύνων [4].

Κατά την εξέλιξή του, το CVSS έχει περάσει από πολλαπλές εκδόσεις με στόχο τη βελτίωση της ακρίβειας και της χρηστικότητάς του. Η έκδοση v2 χαρακτηριζόταν από απλότητα αλλά και από αδυναμία να διαφοροποιήσει αποτελεσματικά κρίσιμους παράγοντες όπως η πολυπλοκότητα της επίθεσης ή οι απαιτήσεις πρόσβασης. Η v3.0, που εισήχθη το 2015, εισήγαγε νέες μεταβλητές, όπως το Scope και τα Privileges Required, παρέχοντας πιο λεπτομερή περιγραφή των συνθηκών εκμετάλλευσης. Η v3.1, που δημοσιεύθηκε το 2019, επικεντρώθηκε κυρίως στη βελτίωση της τεκμηρίωσης και της κατανόησης των ορισμών, χωρίς να μεταβάλει τις βασικές εξισώσεις. Η πιο πρόσφατη έκδοση v4.0, η οποία ανακοινώθηκε το 2023, εισάγει σημαντικές καινοτομίες, όπως τον Automatable Score, τα Supplemental Metrics και νέους μηχανισμούς για την καλύτερη αποτίμηση ευπαθειών σε περιβάλλοντα OT και IoT, αντανακλώντας τη διεύρυνση του πεδίου εφαρμογών και τις αυξημένες απαιτήσεις των σύγχρονων υποδομών [4].

Παρά την πρόοδο, το CVSS έχει αποτελέσει αντικείμενο κριτικής σε αρκετές μελέτες για ζητήματα ακρίβειας και συνέπειας, ιδίως στις εκδόσεις v2 και v3. Η έρευνα των Balsam et al. [6], [7] επισημαίνει ότι οι βαθμολογίες CVSS ενδέχεται να μην αντανακλούν πάντα τον πραγματικό κίνδυνο, είτε λόγω υποκειμενικότητας κατά την απόδοσή τους είτε λόγω περιορισμένης ικανότητας διαφοροποίησης σε σύνθετα περιβάλλοντα. Αν και η v4.0 εισάγει βελτιώσεις μέσω νέων μετρικών όπως τα Threat και Supplemental Metrics, εξακολουθούν να παραμένουν ανοιχτά ερευνητικά ζητήματα σχετικά με την ακριβή αποτύπωση της σοβαρότητας και τη δυνατότητα έγκαιρης χρήσης των βαθμολογιών από τους οργανισμούς.

Το γεγονός ότι οι περιγραφές των CVE περιλαμβάνουν κρίσιμες πληροφορίες που δεν αποτυπώνονται πάντοτε επαρκώς στις μετρικές CVSS καθιστά επιτακτική την ανάγκη για συμπληρωματικές μεθόδους πρόβλεψης. Η αξιοποίηση τεχνικών μηχανικής μάθησης και επεξεργασίας φυσικής γλώσσας μπορεί να εμπλουτίσει ή και να βελτιώσει τη διαδικασία, παρέχοντας αυτόματες και ενδεχομένως ακριβέστερες εκτιμήσεις κινδύνου. Η παρούσα εργασία τοποθετείται σε αυτό το πλαίσιο, διερευνώντας την πρόβλεψη

μεταβλητών CVSS και συνολικών σκορ από περιγραφές ευπαθειών με χρήση παραδοσιακών και σύγχρονων μοντέλων μάθησης.

2.4 Μοντέλα Μηχανικής Μάθησης για Πρόβλεψη Κινδύνου

Η πρόβλεψη της σοβαρότητας ή των μεταβλητών CVSS από περιγραφές CVE έχει εξελιχθεί σε μια από τις πιο δραστήριες ερευνητικές κατευθύνσεις της κυβερνοασφάλειας. Οι περιγραφές αποτελούν κείμενα σε φυσική γλώσσα που συνοψίζουν κρίσιμες λεπτομέρειες σχετικά με την ευπάθεια, το λογισμικό που επηρεάζουν, τις πιθανές συνέπειες και τις συνθήκες εκμετάλλευσης. Επειδή περιέχουν πληροφορίες που δεν έχουν ακόμη αποτυπωθεί στα τυποποιημένα πεδία του CVSS, η αξιοποίησή τους μέσω τεχνικών μηχανικής μάθησης μπορεί να οδηγήσει σε ακριβέστερη και ταχύτερη αποτίμηση κινδύνου [1], [3].

Η βιβλιογραφία έχει προσεγγίσει το πρόβλημα με δύο βασικές στρατηγικές. Η πρώτη στηρίζεται σε παραδοσιακές αναπαραστάσεις κειμένου (BoW, TF-IDF) και κλασικούς αλγορίθμους ταξινόμησης. Η εργασία των Khazaei et al. (2016) [11] ήταν από τις πρώτες που δοκίμασαν Support Vector Machines και Random Forests σε περιγραφές CVE, καταγράφοντας επιδόσεις γύρω στο 70% σε ακρίβεια. Οι Elbaz et al. (2019) [12] υλοποίησαν πρόβλεψη CVSS scores χρησιμοποιώντας γραμμικά μοντέλα με BoW/TF-IDF, με αποτελέσματα μέτριας ακρίβειας, υποδεικνύοντας ότι οι μέθοδοι αυτές είναι απλές αλλά περιορισμένες. Αντίστοιχα, οι Mandal & Kösesoy (2021) [18] εφάρμοσαν πληθώρα κλασικών ταξινομητών (Decision Trees, Logistic Regression, Naive Bayes, k-NN, SVM) με αναπαραστάσεις TF-IDF και Doc2Vec, καταλήγοντας ότι τα Decision Trees σε συνδυασμό με TF-IDF προσέφεραν την καλύτερη επίδοση (97%), γεγονός που καταδεικνύει την ανθεκτικότητα των δενδροβασισμένων μεθόδων σε αραιά χαρακτηριστικά.

Η δεύτερη στρατηγική εστιάζει σε νευρωνικές προσεγγίσεις και contextual embeddings. Οι Zhang et al. (2020) [13] αξιοποίησαν CNNs για πρόβλεψη CVSS base metrics, επιτυγχάνοντας F1-score ≈0.88, με κόστος όμως αυξημένης υπολογιστικής πολυπλοκότητας και χωρίς σύγκριση με απλούστερα baseline μοντέλα. Οι Costa et al. (2022) [8] εφάρμοσαν DistilBERT σε περιγραφές CVE και έδειξαν ότι η προσεκτική προεπεξεργασία (lemmatization, stop-word removal) βελτιώνει την balanced accuracy, ακόμα και σε μοντέλα που υποτίθεται ότι λειτουργούν καλά με ακατέργαστο κείμενο. Οι Shahid & Debar (2022) [14] εισήγαγαν το CVSS-BERT, ένα εξηγησίμο πλαίσιο NLP, το οποίο αποδίδει υψηλή ακρίβεια και ταυτόχρονα παρέχει επεξηγησιμότητα μέσω attention weights. Η μελέτη αυτή έδειξε ότι η εξηγήσιμη τεχνητή νοημοσύνη μπορεί να είναι πρακτικά εφαρμόσιμη σε οργανισμούς ασφάλειας.

Νεότερες εργασίες προχωρούν σε ακόμη πιο προηγμένες μεθοδολογίες. Οι Singh et al. (2022) [15] αξιοποίησαν CNN, BiLSTM και BERT για την ανίχνευση ευπαθειών σε πηγαίο κώδικα, φτάνοντας σε ακρίβεια έως 95%, ενώ οι Mandal & Kösesoy (2023) [10] εξέτασαν χαρακτηριστικά από τον πηγαίο κώδικα Apache Tomcat, καταλήγοντας σε υψηλή ακρίβεια αλλά με περιορισμένο εύρος δεδομένων. Η πιο πρόσφατη γενιά ερευνών, με εκπροσώπους τους Tiwari (2025) [17] και Marchiori et al. (2025) [16], φέρνει στο προσκήνιο τα BERT-based πολυαντικειμενικά μοντέλα και τα Large Language Models (LLMs). Ο Tiwari ανέπτυξε έναν multi-objective ταξινομητή που προβλέπει ταυτόχρονα σοβαρότητα και τύπο ευπάθειας με σχεδόν real-time χρήση, ενώ ο Marchiori έδειξε ότι τα LLMs σε συνδυασμό με embeddings βελτιώνουν την ακρίβεια ειδικά στις μεταβλητές Confidentiality, Integrity και Availability.

Η συγκριτική αποτίμηση αυτών των εργασιών αναδεικνύει τρία κρίσιμα συμπεράσματα. Πρώτον, οι απλές μέθοδοι (TF-IDF με Decision Trees ή XGBoost) εξακολουθούν να αποτελούν ισχυρό baseline με καλή ερμηνευσιμότητα και ανθεκτικότητα σε αραιά δεδομένα. Δεύτερον, τα BERT-based μοντέλα και τα CNNs προσφέρουν σαφή βελτίωση στην ακρίβεια, ιδίως όταν χρησιμοποιούνται σε μεγάλα και ισορροπημένα datasets, αλλά το κόστος και η πολυπλοκότητά τους καθιστούν δύσκολη την υιοθέτηση

σε πρακτικά περιβάλλοντα χωρίς υπολογιστικούς πόρους. Τρίτον, η προσεκτική επιλογή preprocessing (lemmatization, stop-word removal) παραμένει κρίσιμη, ακόμη και στα πιο προηγμένα νευρωνικά δίκτυα, καθώς καθορίζει την ποιότητα των χαρακτηριστικών εισόδου και την ανθεκτικότητα του μοντέλου σε θόρυβο.

Συνολικά, η βιβλιογραφία καταδεικνύει ότι δεν υπάρχει ένα «καθολικά καλύτερο» μοντέλο. Η απόδοση εξαρτάται από τον τύπο των δεδομένων, την ποιότητα της προεπεξεργασίας και τον στόχο της εφαρμογής (π.χ. εξηγήσιμη πρόβλεψη vs. state-of-the-art ακρίβεια). Με βάση αυτά τα ευρήματα, η παρούσα εργασία επιλέγει να συγκρίνει παραδοσιακές μεθόδους (TF-IDF) και σύγχρονες αναπαραστάσεις (BERT embeddings), εφαρμόζοντάς τες σε ισχυρούς ταξινομητές όπως το XGBoost, ώστε να μετρηθεί η πραγματική υπεροχή κάθε προσέγγισης σε δεδομένα CVE/NVD.

Με βάση τον Πίνακα 2.1 παρατηρείται ότι η βιβλιογραφία έχει προσεγγίσει την πρόβλεψη CVSS είτε με παραδοσιακά μοντέλα μηχανικής μάθησης (TF-IDF με Decision Trees, Random Forests, XGBoost), είτε με σύγχρονα νευρωνικά δίκτυα (CNNs, BERT, LLMs). Οι πρώτες προσεγγίσεις πέτυχαν υψηλές επιδόσεις σε σχετικά μικρά ή ελεγχόμενα σύνολα δεδομένων, αλλά στερούνται γενίκευσης και παρουσιάζουν φαινόμενα overfitting. Οι πιο πρόσφατες εργασίες με BERT και LLMs δείχνουν βελτίωση στην ακρίβεια και στη δυνατότητα εξαγωγής σημασιολογικής πληροφορίας, όμως συνοδεύονται από σημαντικές απαιτήσεις σε υπολογιστικούς πόρους και δεν ενσωματώνουν πάντα σύγκριση με ισχυρά baseline όπως TF-IDF+XGBoost. Το επιστημονικό κενό που προκύπτει έγκειται στην ανάγκη για συστηματική, πειραματική αξιολόγηση διαφορετικών τεχνικών αναπαράστασης κειμένου (παραδοσιακών και σύγχρονων) σε συνδυασμό με ensemble μοντέλα, με έμφαση σε robust validation (π.χ. temporal split). Η παρούσα εργασία επιδιώκει να καλύψει αυτό το κενό, συγκρίνοντας TF-IDF και BERT embeddings σε συνδυασμό με XGBoost, ώστε να διερευνηθεί ποιος συνδυασμός επιτυγχάνει καλύτερη ισορροπία ακρίβειας, αποδοτικότητας και πρακτικής εφαρμοστικότητας.

Πίνακας 2.1:βιβλογραφικές προσέγγισης

Συγγραφείς (Ετος)	Dataset (CVE/NVD, έτη, N)	Χαρακτηριστικά (Features)	Μοντέλα	Μετρικές	Αποτελέσματα (κύρια)	Περιορισμοί
Khazaei et al. (2016) [11]	NVD, ~50k εγγραφές	TF-IDF	SVM, RF	Acc	≈70%	Εστίαση μόνο στο Base Score, χωρίς contextual embeddings
Elbaz et al. (2019) [12]	NVD, CVE descriptions	BoW, TF-IDF	LR	RMSE, Corr.	Μέτρια πρόβλεψη score	Χαμηλή ακρίβεια, χωρίς deep models
Zhang et al. (2020) [13]	NVD, ~70k	CVE text	CNN, Multi-CNN	F1, Acc	F1 ≈ 0.88	Υπολογιστικά δαπανηρό, χωρίς baseline TF-IDF
Aivatoglou et al. (2021) [9]	NVD, 2016–2020	Lemmatized text	DT, RF, XGB	Acc, F1	XGB ≈ 80%	Χωρίς deep learning για σύγκριση
Mandal & Kösesoy (2021) [18]	CVE Details	TF-IDF, Doc2Vec	kNN, DT, LR, NB, SVM	Acc	DT+TF-IDF ≈ 97%	Overfitting, περιορισμένο dataset
Costa et al. (2022) [8]	NVD 2016–20	CVE desc.	DistilBERT	Balanced Acc	Σημαντική βελτίωση με lemmatization	Δεν εφαρμόστηκε temporal split
Shahid & Debar (2022) [14]	NVD	BERT embeddings	CVSS-BERT	Acc, F1	Υψηλή ακρίβεια + εξηγησιμότητα	Υψηλό υπολογιστικό κόστος
Singh et al. (2022) [15]	CWE/SARD (code)	Source code metrics	CNN, BiLSTM, BERT	Acc, F1	≈95%	Δεν εστιάζει σε CVE text
Mandal & Kösesoy (2023) [10]	Apache Tomcat	Code-level metrics	kNN, DT, RF, NB	Acc	≈80%	Εφαρμογή σε μόνο ένα project
Tiwari (2025) [17]	NVD	CVE desc. + CVSS	BERT multi-task	Acc, F1	Near real-time API	Πολύπλοκο στην εκπαίδευση
Marchiori et al. (2025) [16]	NVD 2018–24	CVE text + embeddings	GPT-based LLMs	F1, CIA metrics	Βελτίωση σε Confidentiality, Integrity, Availability	Υψηλό κόστος, ανάγκη fine-tuning

2.5 Αναπαράσταση Κειμένου με Παραδοσιακές και Σύγχρονες Μεθόδους

Η αναπαράσταση κειμένου αποτελεί κρίσιμο στάδιο στην ανάπτυξη μοντέλων μηχανικής μάθησης για πρόβλεψη CVSS scores, καθώς καθορίζει ποια πληροφορία αποτυπώνεται ως χαρακτηριστικά εισόδου. Η βιβλιογραφία αναδεικνύει δύο κύριες κατηγορίες μεθόδων: τις παραδοσιακές στατιστικές αναπαραστάσεις όρων και τις σύγχρονες νευρωνικές προσεγγίσεις που αξιοποιούν προεκπαιδευμένα γλωσσικά μοντέλα.

2.5.1 Παραδοσιακές Μέθοδοι: Bag-of-Words και TF-IDF

Η μέθοδος Bag-of-Words (BoW) αναπαριστά κάθε έγγραφο ως ένα πολυσύνολο λέξεων, αγνοώντας σειρά, συμφραζόμενα και γραμματική. Αν και απλουστευμένη, αποτέλεσε το πρώτο βήμα σε μελέτες πρόβλεψης CVSS, προσφέροντας χαμηλό υπολογιστικό κόστος [12].

Η πιο διαδεδομένη παραδοσιακή μέθοδος είναι το TF–IDF (Term Frequency – Inverse Document Frequency), το οποίο συνδυάζει τη συχνότητα εμφάνισης ενός όρου σε συγκεκριμένο έγγραφο με τη σπανιότητά του στο corpus. Ο μαθηματικός ορισμός δίνεται από τον τύπο:

$$tfidf(t,d,D) = tf(t,d) \cdot \log\left(\frac{|D|}{|\{d' \in D \mid t \in d'\}|}\right)$$

όπου tf(t,d) είναι η συχνότητα του όρου t στο έγγραφο d, ενώ το κλάσμα υπολογίζει το αντίστροφο βάρος σπανιότητας στο σύνολο D.

Στην πράξη, το TF–IDF έχει χρησιμοποιηθεί εκτενώς σε εργασίες πρόβλεψης ευπαθειών. Οι Mandal και Kösesoy [18] κατέγραψαν ακρίβειες άνω του 95% όταν συνδυάστηκε με Decision Trees, καταδεικνύοντας τη σημασία του ως baseline. Ωστόσο, οι περιορισμοί του έγκεινται στην αδυναμία αποτύπωσης συμφραζομένων και σύνθετων τεχνικών σχέσεων, ειδικά σε σύντομες περιγραφές CVE.

2.5.2 Αναπαραστάσεις λέξεων: Word2Vec, fastText και Doc2Vec

Η ανάγκη για πλουσιότερη σημασιολογική αναπαράσταση οδήγησε σε πιο εξελιγμένες τεχνικές. Το Word2Vec δημιουργεί συνεχή διανύσματα για κάθε λέξη, αποτυπώνοντας σημασιολογικές εγγύτητες. Το fastText επεκτείνει την ιδέα ενσωματώνοντας υπολέξεις (subword embeddings), χρήσιμο για τεχνικούς όρους και νεολογισμούς. Το Doc2Vec, τέλος, παράγει αναπαραστάσεις σε επίπεδο εγγράφου.

Οι Mandal και Kösesoy [18] εφάρμοσαν Doc2Vec για ταξινόμηση CVEs, όμως τα αποτελέσματα δεν ξεπέρασαν το TF–IDF, καταδεικνύοντας ότι οι τεχνικές αυτές αποδίδουν καλύτερα σε μεγαλύτερα corpora με πλουσιότερα συμφραζόμενα.

2.5.3 Σύγχρονες μέθοδοι: BERT και Sentence-BERT

Η εισαγωγή του BERT [19] αποτέλεσε τομή στην κατανόηση φυσικής γλώσσας, προσφέροντας contextual embeddings που ενσωματώνουν συμφραζόμενα και σημασιολογικές σχέσεις. Το Sentence-BERT (SBERT) [20] επεκτείνει αυτή τη λογική σε επίπεδο προτάσεων, καθιστώντας το ιδανικό για ταξινομήσεις και semantic similarity.

Οι Shahid και Debar [14] με το CVSS-BERT έδειξαν ότι contextual embeddings υπερτερούν συστηματικά των παραδοσιακών αναπαραστάσεων, βελτιώνοντας την ακρίβεια πρόβλεψης και παρέχοντας δυνατότητα εξηγησιμότητας. Οι Costa et al. [8] απέδειξαν ότι preprocessing στάδια όπως το lemmatization ενισχύουν την απόδοση DistilBERT, αυξάνοντας τη balanced accuracy.

Η παρούσα εργασία επιλέγει το all-mpnet-base-v2 [21], ένα sentence transformer με υψηλή ακρίβεια και υπολογιστική αποδοτικότητα, το οποίο έχει αποδειχθεί ιδανικό για τεχνικά και μικτά κείμενα.

2.5.4 CNN-based feature extraction

Πέρα από τις αναπαραστάσεις λέξεων και προτάσεων, ορισμένες μελέτες έχουν χρησιμοποιήσει CNNs για εξαγωγή χαρακτηριστικών απευθείας από ακολουθίες κειμένου. Οι Zhang et al. [13] πρότειναν Multi-CNN αρχιτεκτονική, η οποία ανίχνευσε τεχνικά μοτίβα και n-grams σε CVE περιγραφές, πετυχαίνοντας F1 score ≈ 0.88, καλύτερο από αρκετές παραδοσιακές μεθόδους.

2.5.5 LLM-based embeddings

Οι πιο πρόσφατες εξελίξεις περιλαμβάνουν τη χρήση μεγάλων γλωσσικών μοντέλων (LLMs). Οι Marchiori et al. [16] έδειξαν ότι GPT-based embeddings μπορούν να προβλέψουν CVSS vectors με βελτιωμένη ακρίβεια, ειδικά στις διαστάσεις Confidentiality, Integrity και Availability. Ωστόσο, οι απαιτήσεις σε υπολογιστικούς πόρους και τα ζητήματα επαναληψιμότητας παραμένουν σημαντικά εμπόδια για την πρακτική υιοθέτηση.

2.5.6 Συμπέρασμα

Η συγκριτική ανάλυση δείχνει ότι:

- Τα TF–IDF και BoW παραμένουν ισχυρά baseline λόγω απλότητας και χαμηλού κόστους.
- Τα Word2Vec, fastText και Doc2Vec βελτιώνουν τη σημασιολογική πληροφορία, αλλά η απόδοσή τους εξαρτάται από το μέγεθος και την ποικιλία του corpus.
- Τα BERT/SBERT embeddings παρέχουν state-of-the-art αποτελέσματα, ειδικά σε περιγραφές CVE.
- CNN-based προσεγγίσεις μπορούν να αναδείξουν τοπικά τεχνικά μοτίβα.
- LLM embeddings αποτελούν την αιχμή της έρευνας, αλλά με σημαντικό κόστος και προκλήσεις επαναληψιμότητας.

Η παρούσα εργασία εστιάζει στη σύγκριση TF–IDF και contextual embeddings (SBERT all-mpnet-base-v2), αξιοποιώντας το XGBoost ως ταξινομητή. Η επιλογή αυτή βασίζεται τόσο στην αποδοτικότητα όσο και στην ανάγκη κάλυψης του ερευνητικού κενού που προκύπτει από τη βιβλιογραφία σχετικά με υβριδικές προσεγγίσεις.

2.6 Μοντέλα Μηχανικής Μάθησης για Προβλήματα Ταξινόμησης

Σε ένα πρόβλημα ταξινόμησης (classification), ο στόχος είναι να προβλεφθεί ποια κατηγορία ή ετικέτα ανήκει σε ένα δεδομένο δείγμα εισόδου όπως για παράδειγμα, την ταξινόμηση μιας ευπάθειας ως “High” ή “Critical” βάσει χαρακτηριστικών που προκύπτουν από το κείμενο και τις μετρικές CVSS. Αυτά τα μοντέλα μαθαίνουν να συσχετίζουν τα χαρακτηριστικά (features) κάθε δείγματος με την αντίστοιχη κλάση μέσω εκπαίδευσης πάνω σε επισημασμένα δεδομένα (labeled data), και στη συνέχεια εφαρμόζονται σε μη ορατά νέα δεδομένα για παραγωγή προβλέψεων.

Όπως περιγράφεται στο βιβλίο Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (Géron, 2019), η ταξινόμηση αποτελεί βασικό εργαλείο στην εποπτευόμενη μάθηση (supervised

learning) και εμπεριέχει ευρύ φάσμα αλγορίθμων από γραμμικές μεθόδους έως σύνθετες αρχιτεκτονικές deep learning , για να εξυπηρετεί διαφορετικά είδη προβλημάτων και δεδομένων .

Ακολουθεί συνοπτική επισκόπηση των κύριων αλγορίθμων ταξινόμησης:

- Logistic Regression: Μοντέλο γραμμικής ταξινόμησης που χρησιμοποιεί τη λογιστική συνάρτηση για προβλεπόμενη πιθανότητα.
- Naive Bayes: Πιθανοτικό μοντέλο βασισμένο στον κανόνα του Bayes με υπόθεση ανεξαρτησίας χαρακτηριστικών .
- k-Nearest Neighbors (k-NN): Κατηγοριοποίηση βάσει των πιο κοντινών γειτονικών δειγμάτων.
- Support Vector Machines (SVMs): Βρίσκουν το βέλτιστο υπερεπίπεδο που διαχωρίζει τις κλάσεις με μέγιστο περιθώριο.
- Decision Trees: Δοκιμάζονται ήδη προηγούμενα στο thesis για βάση και εξηγήσιμη επεξήγηση.
- Ensemble Methods: Όπως Random Forest (bagging πολλαπλών δέντρων) και Boosting (διαδοχικά δέντρα) – βελτιώνουν την ακρίβεια και γενίκευση.
- Νευρωνικά Δίκτυα (MLP, CNNs, Transformers): Αποτελούν state-of-the-art επιλογές σε πολύπλοκα και μεγάλα δεδομένα, ιδιαίτερα για κείμενο και εικόνα.

Στα συγγράμματα των Hastie, Tibshirani & Friedman (The Elements of Statistical Learning), Bishop (Pattern Recognition and Machine Learning) και Murphy (Machine Learning: A Probabilistic Perspective), παρατίθενται αναλυτικές θεωρητικές βάσεις και εφαρμογές για όλα τα παραπάνω μοντέλα, καλύπτοντας τόσο τη στατιστική θεωρία όσο και πρακτικά παραδείγματα.

2.6.1 Δέντρα Απόφασης (Decision Trees)

Τα Δέντρα Απόφασης αποτελούν μία από τις πιο ευρέως χρησιμοποιούμενες και κατανοητές μεθόδους στη μηχανική μάθηση, καθώς εφαρμόζονται τόσο σε προβλήματα ταξινόμησης όσο και παλινδρόμησης. Η βασική τους αρχή έγκειται στη διαδοχική διάσπαση ενός συνόλου δεδομένων σε μικρότερα υποσύνολα, με γνώμονα χαρακτηριστικά που μεγιστοποιούν την καθαρότητα των κόμβων. Η δομή τους περιλαμβάνει κόμβους απόφασης (decision nodes), κλαδιά (branches) και φύλλα (leaves), όπου τα φύλλα αντιπροσωπεύουν τις τελικές κλάσεις ή τιμές εξόδου. Η ερμηνευσιμότητα αυτής της διαδικασίας, καθώς και η δυνατότητα απεικόνισης της λογικής λήψης αποφάσεων, καθιστούν τα δέντρα ιδιαίτερα ελκυστικά σε τομείς όπου απαιτείται διαφάνεια και εξηγήσιμη τεχνητή νοημοσύνη, όπως η κυβερνοασφάλεια [16], [18].

Η επιλογή του βέλτιστου χαρακτηριστικού σε κάθε κόμβο βασίζεται σε μετρικές καθαρότητας. Η πιο διαδεδομένη είναι η εντροπία, η οποία ορίζεται ως:

$$H(S) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

όπου p_i είναι η πιθανότητα ένα δείγμα να ανήκει στην κλάση i. Με βάση αυτήν, υπολογίζεται το πληροφοριακό κέρδος (Information Gain), το οποίο μετρά τη μείωση της αβεβαιότητας που επιτυγχάνεται έπειτα από έναν διαχωρισμό:

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

Εναλλακτικά, χρησιμοποιείται ο δείκτης Gini Impurity, ο οποίος υπολογίζει την πιθανότητα εσφαλμένης ταξινόμησης αν η κλάση ενός δείγματος επιλεγεί τυχαία:

$$Gini(S) = 1 - \sum_{i=1}^{k} p_i^2$$

Οι μετρικές αυτές εφαρμόζονται επαναληπτικά μέχρι το δέντρο να φτάσει σε ικανοποιητικό βάθος ή να επιτευχθεί πλήρης καθαρότητα στους κόμβους τερματισμού. Για την αποφυγή υπερπροσαρμογής (overfitting), συνήθως εφαρμόζονται τεχνικές pruning, οι οποίες περιορίζουν το μέγιστο βάθος ή απαιτούν ελάχιστο αριθμό δειγμάτων ανά φύλλο [16], [19].

Ένα από τα βασικά πλεονεκτήματα των δέντρων απόφασης είναι η υψηλή τους εξηγησιμότητα καθώς τα αποτελέσματά τους μπορούν να γίνουν εύκολα κατανοητά από ανθρώπους και να απεικονιστούν σε μορφή διαγράμματος. Είναι επίσης ιδιαίτερα ευέλικτα, καθώς μπορούν να χρησιμοποιηθούν με κατηγορικά και αριθμητικά δεδομένα, ενώ προσφέρουν αυτόματη επιλογή των πιο κρίσιμων χαρακτηριστικών. Παράλληλα παρουσιάζουν και σημαντικές αδυναμίες. Συγκεκριμένα, έχουν ισχυρή τάση υπερπροσαρμογής, ιδίως όταν εφαρμόζονται σε μεγάλα ή θορυβώδη σύνολα δεδομένων, και είναι ασταθή, αφού μικρές αλλαγές στο σύνολο εκπαίδευσης μπορούν να οδηγήσουν σε πολύ διαφορετικά δέντρα. Επιπλέον, υστερούν σε ακρίβεια σε σχέση με πιο εξελιγμένες ensemble μεθόδους, όπως τα Random Forests και το XGBoost [18], [20].

Στον τομέα της κυβερνοασφάλειας, τα Δέντρα Απόφασης έχουν βρει εφαρμογή στην αυτόματη κατηγοριοποίηση ευπαθειών. Η μελέτη των Mandal και Kösesoy [21], η οποία εφάρμοσε και συνέκρινε διαφορετικούς αλγορίθμους (k-NN, Decision Trees, Logistic Regression, Naive Bayes, Random Forest, SVM) σε συνδυασμό με τεχνικές εξαγωγής χαρακτηριστικών όπως το TF-IDF και το Doc2Vec, κατέληξε στο συμπέρασμα ότι τα Decision Trees, σε συνδυασμό με το TF-IDF, προσέφεραν την υψηλότερη απόδοση, με ακρίβεια που έφτασε το 97%, αναδεικνύοντας τη συμβατότητα των μοντέλων βασισμένων σε δέντρα με αραιές και υψηλής διαστατικότητας αναπαραστάσεις..

2.6.2 Ensemble Methods: Random Forests και Boosting

Οι μέθοδοι συνόλων (ensemble methods) συνιστούν προσεγγίσεις της μηχανικής μάθησης που συνδυάζουν πολλαπλούς αδύναμους ταξινομητές με στόχο τη βελτίωση της ακρίβειας, της σταθερότητας και της γενίκευσης. Δύο από τις πιο διαδεδομένες κατηγορίες ensemble είναι το bagging και το boosting, με χαρακτηριστικούς εκπροσώπους τα Random Forests και το XGBoost αντίστοιχα.

Τα Random Forests στηρίζονται στην ιδέα της τυχαίας δειγματοληψίας (bagging). Συγκεκριμένα, δημιουργούν πολλά ανεξάρτητα δέντρα απόφασης σε τυχαία δείγματα του συνόλου εκπαίδευσης και συνδυάζουν τα αποτελέσματά τους μέσω ψηφοφορίας (για classification) ή μέσου όρου (για regression). Η τυχαιότητα αυτή μειώνει το variance και καθιστά το μοντέλο πιο ανθεκτικό σε overfitting, διατηρώντας υψηλή γενίκευση [16], [18].

Αντίθετα, το Boosting αποτελεί μια διαδοχική διαδικασία όπου κάθε νέο δέντρο εκπαιδεύεται ώστε να διορθώσει τα λάθη των προηγούμενων. Η κεντρική ιδέα είναι η εστίαση σε δύσκολες περιπτώσεις εκπαίδευσης, αυξάνοντας σταδιακά το βάρος τους, ώστε τα επόμενα μοντέλα να βελτιώνουν την απόδοση. Το XGBoost (Extreme Gradient Boosting) είναι μια βελτιστοποιημένη παραλλαγή του gradient boosting, η οποία ενσωματώνει regularization, παράλληλη εκπαίδευση, χειρισμό ελλιπών δεδομένων και άλλες βελτιώσεις που το καθιστούν εξαιρετικά αποδοτικό σε δομημένα δεδομένα [16], [20].

Η σημασία αυτών των μεθόδων έχει αναδειχθεί σε ερευνητικές εφαρμογές κυβερνοασφάλειας. Στη μελέτη των Aivatoglou et al. [15], η οποία εφάρμοσε tree-based μεθοδολογία για αυτόματη ταξινόμηση ευπαθειών, συγκρίθηκαν τα Decision Trees, τα Random Forests και το XGBoost χρησιμοποιώντας

δεδομένα από το NVD. Τα αποτελέσματα έδειξαν ότι το XGBoost ξεπέρασε συστηματικά τους άλλους δύο αλγορίθμους σε όλους τους δείκτες απόδοσης, συμπεριλαμβανομένων της ακρίβειας (Accuracy), της ακρίβειας προβλέψεων (Precision), της ευαισθησίας (Recall) και του F1-score [15]. Συγκεκριμένα, ενώ τα Decision Trees εμφάνισαν ακρίβεια γύρω στο 66% και τα Random Forests περίπου 76%, το XGBoost κατέγραψε τις υψηλότερες επιδόσεις σε κάθε μετρική, επιβεβαιώνοντας την ανωτερότητά του σε πολύπλοκα και ανισόρροπα σύνολα δεδομένων.

Με βάση τα παραπάνω, η παρούσα εργασία επιλέγει το XGBoost ως κύριο αλγόριθμο για την πρόβλεψη κινδύνου ευπαθειών. Η επιλογή αυτή δικαιολογείται τόσο από τα πλεονεκτήματα του μοντέλου σε επίπεδο θεωρίας (regularization, αποδοτικότητα, υψηλή ακρίβεια) όσο και από τα εμπειρικά αποτελέσματα της σχετικής βιβλιογραφίας, τα οποία καταδεικνύουν την υπεροχή του έναντι άλλων δέντρων και ensemble μεθόδων.

2.6.3 Ο Αλγόριθμος XGBoost

Το **XGBoost (Extreme Gradient Boosting)** είναι βελτιστοποιημένη υλοποίηση του gradient boosting με δέντρα απόφασης ως «ασθενείς» μαθητές. Στόχος του είναι να κατασκευάσει ένα **αθροιστικό μοντέλο** από M δέντρα

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i), \qquad f_t \in \mathcal{F},$$

όπου $\mathcal{F}$ το σύνολο των δέντρων παλινδρόμησης/ταξινόμησης, ώστε κάθε νέο δέντρο f_1 να διορθώνει τα σφάλματα των προηγουμένων [18], [22].

Αντικειμενική συνάρτηση και κανονικοποίηση

Στον βηματικό χρόνο t, η αντικειμενική συνάρτηση λαμβάνει τη μορφή

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t),$$

όπου $l(.,.)$ είναι η συνάρτηση απώλειας (π.χ. λογιστική απώλεια για δυαδική ταξινόμηση) και $\Omega(f)$ όρος κανονικοποίησης που ελέγχει την πολυπλοκότητα του δέντρου:

$$\Omega(f) = \Upsilon \mathrm{T} + \frac{1}{2}\lambda \parallel w \parallel^2$$

με T τον αριθμό φύλλων και $w \epsilon R^T$ τα βάρη (scores) των φύλλων. Η ποινή γ αποτρέπει υπερβολικό «θρυμματισμό» του δέντρου (πολλά φύλλα), ενώ η λ (L2) περιορίζει τα μεγέθη των leaf weights, βελτιώνοντας τη γενίκευση [22].

Δεύτερης τάξης προσεγγίσεις (Taylor)

Το XGBoost χρησιμοποιεί ανάπτυγμα Taylor 2ης τάξης γύρω από $\hat{y}i^{(t-i)}$:

$$\mathcal{L}^{(t)} \approx \sum i = 1^n \left[g_i f_i(x_i) + \frac{1}{2} h_i f_i(x_i)^2\right] + \Omega(f_t)$$

Όπου $\mathrm{g_i} = \partial_{\hat{y}} \ell(\mathrm{y_i}, \hat{\mathrm{y}}\mathrm{i}^{(\mathrm{t}-1)})$ και $\mathrm{h_i} = \partial^2 \hat{y} \ell(\mathrm{y_i}, \hat{\mathrm{y}}\mathrm{i}^{(\mathrm{t}-1)})$ είναι gradient και Hessian αντίστοιχα. Αν το δέντρο $\mathrm{f_t}$ αναθέτει κάθε δείγμα σε ένα φύλλο j με βάρος , και αν $\mathrm{I_j}$ είναι τα δείγματα του φύλλου, τότε ο προσεγγιστικός στόχος γράφεται ως άθροισμα κατά φύλλο:

$$\tilde{\mathcal{L}}^{(t)} = \sum j = 1^T \left(G_j w_j + \frac{1}{2}\left(H_j + \lambda\right) w_j^2\right) + \gamma \mathrm{T}$$

με $G_j = \sum i \epsilon I_j{}^{gi}$ και $H_j = \sum i \epsilon I_j{}^{hi}$. Η βέλτιστη τιμή φύλλου είναι

$w_j^* = -\frac{G_j}{H_j+\lambda}$,

και η «ποιότητα» (score) του δέντρου προκύπτει κλειστά:

$$\tilde{\mathcal{L}}^{(t)}(w^*) = -\frac{1}{2}\sum_{j=1}^{\mathrm{T}} \frac{G_j^2}{H_j+\lambda} + \gamma \mathrm{T}$$

Για έναν πιθανό διαχωρισμό κόμβου σε αριστερό/δεξί παιδί, το **gain** δίνεται από

$$Gain = \frac{1}{2}\left(\frac{G_L^2}{H_l+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right) - \gamma$$

όπου τα υποσύμβολα L, R αναφέρονται στα δύο υποσύνολα. Ο αλγόριθμος επιλέγει τον διαχωρισμό με το μέγιστο θετικό gain [22].

Καταλληλότητα για το παρόν πρόβλημα

Στο πρόβλημα πρόβλεψης κινδύνου με βάση περιγραφές CVE και μεταβλητές CVSS, το XGBoost είναι ιδιαιτέρως κατάλληλο διότι: (α) μετατρέπει σε μοντέλα μη γραμμικότητες και αλληλεπιδράσεις μεταξύ χαρακτηριστικών, (β) διαχειρίζεται αραιές και υψηλής διαστατικότητας αναπαραστάσεις (π.χ. TF-IDF, embeddings) χάρη στον sparsity-aware πυρήνα του, (γ) προσφέρει σταθερή γενίκευση μέσω ισχυρής κανονικοποίησης και δειγματοληψίας, και (δ) αποδίδει υψηλά σε πρακτικές μετρικές (Accuracy, Precision, Recall, F1), όπως έχει καταγραφεί σε συγκρίσεις με Decision Trees και Random Forests στη βιβλιογραφία ευπαθειών [15], [16].

2.7 Σχετική Έρευνα και Συγκριτικά Μοντέλα

Η διεθνής βιβλιογραφία για την αυτόματη πρόβλεψη μεταβλητών CVSS και τη γενικότερη εκτίμηση κινδύνου από δεδομένα CVE ακολουθεί δύο βασικές κατευθύνσεις. Η πρώτη αξιοποιεί δομημένα χαρακτηριστικά, όπως οι μεταβλητές του CVSS ή τα Metadata της NVD, σε συνδυασμό με κλασικούς αλγορίθμους μηχανικής μάθησης. Η δεύτερη αντλεί πληροφορία απευθείας από τις περιγραφές ευπαθειών σε φυσική γλώσσα, εφαρμόζοντας τεχνικές επεξεργασίας κειμένου και νευρωνικά embeddings για ταξινόμηση

Στην πρώτη κατεύθυνση, οι αλγόριθμοι βασισμένοι σε δέντρα και τα ensemble μοντέλα έχουν καταγράψει υψηλές επιδόσεις, καθώς συλλαμβάνουν μη γραμμικότητες και αλληλεπιδράσεις χαρακτηριστικών. Οι Aivatoglou et al. [9] ανέπτυξαν μια tree-based μεθοδολογία για αυτόματη κατηγοριοποίηση ευπαθειών και συνέκριναν Decision Trees, Random Forests και XGBoost σε δεδομένα από το NVD. Τα αποτελέσματά τους έδειξαν ότι το XGBoost υπερείχε σταθερά σε Accuracy, Precision, Recall και F1-score, επιβεβαιώνοντας την υπεροχή του boosting έναντι των bagging και μονοδενδρικών προσεγγίσεων. Συμπληρωματικά, οι Mandal και Kösesoy [18] μελέτησαν την πρόβλεψη ευπαθειών από χαρακτηριστικά πηγαίου κώδικα και αναπαραστάσεις κειμένου με TF-IDF και Doc2Vec, δοκιμάζοντας κλασικούς αλγορίθμους ταξινόμησης όπως k-NN, Decision Trees, Logistic Regression, Naive Bayes, Random Forest και SVM. Διαπίστωσαν ότι ο συνδυασμός Decision Trees με TF-IDF απέδωσε την υψηλότερη ακρίβεια (97%), αναδεικνύοντας τη συμβατότητα των μοντέλων βασισμένων σε αλγορίθμων δέντρων με αραιές και υψηλής διαστατικότητας αναπαραστάσεις

Η δεύτερη κατεύθυνση επικεντρώνεται στην αξιοποίηση των περιγραφών CVE με τεχνικές NLP. Οι Costa et al. [8] διερεύνησαν την πρόβλεψη μεταβλητών CVSS από περιγραφές ευπαθειών και

κατέδειξαν ότι τεχνικές preprocessing όπως το lemmatization και σε μικρότερο βαθμό το stemming βελτιώνουν σημαντικά την balanced accuracy, ακόμη και σε μοντέλα τύπου DistilBERT. Αντίστοιχα, οι Elbaz et al. [12] παρουσίασαν μια αυτόματη μέθοδο πρόβλεψης CVSS score βασισμένη σε περιγραφές ευπαθειών, επιτυγχάνοντας βελτιώσεις έναντι παραδοσιακών μεθόδων. Ο Zhang et al. [13] εφάρμοσαν Multi-CNN για την πρόβλεψη base metrics του CVSS, δείχνοντας ότι οι συνελικτικές αρχιτεκτονικές μπορούν να αποδώσουν καλύτερα σε μεγάλα και σύνθετα corpora. Επιπλέον, οι Shahid και Debar [14] πρότειναν το CVSS-BERT, ένα εξηγήσιμο NLP-based μοντέλο που ανέλυε τις περιγραφές ευπαθειών για να προσδιορίσει τη σοβαρότητα, καταγράφοντας σημαντική βελτίωση έναντι των παραδοσιακών TF-IDF. Παράλληλα, οι Singh et al. [15] έδειξαν ότι ακόμη και βασικές τεχνικές NLP μπορούν να αξιοποιηθούν για την ανίχνευση ευπαθειών, αναδεικνύοντας την πρακτική σημασία της γλωσσικής επεξεργασίας

Οι δύο κατευθύνσεις συγκλίνουν σε υβριδικές προσεγγίσεις όπου δομημένα χαρακτηριστικά (μεταβλητές CVSS, χρονικά ή τεχνικά μεταδεδομένα) συνδυάζονται με αναπαραστάσεις κειμένου (TF-IDF ή contextual embeddings) και τροφοδοτούν ισχυρούς ταξινομητές όπως το XGBoost. Η σύζευξη αυτή καταγράφει σταθερά κέρδη σε όλες τις κλασικές μετρικές (Accuracy, Precision, Recall, F1), σε σχέση με μοντέλα που χρησιμοποιούν μόνο μία πηγή πληροφορίας [8], [9], [14]

Πρόσφατες εξελίξεις ενισχύουν περαιτέρω την εικόνα. Ο Tiwari [17] πρότεινε έναν BERT-based multi-objective ταξινομητή που προβλέπει ταυτόχρονα σοβαρότητα και τύπο ευπάθειας, παρέχοντας REST API και UI για real-time χρήση. Οι Marchiori et al. [16] μελέτησαν τη χρήση μεγάλων γλωσσικών μοντέλων (LLMs) για την παραγωγή CVSS vectors απευθείας από περιγραφές CVE, δείχνοντας ότι η ενσωμάτωση LLMs με embedding-based προσεγγίσεις βελτιώνει την ακρίβεια, ιδίως στις κατηγορίες Confidentiality, Integrity και Availability. Τέλος, η συνεχιζόμενη κριτική προς το CVSS v2 και v3 και οι βελτιώσεις της έκδοσης v4.0 (Threat και Supplemental metrics) [6], [7] ενισχύουν το ερευνητικό κίνητρο για μεθόδους πρόβλεψης που αξιοποιούν πλήρως το κείμενο και τα δομημένα πεδία

Με βάση τα παραπάνω, η παρούσα εργασία υιοθετεί μια υβριδική προσέγγιση που συνδυάζει παραδοσιακές και σύγχρονες αναπαραστάσεις κειμένου (TF-IDF και BERT-based embeddings) με XGBoost ως κύριο ταξινομητή, επιλογή που εδράζεται τόσο στη θεωρητική τεκμηρίωση του boosting όσο και στην εμπειρική υπεροχή του σε συγκριτικές μελέτες ευπαθειών

2.8 Συμπεράσματα Βιβλιογραφικής Ανασκόπησης

Στο παρόν κεφάλαιο παρουσιάστηκαν τα θεμελιώδη στοιχεία και η σχετική ερευνητική πρόοδος στον τομέα της αυτόματης πρόβλεψης κινδύνου ευπαθειών. Αρχικά εξετάστηκαν οι βάσεις δεδομένων CVE και NVD, οι οποίες συνιστούν το διεθνές πλαίσιο αναφοράς για την τυποποιημένη καταγραφή και τεκμηρίωση ευπαθειών. Ακολούθως αναλύθηκε το CVSS, ως το καθιερωμένο πρότυπο ποσοτικοποίησης της σοβαρότητας, αναδεικνύοντας τη χρησιμότητά του στην ιεράρχηση διορθώσεων αλλά και τους περιορισμούς του, ιδίως στις εκδόσεις v2 και v3, γεγονός που οδήγησε σε βελτιώσεις στη v4.0

Παράλληλα δόθηκε έμφαση στις περιγραφές CVE ως πολύτιμη πηγή πληροφορίας που συχνά δεν αποτυπώνεται πλήρως στα τυποποιημένα πεδία του CVSS. Η ανάλυση των τεχνικών NLP κατέδειξε ότι στάδια όπως το tokenization, το stop-word removal και ιδίως η lemmatization βελτιώνουν την ποιότητα των δεδομένων και ενισχύουν την απόδοση των μοντέλων. Σε επίπεδο αναπαράστασης κειμένου παρουσιάστηκαν τόσο οι παραδοσιακές μέθοδοι (BoW, TF-IDF) όσο και οι σύγχρονες προσεγγίσεις με contextual embeddings (BERT, SBERT), αναδεικνύοντας τη συμπληρωματική τους αξία

Σε ό,τι αφορά τα μοντέλα μηχανικής μάθησης, αναλύθηκαν οι κλασικοί αλγόριθμοι ταξινόμησης με ιδιαίτερη έμφαση στα Δέντρα Απόφασης και στις ensemble μεθόδους (Random Forests, Gradient Boosting). Η βιβλιογραφία τεκμηριώνει ότι τα Decision Trees αποτελούν ερμηνεύσιμο baseline, ενώ τα μοντέλα boosting και ειδικά το XGBoost επιτυγχάνουν ανώτερες επιδόσεις σε Accuracy, Precision, Recall και F1-score, ιδίως όταν εφαρμόζονται σε υψηλής διαστατικότητας δεδομένα όπως οι περιγραφές CVE. Επιπρόσθετα, ερευνητικές εργασίες έχουν αναδείξει την αξία νευρωνικών αρχιτεκτονικών όπως τα Multi-CNN για πρόβλεψη CVSS base metrics, καθώς και εξειδικευμένων προσεγγίσεων όπως το CVSS-BERT, που αξιοποιούν εξηγήσιμα embeddings. Οι πιο πρόσφατες εξελίξεις με LLM-based μεθόδους (π.χ. Tiwari 2025, Marchiori et al. 2025) καταδεικνύουν ότι η ενσωμάτωση μεγάλων γλωσσικών μοντέλων μπορεί να βελτιώσει την ακρίβεια ακόμη και σε δύσκολες κατηγορίες όπως Confidentiality, Integrity και Availability

Η σύγκριση σχετικών ερευνών καταδεικνύει ότι τα καλύτερα αποτελέσματα προκύπτουν όταν συνδυάζονται δομημένα χαρακτηριστικά (μεταβλητές CVSS, χρονικά ή τεχνικά μεταδεδομένα) με αναπαραστάσεις κειμένου (TF-IDF, embeddings), και όταν αξιοποιούνται ισχυροί ταξινομητές όπως το XGBoost. Το επιστημονικό κενό που προκύπτει αφορά τη συστηματική εμπειρική αξιολόγηση διαφορετικών τεχνικών αναπαράστασης (παραδοσιακές έναντι σύγχρονων) σε συνδυασμό με ensemble μοντέλα, τόσο για την πρόβλεψη επιμέρους μεταβλητών CVSS όσο και για την εκτίμηση του συνολικού κινδύνου

Η παρούσα εργασία επιδιώκει να καλύψει αυτό το κενό, συγκρίνοντας εμπειρικά την απόδοση των TF-IDF, BERT embeddings και του συνδυασμού τους, εφαρμόζοντάς τα σε XGBoost για την πρόβλεψη κρίσιμων μεταβλητών του CVSS. Η ανάλυση αυτή φιλοδοξεί να προσφέρει μια πιο ολοκληρωμένη και ακριβή προσέγγιση στην αυτοματοποιημένη πρόβλεψη κινδύνου, συμβάλλοντας τόσο στη θεωρητική πρόοδο της βιβλιογραφίας όσο και στην πρακτική εφαρμογή σε πραγματικά σενάρια κυβερνοασφάλειας

Κεφάλαιο 3ο: Μεθοδολογία

3.1 Εισαγωγή

Η μεθοδολογία αποτελεί τον πυρήνα κάθε επιστημονικής έρευνας, καθώς καθορίζει με σαφήνεια τα βήματα που ακολουθήθηκαν για τη συλλογή, την προετοιμασία και την ανάλυση των δεδομένων, καθώς και για την ανάπτυξη και αξιολόγηση των μοντέλων που προτείνονται. Στην παρούσα εργασία, η μεθοδολογική διαδικασία σχεδιάστηκε με σκοπό να απαντήσει σε ένα διπλό ερευνητικό ερώτημα: αρχικά το ποια είναι η βέλτιστη μέθοδος αναπαράστασης κειμένου για την πρόβλεψη μεταβλητών του CVSS και έπειτα το πώς μπορεί να αξιοποιηθεί ένας αλγόριθμος ταξινόμησης όπως ο XGBoost ώστε να επιτευχθεί υψηλή ακρίβεια και γενίκευση σε δεδομένα με έντονη ανισορροπία και ετερογένεια.

Η διαδικασία περιλάμβανε διαδοχικά στάδια, ξεκινώντας από τη συλλογή και την οργάνωση ενός εκτενούς συνόλου δεδομένων CVE μέσω του API της National Vulnerability Database (NVD). Στη συνέχεια, το dataset επικεντρώθηκε αποκλειστικά σε καταχωρήσεις που συνοδεύονται από βαθμολογήσεις CVSS v3.1, καθώς αυτή η έκδοση θεωρείται πλέον το επικρατέστερο και πιο αξιόπιστο πρότυπο αξιολόγησης. Ακολούθησε η δημιουργία ενός αντιπροσωπευτικού και καθαρού συνόλου δεδομένων, το οποίο αποτέλεσε τη βάση για την εκπαίδευση και την αξιολόγηση των μοντέλων.

Σε επόμενο στάδιο εφαρμόστηκαν τεχνικές αναπαράστασης κειμένου με δύο διαφορετικές λογικές: την παραδοσιακή, μέσω TF-IDF, η οποία βασίζεται στη στατιστική συχνότητα των όρων, και τη σύγχρονη, μέσω Sentence-BERT (SBERT), η οποία αξιοποιεί προεκπαιδευμένα νευρωνικά δίκτυα Transformer για τη δημιουργία σημασιολογικών αναπαραστάσεων. Οι δύο αυτές προσεγγίσεις εξετάστηκαν τόσο αυτόνομα όσο και σε συνδυασμό, ώστε να αξιολογηθεί αν η στατιστική και η σημασιολογική πληροφορία είναι συμπληρωματικές.

Η μεθοδολογική διαδικασία δεν περιορίστηκε στην επιλογή αναπαραστάσεων, αλλά έλαβε υπόψη και την ανάγκη για σωστή κωδικοποίηση των κατηγορικών labels του CVSS, την αντιμετώπιση της ανισορροπίας των τάξεων μέσω στρωματοποιημένης δειγματοληψίας και βαρών εκπαίδευσης, καθώς και τη διαμόρφωση ενός αξιόπιστου πλαισίου αξιολόγησης με χρήση μετρικών όπως Accuracy, Precision, Recall και Macro-F1.

Τέλος, σχεδιάστηκε ένα σύνολο έξι πειραμάτων που εξετάζουν διαδοχικά την επίδραση διαφορετικών επιλογών σε κάθε στάδιο, με στόχο να εξαχθούν ασφαλή και συγκρίσιμα συμπεράσματα. Μέσα από αυτή τη διαδικασία, η μεθοδολογία λειτουργεί ως το συνδετικό στοιχείο ανάμεσα στο θεωρητικό υπόβαθρο που παρουσιάστηκε στο Κεφάλαιο 2 και στα αποτελέσματα και τη συζήτησή τους που ακολουθούν στα Κεφάλαια 4 και 5.

3.2 Ερευνητική Προσέγγιση

Η ερευνητική προσέγγιση της παρούσας μελέτης ακολούθησε ένα διττό σχήμα, το οποίο συνδυάζει τη διερευνητική (exploratory) και τη συνθετική (synthetic) στρατηγική.

Στο διερευνητικό στάδιο πραγματοποιήθηκε εκτενής ανασκόπηση της βιβλιογραφίας με στόχο την αποτύπωση των κυρίαρχων μεθόδων στην πρόβλεψη CVSS βαθμολογιών και γενικότερα στην ταξινόμηση ευπαθειών. Η ανασκόπηση ανέδειξε δύο βασικές γραμμές ερευνητικής κατεύθυνσης: αφενός τη χρήση δομημένων χαρακτηριστικών του NVD (π.χ. attack vector, complexity, privileges required) με κλασικούς ταξινομητές, και αφετέρου την αξιοποίηση περιγραφών CVE με τεχνικές επεξεργασίας φυσικής γλώσσας [8], [9], [13], [14]. Η πρώτη γραμμή επέτρεψε σε tree-based

αλγορίθμους όπως Random Forests και XGBoost να καταγράψουν υψηλές επιδόσεις σε ακρίβεια και F1-score [9], ενώ η δεύτερη ανέδειξε τη σημασία του NLP pipeline. Ενδεικτικά, οι Costa et al. [8] έδειξαν ότι βήματα όπως το lemmatization αυξάνουν την balanced accuracy σε DistilBERT μοντέλα, ενώ οι Aivatoglou et al. [9] κατέδειξαν την καταλληλότητα των Decision Trees και XGBoost σε συνδυασμό με TF-IDF για ταξινόμηση ευπαθειών.

Επιπλέον, οι πιο πρόσφατες μελέτες (Tiwari, 2025· Marchiori et al., 2025) ανέδειξαν την ταχεία μετατόπιση προς BERT-based και LLM-based μοντέλα, τα οποία μπορούν να συλλάβουν σημασιολογικές σχέσεις που ξεπερνούν τις απλές μετρήσεις συχνότητας. Τα αποτελέσματά τους δείχνουν ότι η χρήση contextual embeddings βελτιώνει σημαντικά την πρόβλεψη ειδικά σε μεταβλητές που σχετίζονται με επιπτώσεις σε Confidentiality, Integrity και Availability [16], [17].

Με βάση αυτά τα ευρήματα, η παρούσα εργασία πέρασε στη συνθετική φάση, όπου συγκροτήθηκε ένα ενιαίο πειραματικό πλαίσιο. Σε αυτό ενσωματώθηκαν:

- Παραδοσιακές αναπαραστάσεις κειμένου (TF-IDF),
- Σύγχρονες νευρωνικές αναπαραστάσεις (all-mpnet-base-v2 embeddings),
- Και ο συνδυασμός τους, ώστε να εξεταστεί αν η παράλληλη αξιοποίηση στατιστικής και σημασιολογικής πληροφορίας βελτιώνει την απόδοση.

Ο XGBoost επιλέχθηκε ως σταθερό baseline αλγόριθμος, δικαιολογημένα από την υπεροχή του έναντι Decision Trees και Random Forests σε προηγούμενες μελέτες [9], [13]. Η χρήση του ως «κοινό πυρήνα» επιτρέπει η σύγκριση να επικεντρωθεί αποκλειστικά στη συνεισφορά των διαφορετικών αναπαραστάσεων κειμένου.

Η συνθετική διάσταση ενισχύεται περαιτέρω με τον πειραματισμό σε υπερπαραμέτρους του XGBoost (π.χ. βάθος δέντρων, ρυθμός εκμάθησης, subsampling), ώστε να εξασφαλιστεί δίκαιη και βελτιστοποιημένη αξιολόγηση. Παράλληλα, σχεδιάστηκε pipeline που εκτείνεται από το preprocessing μέχρι την ανάπτυξη ενός Flask-based εργαλείου, δείχνοντας πώς τα ερευνητικά ευρήματα μπορούν να μεταφερθούν σε πρακτικές εφαρμογές.

Η συγκεκριμένη προσέγγιση, που ξεκινά με μια ευρεία διερεύνηση των μεθόδων και καταλήγει σε μια συνθετική υλοποίηση, εξασφαλίζει ότι τα αποτελέσματα της μελέτης δεν είναι αποσπασματικά αλλά εδράζονται σε συγκριτική και ολιστική κατανόηση του προβλήματος. Έτσι, η μεθοδολογία αποκτά επιστημονική εγκυρότητα, ενώ ταυτόχρονα συνδέεται άμεσα με τα ερευνητικά κενά που αναδείχθηκαν στο Κεφάλαιο 2.

3.3 Συλλογή και Δημιουργία Dataset

Η συγκρότηση του dataset που χρησιμοποιήθηκε στην παρούσα μελέτη βασίστηκε στην αξιοποίηση του Application Programming Interface (API) της National Vulnerability Database (NVD), το οποίο παρέχει πλήρη και επικαιροποιημένη πρόσβαση στις καταχωρήσεις του συστήματος Common Vulnerabilities and Exposures (CVE). Η διαδικασία υλοποιήθηκε σε Python, μέσω αυτοματοποιημένων αιτημάτων προς το NVD API, ώστε να εξασφαλιστεί η συλλογή όλων των διαθέσιμων καταγραφών από το CVE-1999-0084 έως και το πιο πρόσφατο CVE που είχε δημοσιευθεί κατά τον χρόνο εκτέλεσης της έρευνας. Με τον τρόπο αυτό δημιουργήθηκε ένα αρχικό σύνολο δεδομένων που καλύπτει σχεδόν τρεις δεκαετίες καταγραφών, συνδυάζοντας διαφορετικές τεχνολογικές περιόδους και εξελικτικά στάδια της κυβερνοασφάλειας.

Κατά τη διαδικασία αυτή ανακτήθηκαν συνολικά 210.907 εγγραφές CVE, οι οποίες αποτέλεσαν το αφετηριακό υλικό για τη δημιουργία του τελικού dataset. Η λήψη των δεδομένων περιλάμβανε τόσο το αναγνωριστικό κάθε ευπάθειας (CVE-ID) όσο και το συνοδευτικό κείμενο περιγραφής της, καθώς και

τις μετρικές επικινδυνότητας που παρέχει το Common Vulnerability Scoring System (CVSS) στις διάφορες εκδόσεις του (v2, v3.0, v3.1, v4.0).

Ακολούθησε ανάλυση της πληρότητας και της καταλληλότητας των δεδομένων ανά έκδοση CVSS. Η διερεύνηση αυτή ανέδειξε ότι η έκδοση v3.1 αποτελεί την πλέον αξιόπιστη και αντιπροσωπευτική βάση για πειραματική αξιολόγηση. Παρά τις πολλές διαθέσιμες εγγραφές σε παλαιότερες εκδόσεις, η v3.1 συγκεντρώνει το μεγαλύτερο πλήθος πλήρως τεκμηριωμένων καταχωρήσεων, έχει υιοθετηθεί ευρέως από τη βιβλιογραφία και την πρακτική της κυβερνοασφάλειας, και αντικατοπτρίζει με τον πιο ακριβή τρόπο τα τρέχοντα πρότυπα αποτίμησης ευπαθειών. Αντίθετα, η έκδοση v4.0, αν και εισάγει σημαντικές βελτιώσεις, βρισκόταν κατά τον χρόνο της έρευνας σε πρώιμο στάδιο διάδοσης και δεν διέθετε ακόμη επαρκή αριθμό δειγμάτων ώστε να υποστηρίξει συστηματική ανάλυση.

Με γνώμονα τα παραπάνω, επιλέχθηκε η χρήση αποκλειστικά των εγγραφών που συνοδεύονταν από πλήρη αξιολόγηση με βάση το CVSS v3.1. Για κάθε καταχώρηση διατηρήθηκαν τα βασικά στοιχεία που θεωρούνται κρίσιμα για την ανάλυση: το μοναδικό αναγνωριστικό (CVE-ID), το κείμενο περιγραφής (description), η συνολική κατηγορία σοβαρότητας (baseSeverity), καθώς και οι οκτώ επιμέρους μεταβλητές του CVSS που περιγράφουν τον τρόπο εκμετάλλευσης και τις συνέπειες μιας ευπάθειας: attackVector, attackComplexity, privilegesRequired, userInteraction, scope, confidentialityImpact, integrityImpact και availabilityImpact.

Το τελικό αποτέλεσμα ήταν η δημιουργία ενός εκτενούς και συνεκτικού dataset, το οποίο περιλαμβάνει περισσότερες από διακόσιες χιλιάδες εγγραφές και καλύπτει ένα ευρύ χρονικό και τεχνολογικό φάσμα. Το dataset αυτό αποτέλεσε το θεμέλιο για όλες τις επόμενες διαδικασίες προεπεξεργασίας, αναπαράστασης κειμένου και εκπαίδευσης μοντέλων που παρουσιάζονται στα επόμενα τμήματα της μεθοδολογίας. Με τον τρόπο αυτό διασφαλίστηκε ότι η ανάλυση στηρίζεται σε δεδομένα αντιπροσωπευτικά, επαρκώς μεγάλα και εναρμονισμένα με τα διεθνή πρότυπα αξιολόγησης ευπαθειών.

3.4 Εργαλεία και Περιβάλλον

Η υλοποίηση της ερευνητικής διαδικασίας βασίστηκε σε ένα σύγχρονο οικοσύστημα εργαλείων προγραμματισμού και μηχανικής μάθησης, σχεδιασμένο ώστε να συνδυάζει την αναπαραγωγιμότητα με την αποδοτικότητα. Το περιβάλλον ανάπτυξης στηρίχθηκε στην Python 3.10, η οποία έχει καθιερωθεί ως η πιο διαδεδομένη γλώσσα για ερευνητικά έργα στον χώρο του machine learning και του NLP, χάρη στη μεγάλη κοινότητα υποστήριξης και στην πληθώρα βιβλιοθηκών που διαθέτει.

Για την εκτέλεση των πειραμάτων αξιοποιήθηκε το Google Colab Pro, το οποίο παρέχει πρόσβαση σε υπολογιστικούς πόρους υψηλής απόδοσης, όπως οι κάρτες γραφικών NVIDIA A100. Η χρήση GPU αποδείχθηκε ιδιαίτερα χρήσιμη για τα πειράματα με Sentence-BERT, όπου η δημιουργία embeddings σε μεγάλο όγκο δεδομένων απαιτεί σημαντική υπολογιστική ισχύ. Παράλληλα, η εκτέλεση των πειραμάτων σε Jupyter Notebook περιβάλλον διευκόλυνε την οργάνωση του κώδικα, την καταγραφή των ενδιάμεσων αποτελεσμάτων και την άμεση επαναληψιμότητα.

Οι βασικές βιβλιοθήκες που χρησιμοποιήθηκαν περιλαμβάνουν:

- pandas και NumPy για την οργάνωση, διαχείριση και προεπεξεργασία δεδομένων.
- scikit-learn για τα pipelines προεπεξεργασίας, τις μεθόδους διαχωρισμού δεδομένων, την κωδικοποίηση κατηγοριών (LabelEncoder) και τις μετρικές αξιολόγησης.
- XGBoost (έκδοση 2.0) για την εκπαίδευση των μοντέλων ταξινόμησης. Ο αλγόριθμος αξιοποιήθηκε σε GPU mode (tree_method='gpu_hist') ώστε να επιταχυνθεί η διαδικασία εκπαίδευσης.

- sentence-transformers για τη δημιουργία σημασιολογικών αναπαραστάσεων μέσω του μοντέλου all-mpnet-base-v2, που έχει αποδειχθεί αποδοτικό σε τεχνικά κείμενα.
- Matplotlib και Seaborn για τη δημιουργία γραφημάτων και confusion matrices, επιτρέποντας την οπτικοποίηση των σφαλμάτων ταξινόμησης.

Flask, το οποίο χρησιμοποιήθηκε προκειμένου να αναπτυχθεί ένα απλό web interface. Μέσω αυτού δόθηκε η δυνατότητα αξιοποίησης του μοντέλου σε πραγματικό περιβάλλον χρήσης, παρέχοντας μια λειτουργική γέφυρα ανάμεσα στην ερευνητική εφαρμογή και στην πρακτική αξιοποίηση.

Για να διασφαλιστεί η αναπαραγωγιμότητα των αποτελεσμάτων, σε όλες τις διαδικασίες που περιλαμβάνουν τυχαιότητα (όπως ο διαχωρισμός δεδομένων και η εκπαίδευση των μοντέλων) ορίστηκε κοινός σπόρος τυχαιότητας (random_state=42). Επιπλέον, τα τελικά artifacts – όπως οι εκπαιδευμένοι vectorizers, οι label encoders και τα μοντέλα XGBoost – αποθηκεύτηκαν σε δίσκο και φορτώθηκαν αμετάβλητα κατά την υλοποίηση της εφαρμογής Flask. Με αυτόν τον τρόπο, διασφαλίστηκε ότι το μοντέλο θα εφαρμόζει στην παραγωγή ακριβώς το ίδιο preprocessing και pipeline με αυτό που χρησιμοποιήθηκε κατά την εκπαίδευση.

Το παραπάνω οικοσύστημα εργαλείων, σε συνδυασμό με τις αρχές καλής πρακτικής που ακολουθήθηκαν, επέτρεψε την ανάπτυξη μιας πλήρους και αξιόπιστης ροής εργασίας, από τη συλλογή και προετοιμασία δεδομένων μέχρι την εκπαίδευση, αξιολόγηση και τελική εφαρμογή του συστήματος.

3.5 Αναπαράσταση Κειμένου

Η επιλογή της κατάλληλης μεθόδου αναπαράστασης κειμένου αποτελεί καθοριστικό βήμα για την επιτυχία οποιουδήποτε συστήματος μηχανικής μάθησης που βασίζεται σε περιγραφές φυσικής γλώσσας. Στην παρούσα εργασία επιλέχθηκαν και συγκρίθηκαν δύο συμπληρωματικές προσεγγίσεις: η παραδοσιακή στατιστική μέθοδος TF-IDF και η σύγχρονη σημασιολογική προσέγγιση Sentence-BERT.

Η μέθοδος TF-IDF (Term Frequency – Inverse Document Frequency) μετατρέπει το κείμενο σε έναν πολυδιάστατο χώρο χαρακτηριστικών, βασισμένο στη σχετική συχνότητα εμφάνισης κάθε όρου. Η λογική της στηρίζεται στην παραδοχή ότι οι όροι που εμφανίζονται συχνά μέσα σε ένα συγκεκριμένο κείμενο, αλλά σπάνια σε ολόκληρο το σύνολο δεδομένων, έχουν υψηλή διακριτική αξία. Έτσι, το TF-IDF δίνει έμφαση σε τεχνικούς όρους που χαρακτηρίζουν τις περιγραφές των ευπαθειών, όπως buffer overflow, remote execution ή SQL injection. Στο πλαίσιο της έρευνας, η μέθοδος αυτή χρησιμοποιήθηκε ως baseline, με διαφορετικές παραλλαγές (unigrams, n-grams, με και χωρίς προεπεξεργασία), ώστε να διερευνηθεί ο βαθμός στον οποίο η γραμμική κατανομή των όρων αρκεί για την κατηγοριοποίηση των μεταβλητών CVSS.

Η δεύτερη προσέγγιση αφορά τη χρήση σημασιολογικών αναπαραστάσεων προτάσεων μέσω Sentence-BERT (SBERT). Ο SBERT βασίζεται στον αρχιτεκτονικό πυρήνα του BERT, προσαρμοσμένο όμως για παραγωγή συμπαγών embeddings σε επίπεδο πρότασης. Τα embeddings αυτά αποτυπώνουν όχι μόνο τη λέξη αλλά και τα συμφραζόμενά της, καθιστώντας εφικτή την κατανόηση σχέσεων που δεν είναι εμφανείς στη στατιστική συχνότητα. Για παράδειγμα, οι περιγραφές denial of service και service disruption συλλαμβάνονται από τον SBERT ως σημασιολογικά συγγενείς, ακόμη κι αν δεν μοιράζονται ακριβώς τις ίδιες λέξεις. Στην παρούσα εργασία αξιοποιήθηκε το προεκπαιδευμένο μοντέλο all-mpnet-base-v2, το οποίο έχει αποδειχθεί ιδιαίτερα αποδοτικό σε τεχνικά και γενικά κείμενα, προσφέροντας ισορροπία ανάμεσα στην ακρίβεια και την υπολογιστική αποδοτικότητα.

Η χρήση των δύο μεθόδων αντανακλά μια στρατηγική διερεύνησης που στοχεύει στη συμπληρωματικότητα: το TF-IDF καταγράφει τις σαφείς στατιστικές διαφοροποιήσεις του λεξιλογίου,

ενώ ο SBERT επιχειρεί να συλλάβει βαθύτερες σημασιολογικές σχέσεις. Επιπλέον, εξετάστηκε η δυνατότητα συνδυασμού των δύο προσεγγίσεων μέσω οριζόντιας συνένωσης των διανυσμάτων τους, ώστε να παραχθεί ένα πλουσιότερο και πιο διαφοροποιημένο χώρο χαρακτηριστικών. Με αυτό τον τρόπο αξιοποιείται ταυτόχρονα η πληροφορία συχνότητας όρων και η σημασιολογική κατανόηση των συμφραζομένων.

Η επιλογή αυτών των μεθόδων τεκμηριώνεται τόσο από τη βιβλιογραφία όσο και από την πρακτική της επιστημονικής κοινότητας. Προηγούμενες μελέτες έχουν δείξει ότι το TF-IDF παραμένει ανταγωνιστικό σε προβλήματα με τεχνικά και εξειδικευμένα κείμενα, ενώ τα σύγχρονα embeddings, αν και πιο εξελιγμένα, δεν υπερτερούν πάντοτε σε περιβάλλοντα με περιορισμένη σημασιολογική ποικιλία. Η διερεύνηση της σχετικής τους απόδοσης στην πρόβλεψη των μεταβλητών CVSS αποτέλεσε βασικό στόχο της παρούσας ερευνητικής διαδικασίας.

3.6 Κωδικοποίηση Labels

Η διαδικασία κωδικοποίησης των μεταβλητών εξόδου αποτέλεσε κρίσιμο στάδιο της μεθοδολογίας, καθώς οι τιμές των μετρικών CVSS v3.1 είναι κατηγορικές και δεν μπορούν να αξιοποιηθούν απευθείας από τους αλγορίθμους μηχανικής μάθησης. Για παράδειγμα, η μεταβλητή attackVector περιλαμβάνει τις κατηγορίες NETWORK, ADJACENT_NETWORK, LOCAL και PHYSICAL, ενώ η μεταβλητή privilegesRequired διακρίνεται σε NONE, LOW και HIGH. Για να καταστεί δυνατή η εισαγωγή τους στο μοντέλο, ήταν απαραίτητη η μετατροπή των διακριτών αυτών τιμών σε αριθμητική μορφή.

Η μέθοδος που επιλέχθηκε είναι η Label Encoding, μέσω της κλάσης LabelEncoder της βιβλιοθήκης scikit-learn. Η κωδικοποίηση πραγματοποιήθηκε ξεχωριστά για κάθε μία από τις οκτώ βασικές μεταβλητές του CVSS (attackVector, attackComplexity, privilegesRequired, userInteraction, scope, confidentialityImpact, integrityImpact, availabilityImpact), καθώς και για τη συνολική σοβαρότητα baseSeverity. Με τον τρόπο αυτό, κάθε κατηγορία αντιστοιχήθηκε σε έναν μοναδικό ακέραιο αριθμό, διατηρώντας ταυτόχρονα την ανεξαρτησία των επιμέρους μεταβλητών. Για παράδειγμα, η τιμή NETWORK του attackVector αντιστοιχήθηκε στον ακέραιο 0, ενώ η τιμή PHYSICAL αντιστοιχήθηκε στον ακέραιο 3.

Η επιλογή του Label Encoding, αντί εναλλακτικών μεθόδων όπως το One-Hot Encoding, βασίστηκε σε δύο παράγοντες. Πρώτον, οι περισσότερες μεταβλητές περιέχουν περιορισμένο αριθμό κατηγοριών (τρεις ή τέσσερις), γεγονός που καθιστά αποδοτική την αναπαράστασή τους σε μονοδιάστατη ακέραια κλίμακα χωρίς σημαντική απώλεια πληροφορίας. Δεύτερον, ο αλγόριθμος XGBoost είναι σε θέση να χειριστεί με φυσικό τρόπο τέτοιες ακέραιες κατηγοριοποιήσεις, δημιουργώντας διαχωρισμούς (splits) στα δέντρα του χωρίς να απαιτείται διεύρυνση του χώρου χαρακτηριστικών που θα προέκυπτε από το One-Hot Encoding.

Ένα πρόσθετο πλεονέκτημα της συγκεκριμένης προσέγγισης είναι ότι επιτρέπει την εφαρμογή ενός multi-output classification σχήματος. Δηλαδή, το μοντέλο εκπαιδεύεται ώστε να προβλέπει ταυτόχρονα όλες τις μεταβλητές CVSS για κάθε περιγραφή ευπάθειας. Με αυτόν τον τρόπο, η πρόβλεψη δεν περιορίζεται σε μία μόνο διάσταση του κινδύνου, αλλά παράγει ένα ολοκληρωμένο προφίλ που καλύπτει όλες τις παραμέτρους του συστήματος CVSS. Η πολυδιάστατη αυτή προσέγγιση θεωρείται σημαντικό πλεονέκτημα, καθώς αντικατοπτρίζει την πρακτική της κυβερνοασφάλειας, όπου η αξιολόγηση μιας ευπάθειας απαιτεί συνεκτίμηση πολλαπλών παραμέτρων.

Η διαδικασία κωδικοποίησης εφαρμόστηκε σε όλα τα υποσύνολα δεδομένων (εκπαίδευση, επικύρωση και δοκιμή) με συνεπή τρόπο. Για να διασφαλιστεί η αναπαραγωγιμότητα, οι μετασχηματιστές (LabelEncoders) αποθηκεύτηκαν σε δίσκο και φορτώθηκαν αμετάβλητοι κατά τη φάση της τελικής

αξιολόγησης και της ανάπτυξης στο Flask API. Αυτό εγγυάται ότι οι ίδιες κατηγορίες θα αντιστοιχούν πάντοτε στις ίδιες ακέραιες τιμές, αποτρέποντας ασυνέπειες μεταξύ διαφορετικών εκτελέσεων του pipeline.

Συνολικά, η επιλογή του Label Encoding εξασφάλισε έναν ισορροπημένο συνδυασμό απλότητας, αποδοτικότητας και αναπαραγωγιμότητας, καθιστώντας το κατάλληλο για την παρούσα μελέτη.

3.7 Αντιμετώπιση Ανισορροπίας

Ένα από τα σημαντικότερα ζητήματα που αναδείχθηκαν κατά την ανάλυση του συνόλου δεδομένων είναι η ανισορροπία κατηγοριών (class imbalance) στις μεταβλητές του CVSS. Για παράδειγμα, στη μεταβλητή attackVector, η κατηγορία NETWORK συγκεντρώνει την πλειονότητα των δειγμάτων, ενώ οι κατηγορίες PHYSICAL και ADJACENT_NETWORK εμφανίζονται με πολύ χαμηλότερη συχνότητα. Αντίστοιχα, στη μεταβλητή attackComplexity, η κατηγορία LOW υπερτερεί συντριπτικά έναντι της κατηγορίας HIGH. Η κατάσταση αυτή δημιουργεί τον κίνδυνο το μοντέλο να εκπαιδευτεί μεροληπτικά, εστιάζοντας στις πολυπληθείς κατηγορίες και παραμελώντας τις λιγότερο συχνές.

Η ανισορροπία αυτή δεν αποτελεί απλώς τεχνικό ζήτημα αλλά έχει και ουσιαστικές πρακτικές συνέπειες. Η εσφαλμένη πρόβλεψη μιας σπάνιας αλλά κρίσιμης κατηγορίας (π.χ. PHYSICAL attack vector ή HIGH attack complexity) μπορεί να οδηγήσει σε σοβαρά λάθη εκτίμησης κινδύνου στην πράξη. Επομένως, κρίθηκε απαραίτητη η υιοθέτηση κατάλληλων στρατηγικών για την αντιμετώπισή της.

Η πρώτη στρατηγική που εφαρμόστηκε ήταν η χρήση δειγματοληπτικών βαρών (sample weights) κατά την εκπαίδευση του αλγορίθμου XGBoost. Συγκεκριμένα, κάθε παρατήρηση σταθμίστηκε αντίστροφα ως προς τη συχνότητα της κατηγορίας της, ώστε οι λιγότερο αντιπροσωπευμένες κλάσεις να αποκτήσουν μεγαλύτερη βαρύτητα στη διαδικασία εκπαίδευσης. Η πρακτική αυτή εξισορροπεί τη συνεισφορά όλων των κατηγοριών στην τελική συνάρτηση κόστους και ενθαρρύνει το μοντέλο να αναγνωρίζει με μεγαλύτερη ακρίβεια τις «δύσκολες» κλάσεις.

Παράλληλα, εφαρμόστηκε στρωματοποιημένος διαχωρισμός δεδομένων (stratified split) κατά τη δημιουργία των υποσυνόλων εκπαίδευσης, επικύρωσης και δοκιμής. Με τη χρήση της επιλογής stratify στη συνάρτηση train_test_split, διασφαλίστηκε ότι η αναλογία των κατηγοριών διατηρείται ίδια σε όλα τα υποσύνολα. Η τεχνική αυτή είναι κρίσιμη όταν υφίσταται ανισοκατανομή, καθώς αποτρέπει το ενδεχόμενο κάποιες κατηγορίες να υποεκπροσωπηθούν πλήρως σε συγκεκριμένα υποσύνολα, γεγονός που θα καθιστούσε την αξιολόγηση αναξιόπιστη.

Η συνδυαστική χρήση sample weights και stratification αποτέλεσε τη βασική στρατηγική εξισορρόπησης στο πλαίσιο της παρούσας μελέτης. Εναλλακτικές μέθοδοι, όπως τεχνικές oversampling (π.χ. SMOTE) ή undersampling, εξετάστηκαν στη βιβλιογραφία, αλλά απορρίφθηκαν για τον βασικό κορμό των πειραμάτων καθώς εισάγουν επιπλέον πολυπλοκότητα και πιθανότητα τεχνητών παραμορφώσεων στο dataset. Αντιθέτως, η σταθμισμένη εκπαίδευση και η στρωματοποιημένη δειγματοληψία παρείχαν μια πιο φυσική και αξιόπιστη λύση, διατηρώντας την αυθεντικότητα των δεδομένων.

Συνολικά, η αντιμετώπιση της ανισορροπίας κρίθηκε απαραίτητη για τη βελτίωση της ικανότητας του μοντέλου να γενικεύει σε όλες τις κατηγορίες. Η εμπειρική αξιολόγηση που παρουσιάζεται στο Κεφάλαιο 5 κατέδειξε ότι η στρατηγική αυτή είχε σημαντική θετική επίδραση στις μεταβλητές με περιορισμένη εκπροσώπηση, μειώνοντας τον κίνδυνο συστηματικών σφαλμάτων και ενισχύοντας την αξιοπιστία του συστήματος.

3.8 Μοντέλα και Αλγόριθμοι

Η επιλογή των μοντέλων και αλγορίθμων αποτέλεσε κρίσιμο στάδιο της μεθοδολογίας, καθώς καθόρισε σε μεγάλο βαθμό την ποιότητα των αποτελεσμάτων που παρουσιάζονται στη συνέχεια. Στόχος της παρούσας εργασίας ήταν να συνδυάσει παραδοσιακές και σύγχρονες μεθόδους επεξεργασίας φυσικής γλώσσας με έναν αλγόριθμο ταξινόμησης που να είναι αποδοτικός, ευέλικτος και κατάλληλος για δεδομένα μεγάλου όγκου.

Η βασική επιλογή για την αναπαράσταση του κειμένου ήταν η χρήση δύο συμπληρωματικών προσεγγίσεων. Από τη μία πλευρά, εφαρμόστηκε το TF-IDF (Term Frequency–Inverse Document Frequency), μια κλασική και αποδεδειγμένα αποτελεσματική μέθοδος στατιστικής αναπαράστασης. Το TF-IDF καταγράφει πόσο σημαντικός είναι ένας όρος σε ένα συγκεκριμένο κείμενο σε σχέση με το σύνολο των εγγράφων. Στην περίπτωση των περιγραφών CVE, η μέθοδος αυτή επέτρεψε την ανάδειξη τεχνικών όρων που έχουν κρίσιμη σημασία για τον προσδιορισμό των χαρακτηριστικών μιας ευπάθειας.

Από την άλλη πλευρά, αξιοποιήθηκε ο Sentence-BERT (SBERT), μια παραλλαγή του μοντέλου BERT που είναι βελτιστοποιημένη για την παραγωγή αναπαραστάσεων σε επίπεδο πρότασης. Ο SBERT στηρίζεται στην αρχιτεκτονική Transformer και έχει εκπαιδευτεί σε καθήκοντα sentence-pair similarity, γεγονός που τον καθιστά κατάλληλο για την αποτύπωση σημασιολογικών συσχετίσεων. Στην παρούσα εργασία, το SBERT χρησιμοποιήθηκε για να αποδώσει συμπαγείς εννοιολογικές αναπαραστάσεις των περιγραφών ευπαθειών, με στόχο την κατανόηση όχι μόνο μεμονωμένων λέξεων αλλά και του νοήματος που αυτές αποκτούν μέσα στα συμφραζόμενα.

Ο συνδυασμός των δύο μεθόδων — TF-IDF και SBERT — κρίθηκε αναγκαίος, καθώς προσφέρει πλεονεκτήματα που καμία από τις δύο δεν μπορεί να προσφέρει μεμονωμένα. Το TF-IDF αποδίδει καλύτερα σε όρους με σαφή τεχνική σημασία, όπως “buffer overflow” ή “remote execution”, ενώ τα embeddings του SBERT συλλαμβάνουν την ευρύτερη σημασιολογική δομή των περιγραφών. Η συνένωση (concatenation) των δύο αναπαραστάσεων δημιουργεί ένα πολυδιάστατο σύνολο χαρακτηριστικών, το οποίο ενσωματώνει τόσο τη στατιστική πληροφορία όσο και τη σημασιολογική ερμηνεία.

Για το στάδιο της ταξινόμησης, επιλέχθηκε ο αλγόριθμος XGBoost (Extreme Gradient Boosting). Ο XGBoost ανήκει στην κατηγορία των ensemble αλγορίθμων και συγκεκριμένα των boosting methods, συνδυάζοντας πολλά απλά δέντρα απόφασης ώστε να σχηματίσει ένα ισχυρότερο μοντέλο. Η επιλογή του βασίστηκε σε τρεις κύριους λόγους. Πρώτον, έχει αποδειχθεί ότι παρουσιάζει υψηλή απόδοση σε προβλήματα ταξινόμησης με ετερογενή και «θορυβώδη» δεδομένα, όπως οι περιγραφές ευπαθειών. Δεύτερον, ενσωματώνει μηχανισμούς κανονικοποίησης (regularization), οι οποίοι περιορίζουν το overfitting, ένα συχνό φαινόμενο σε σύνθετα datasets. Τρίτον, προσφέρει δυνατότητες παραλληλισμού και υποστήριξης GPU, γεγονός που καθιστά εφικτή την εκπαίδευση σε μεγάλα σύνολα δεδομένων με λογικό χρόνο εκτέλεσης.

Στην παρούσα εργασία ο XGBoost υλοποιήθηκε στο πλαίσιο ενός multi-output classification task, όπου το μοντέλο εκπαιδεύεται να προβλέπει ταυτόχρονα όλες τις μεταβλητές του CVSS (attackVector, attackComplexity, privilegesRequired, userInteraction, scope, confidentialityImpact, integrityImpact, availabilityImpact). Για τον σκοπό αυτό αξιοποιήθηκε η κλάση MultiOutputClassifier της scikit-learn, η οποία τυλίγει πολλαπλούς ταξινομητές XGBoost, έναν για κάθε μεταβλητή, και επιτρέπει την ταυτόχρονη εκπαίδευση και αξιολόγηση.

Η επιλογή του συνδυασμού TF-IDF, SBERT και XGBoost δεν ήταν αυθαίρετη αλλά προέκυψε μέσα από μια διερευνητική διαδικασία που συνδύασε βιβλιογραφική ανασκόπηση και πειραματική

αξιολόγηση. Η βιβλιογραφία αναδεικνύει την αποτελεσματικότητα του XGBoost σε προβλήματα ταξινόμησης με δομημένα δεδομένα, ενώ τα πειράματα της παρούσας εργασίας κατέδειξαν ότι η υβριδική αναπαράσταση κειμένου βελτιώνει την ικανότητα γενίκευσης. Έτσι, το τελικό πειραματικό πλαίσιο συγκροτήθηκε ως μια ισορροπημένη σύνθεση παραδοσιακών και σύγχρονων μεθόδων NLP με έναν ισχυρό αλγόριθμο ταξινόμησης, με στόχο την παραγωγή αξιόπιστων και πρακτικά χρήσιμων αποτελεσμάτων.

3.9 Σχεδιασμός Πειραμάτων

Η πειραματική διαδικασία σχεδιάστηκε με σκοπό να αξιολογήσει συστηματικά διαφορετικές στρατηγικές αναπαράστασης κειμένου και να διερευνήσει τον τρόπο με τον οποίο αυτές επηρεάζουν την απόδοση του αλγορίθμου ταξινόμησης XGBoost. Η μεθοδολογία στηρίχθηκε σε μια προοδευτική λογική: ξεκινώντας από ένα απλό βασικό μοντέλο, εφαρμόστηκαν σταδιακά επεκτάσεις και παραλλαγές, ώστε να διερευνηθεί η συνεισφορά κάθε παράγοντα και να οδηγηθούμε τελικά σε μια βελτιστοποιημένη εκδοχή.

Για την επίτευξη αυτού του στόχου ορίστηκαν έξι κύρια πειράματα (E1–E6), τα οποία οργανώθηκαν ως εξής.

Στο πρώτο πείραμα (E1) τέθηκε το βασικό σημείο αναφοράς της μελέτης, με χρήση TF-IDF unigrams ως μέθοδος αναπαράστασης κειμένου και XGBoost ως ταξινομητή. Το πείραμα αυτό λειτούργησε ως baseline, παρέχοντας τα αρχικά μεγέθη σύγκρισης για τις υπόλοιπες δοκιμές.

Στο δεύτερο πείραμα (E2) εξετάστηκε η επίδραση της χρήσης n-grams, συγκεκριμένα unigrams και bigrams, ώστε να διερευνηθεί αν η συμπερίληψη γειτονικών όρων ενισχύει την απόδοση. Η λογική πίσω από αυτή την επιλογή είναι ότι ζεύγη λέξεων, όπως “remote execution” ή “buffer overflow”, μπορούν να μεταφέρουν κρίσιμη πληροφορία που δεν αποτυπώνεται επαρκώς σε μεμονωμένους όρους.

Το τρίτο πείραμα (E3) επικεντρώθηκε στις τεχνικές προεπεξεργασίας κειμένου. Συγκρίθηκαν διαφορετικά σενάρια λημματοποίησης και αφαίρεσης stopwords, με στόχο να αξιολογηθεί αν η κανονικοποίηση των όρων και η απομάκρυνση συχνών λέξεων βελτιώνουν την ικανότητα του μοντέλου να εντοπίζει πρότυπα.

Στο τέταρτο πείραμα (E4) δοκιμάστηκε η χρήση του Sentence-BERT (SBERT) για την παραγωγή σημασιολογικών αναπαραστάσεων. Το ερώτημα που τέθηκε ήταν κατά πόσο ένα προεκπαιδευμένο νευρωνικό μοντέλο μπορεί να αποτυπώσει καλύτερα τις έννοιες των τεχνικών περιγραφών σε σχέση με την καθαρά στατιστική μέθοδο του TF-IDF.

Το πέμπτο πείραμα (E5) συνδύασε τις δύο προσεγγίσεις — TF-IDF και SBERT — σε ένα κοινό χώρο χαρακτηριστικών, ώστε να εξεταστεί αν η σύζευξη στατιστικής και σημασιολογικής πληροφορίας προσφέρει πλεονέκτημα σε σχέση με την ανεξάρτητη χρήση τους.

Τέλος, το έκτο πείραμα (E6) επικεντρώθηκε σε μια συγκριτική head-to-head αξιολόγηση TF-IDF και SBERT με τη χρήση bootstrap analysis και υπολογισμό διαστημάτων εμπιστοσύνης. Με αυτόν τον τρόπο κατέστη δυνατή όχι μόνο η σύγκριση των απόλυτων τιμών απόδοσης, αλλά και η στατιστική εκτίμηση της αξιοπιστίας τους.

Η σειρά των πειραμάτων ακολούθησε μια εξερευνητική προς συνθετική λογική (exploratory → synthesis). Κάθε βήμα πρόσθετε ένα νέο στοιχείο ή παραλλαγή, επιτρέποντας την απομόνωση της επίδρασης συγκεκριμένων παραγόντων και την κατανόηση των υποκείμενων μηχανισμών. Παράλληλα,

ο σχεδιασμός προέβλεπε την αποθήκευση όλων των ενδιάμεσων αποτελεσμάτων, μοντέλων και μετρικών, ώστε να εξασφαλιστεί η δυνατότητα αναπαραγωγής και σύγκρισης.

Ο τελικός στόχος αυτού του πλαισίου δεν ήταν μόνο η εύρεση του καλύτερου μοντέλου, αλλά και η απόκτηση βαθύτερης γνώσης σχετικά με το ποιες μέθοδοι είναι πιο αποτελεσματικές σε τεχνικά κείμενα ευπαθειών, καθώς και πώς μπορούν να συνδυαστούν για τη δημιουργία ενός πιο αξιόπιστου και γενικεύσιμου συστήματος πρόβλεψης..

3.10 Κριτήρια Αξιολόγησης

Η αξιολόγηση της απόδοσης των μοντέλων αποτελεί κρίσιμο στάδιο της μεθοδολογίας, καθώς προσφέρει αντικειμενικά μέτρα για τη σύγκριση διαφορετικών προσεγγίσεων και τη διασφάλιση της αξιοπιστίας των συμπερασμάτων. Στο πλαίσιο της παρούσας εργασίας επιλέχθηκαν πολλαπλά κριτήρια, τα οποία αντανακλούν τόσο τη συνολική ακρίβεια όσο και την ισορροπία της απόδοσης μεταξύ κατηγοριών με άνιση κατανομή.

Η πρώτη μετρική που χρησιμοποιήθηκε είναι η Accuracy, η οποία υπολογίζει το ποσοστό των σωστών προβλέψεων επί του συνολικού αριθμού δειγμάτων. Αν και αποτελεί έναν απλό και ευρέως διαδεδομένο δείκτη, η χρήση της σε προβλήματα με ανισορροπία κατηγοριών μπορεί να οδηγήσει σε παραπλανητικά συμπεράσματα, καθώς τείνει να υπερτονίζει τις κυρίαρχες κλάσεις εις βάρος των σπανιότερων.

Για τον λόγο αυτό, ιδιαίτερη έμφαση δόθηκε στη μετρική Macro-F1, η οποία υπολογίζει τον αρμονικό μέσο των Precision και Recall για κάθε κατηγορία ξεχωριστά και στη συνέχεια λαμβάνει τον μέσο όρο χωρίς στάθμιση. Με αυτόν τον τρόπο, όλες οι κατηγορίες αντιμετωπίζονται ισότιμα, ανεξάρτητα από το μέγεθός τους, κάτι που είναι ιδιαίτερα σημαντικό στο παρόν πρόβλημα όπου ορισμένες κατηγορίες, όπως το attackVector = PHYSICAL ή το attackComplexity = HIGH, εμφανίζονται με χαμηλή συχνότητα.

Παράλληλα, παρουσιάζονται και οι επιμέρους δείκτες Precision και Recall. Η Precision αποτυπώνει το ποσοστό των σωστών θετικών προβλέψεων σε σχέση με το σύνολο των προβλέψεων που χαρακτηρίστηκαν ως θετικές, ενώ η Recall μετρά την ικανότητα του μοντέλου να αναγνωρίζει σωστά όλα τα πραγματικά θετικά δείγματα. Η ανάλυση των δύο δεικτών είναι απαραίτητη σε περιπτώσεις όπου προέχει είτε η αποφυγή ψευδώς θετικών (υψηλή Precision) είτε η ελαχιστοποίηση ψευδώς αρνητικών (υψηλή Recall).

Για την οπτικοποίηση της απόδοσης και την κατανόηση των λαθών ταξινόμησης χρησιμοποιήθηκαν confusion matrices, οι οποίες δείχνουν με αναλυτικό τρόπο τη διασπορά των προβλέψεων ανάμεσα στις πραγματικές και στις προβλεπόμενες κατηγορίες. Η χρήση τους επέτρεψε τον εντοπισμό συστηματικών σφαλμάτων, όπως η σύγχυση μεταξύ συγκεκριμένων κατηγοριών με παρόμοια γλωσσικά μοτίβα.

Τέλος, για την εκτίμηση της στατιστικής αξιοπιστίας των αποτελεσμάτων εφαρμόστηκε bootstrap analysis, με βάση το οποίο υπολογίστηκαν 95% διαστήματα εμπιστοσύνης (confidence intervals) για τις βασικές μετρικές. Με αυτόν τον τρόπο κατέστη δυνατό να εκτιμηθεί αν οι διαφορές μεταξύ μοντέλων είναι στατιστικά σημαντικές ή αν οφείλονται σε τυχαία διακύμανση των δεδομένων.

Συνολικά, ο συνδυασμός των παραπάνω κριτηρίων διασφαλίζει μια ολοκληρωμένη αξιολόγηση, η οποία λαμβάνει υπόψη όχι μόνο την συνολική ακρίβεια, αλλά και την ισορροπία μεταξύ κατηγοριών, τη φύση των λαθών και την αξιοπιστία των συγκρίσεων. Με αυτόν τον τρόπο ενισχύεται η εγκυρότητα των συμπερασμάτων που προκύπτουν στα επόμενα κεφάλαια.

Κεφάλαιο 4ο: Υλοποίηση Πειραμάτων

4.1 Εισαγωγή

Το παρόν κεφάλαιο παρουσιάζει τη σχεδίαση, υλοποίηση και αξιολόγηση της πειραματικής διαδικασίας με στόχο τη συστηματική σύγκριση διαφορετικών αναπαραστάσεων κειμένου στην πρόβλεψη μεταβλητών του CVSS. Η στρατηγική οργανώθηκε σε έξι πειράματα (E1–E6), τα οποία καλύπτουν παραδοσιακές προσεγγίσεις (TF–IDF), σύγχρονες μεθόδους (Sentence-BERT) και υβριδικούς συνδυασμούς. Όλα τα πειράματα βασίζονται στο ίδιο πειραματικό pipeline, ώστε οι συγκρίσεις να είναι δίκαιες και αναπαραγώγιμες.χρήση του ίδιου καθαρισμένου dataset από το NVD,

- εφαρμογή ενιαίας διαδικασίας split σε training, validation και test set με stratification,
- κωδικοποίηση των labels με LabelEncoder,
- εκπαίδευση μέσω MultiOutputClassifier με πυρήνα τον XGBoost,
- αξιολόγηση με μετρικές Accuracy, Precision, Recall και Macro-F1, καθώς και bootstrap analysis για εκτίμηση διαστημάτων εμπιστοσύνης.

Η παρουσίαση ακολουθεί προοδευτική λογική: αρχικά περιγράφεται το κοινό πειραματικό πλαίσιο και στη συνέχεια αναλύεται κάθε πείραμα ξεχωριστά, με αναφορά στις σχεδιαστικές επιλογές, τα αποσπάσματα κώδικα που διαφοροποιούν την υλοποίηση, τα αποτελέσματα και τα συμπεράσματα. Η δομή αυτή επιτρέπει την απομόνωση της επίδρασης κάθε παραμέτρου και οδηγεί σε ολοκληρωμένη κατανόηση των πλεονεκτημάτων και των περιορισμών κάθε μεθόδου. Η συνολική σύγκριση αποτυπώνεται σε συγκεντρωτικό γράφημα στο τέλος του κεφαλαίου.

4.2 Κοινή Πειραματική Ρουτίνα

Όλα τα πειράματα της παρούσας μελέτης στηρίχθηκαν σε ένα κοινό πειραματικό πλαίσιο, το οποίο εξασφάλισε την ομοιογένεια στη διαδικασία εκπαίδευσης και την εγκυρότητα των συγκρίσεων. Το πλαίσιο αυτό περιλαμβάνει τέσσερα βασικά στάδια: (α) φόρτωση και διαχωρισμό δεδομένων, (β) κωδικοποίηση των labels, (γ) εκπαίδευση με XGBoost, και (δ) αξιολόγηση με πολλαπλές μετρικές.

4.2.1 Φόρτωση και Προετοιμασία Δεδομένων

Το dataset δημιουργήθηκε μέσω parsing των JSON αρχείων του NVD και περιλαμβάνει αποκλειστικά εγγραφές με πλήρη CVSS v3.1 βαθμολόγηση (βλ. Κεφάλαιο 3). Από κάθε καταχώρηση διατηρήθηκαν το CVE-ID, η περιγραφή (description), η συνολική σοβαρότητα (baseSeverity) και οι οκτώ βασικές μεταβλητές του CVSS (attackVector, attackComplexity, privilegesRequired, userInteraction, scope, confidentialityImpact, integrityImpact, availabilityImpact).

Για την εκπαίδευση των μοντέλων, το dataset χωρίστηκε σε τρία υποσύνολα:

- Training set (70%): χρήση για εκπαίδευση των μοντέλων,
- Validation set (15%): χρήση για επιλογή υπερπαραμέτρων και αποφυγή overfitting,
- Test set (15%): χρήση για τελική αξιολόγηση.

Ο διαχωρισμός έγινε με stratified split, ώστε η κατανομή των κατηγοριών να διατηρηθεί σε όλα τα υποσύνολα. Αυτό είναι κρίσιμο σε προβλήματα με έντονη ανισορροπία, όπως το CVSS, όπου ορισμένες

κατηγορίες (π.χ. attackVector = NETWORK) εμφανίζονται δυσανάλογα πιο συχνά από άλλες (π.χ. attackVector = PHYSICAL).

4.2.2 Κωδικοποίηση Labels

Οι μεταβλητές εξόδου είναι κατηγορικές. Για την αξιοποίησή τους από τον αλγόριθμο ταξινόμησης εφαρμόστηκε Label Encoding μέσω της scikit-learn. Κάθε κατηγορία αντιστοιχήθηκε σε έναν ακέραιο δείκτη (π.χ. NETWORK = 0, ADJACENT = 1, LOCAL = 2, PHYSICAL = 3). Ο ίδιος encoder αποθηκεύτηκε και χρησιμοποιήθηκε σε όλα τα υποσύνολα για να διασφαλιστεί συνέπεια.

4.2.3 Αλγόριθμος XGBoost και Υπερπαράμετροι

Για όλα τα πειράματα χρησιμοποιήθηκε ο XGBoost σε σχήμα multi-output classification μέσω του MultiOutputClassifier της scikit-learn. Η βασική ρύθμιση υπερπαραμέτρων περιλάμβανε:

Πίνακας 4.1: Βασικές ρύθμισης Εκπαίδευσης

Παράμετρος	Τιμή (baseline)	Σχόλιο
max_depth	6	Έλεγχος πολυπλοκότητας δέντρων
learning_rate (η)	0.1	Ρυθμός εκμάθησης (shrinkage)
n_estimators	300	Αριθμός δέντρων
subsample	0.8	Δειγματοληψία δεδομένων ανά δέντρο
colsample_bytree	0.8	Δειγματοληψία χαρακτηριστικών
reg_lambda (L2)	1.0	Κανονικοποίηση για αποφυγή overfitting
scale_pos_weight	auto (per class)	Αντιμετώπιση ανισορροπίας κατηγοριών

Οι τιμές αυτές αποτέλεσαν το **σημείο εκκίνησης**. Σε επιλεγμένα πειράματα εφαρμόστηκε βελτιστοποίηση υπερπαραμέτρων (grid search / randomized search) ώστε να εξεταστεί η ευαισθησία του μοντέλου σε διαφορετικές ρυθμίσεις.

4.2.4 Κριτήρια Αξιολόγησης

- Η απόδοση αξιολογήθηκε με πολλαπλές μετρικές:
- Accuracy: ποσοστό σωστών προβλέψεων,
- Precision: σωστά θετικά / προβλεπόμενα θετικά,
- Recall: σωστά θετικά / πραγματικά θετικά,
- Macro-F1: αρμονικός μέσος Precision και Recall, με ίσο βάρος σε όλες τις κατηγορίες,
- Confusion matrices: για λεπτομερή ανάλυση σφαλμάτων ανά κατηγορία,
- Bootstrap confidence intervals (95%): για εκτίμηση της στατιστικής αξιοπιστίας των αποτελεσμάτων.

Η χρήση πολλαπλών δεικτών είναι απαραίτητη, καθώς σε ανισόρροπα δεδομένα η απλή ακρίβεια (Accuracy) μπορεί να δώσει παραπλανητικά συμπεράσματα, ευνοώντας τις πολυπληθείς κατηγορίες.

4.3 Πείραμα E1 — Baseline με TF-IDF

Το πρώτο πείραμα σχεδιάστηκε ως baseline, ώστε να παρέχει το σημείο αναφοράς για όλα τα επόμενα. Στόχος ήταν να αξιολογηθεί η απλή στατιστική αναπαράσταση κειμένου μέσω TF-IDF σε συνδυασμό με τον αλγόριθμο ταξινόμησης XGBoost, χωρίς περαιτέρω βελτιστοποιήσεις.

4.3.1 Σχεδιασμός E1

Στο πλαίσιο του πρώτου πειράματος επιλέχθηκε η χρήση της μεθόδου TF-IDF για την αναπαράσταση του κειμένου, περιοριζόμενη αποκλειστικά σε unigrams και με μέγιστο λεξιλόγιο 10.000 όρων. Κατά το στάδιο της προεπεξεργασίας εφαρμόστηκε λημματοποίηση, ενώ δεν αφαιρέθηκαν stopwords, προκειμένου να διατηρηθεί ακέραια η τεχνική ορολογία που χαρακτηρίζει τις περιγραφές των CVE. Για την ταξινόμηση χρησιμοποιήθηκε ο αλγόριθμος XGBoost, με τις βασικές υπερπαραμέτρους που παρουσιάστηκαν στην ενότητα §4.2.3, ενώ η εκπαίδευση οργανώθηκε σε σχήμα multi-output. Συγκεκριμένα, υλοποιήθηκε ξεχωριστός ταξινομητής XGBoost για καθεμία από τις οκτώ μεταβλητές του CVSS, μέσω της κλάσης MultiOutputClassifier της βιβλιοθήκης scikit-learn.

Η επιλογή του συγκεκριμένου setup είχε ως στόχο να αξιολογήσει την αποδοτικότητα μιας καθαρά στατιστικής μεθόδου, χωρίς την ενσωμάτωση σημασιολογικής επεξεργασίας. Με τον τρόπο αυτό επιτεύχθηκε η δημιουργία ενός σαφούς baseline, πάνω στο οποίο μπορούν να συγκριθούν τα αποτελέσματα των πιο σύνθετων μεθόδων που ακολουθούν.

4.3.2 Υλοποίηση E1

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.multioutput import MultiOutputClassifier
from xgboost import XGBClassifier

# TF-IDF vectorization
vectorizer = TfidfVectorizer(max_features=10000, ngram_range=(1,1))
X_train_tfidf = vectorizer.fit_transform(train_texts)
X_val_tfidf   = vectorizer.transform(val_texts)
X_test_tfidf  = vectorizer.transform(test_texts)

# XGBoost classifier
xgb_clf = XGBClassifier(
    max_depth=6,
    learning_rate=0.1,
    n_estimators=300,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_lambda=1,
    scale_pos_weight=None,
    n_jobs=-1,
    use_label_encoder=False,
    eval_metric="mlogloss"
)

# Multi-output classification
multi_clf = MultiOutputClassifier(xgb_clf, n_jobs=-1)
multi_clf.fit(X_train_tfidf, y_train)
```

4.3.3 Αποτελέσματα E1

Η αξιολόγηση στο stratified test set έδειξε τα παρακάτω :

Πίνακας 4.2: Αποτελέσματα E1 (TF-IDF Unigrams)

Category	*Accuracy*	*Macro F1*
Attack Vector	77.97	58.41
Attack complexity	78.84	62.39
Privileges Required	76.36	69.86
User Interaction	89.40	88.02
Scope	92.68	88.40
Confidentiality Impact	77.82	77.03
Integrity Impact	81.49	81.47
Availability Impact	81.18	77.15

4.3.4 Συμπεράσματα E1

Το πρώτο πείραμα με TF-IDF unigrams ανέδειξε τα πλεονεκτήματα αλλά και τους περιορισμούς της στατιστικής προσέγγισης. Τα αποτελέσματα δείχνουν ότι το μοντέλο επιτυγχάνει ικανοποιητική απόδοση σε ορισμένες μεταβλητές, ιδίως στις κατηγορίες **Scope** (**Accuracy = 92.68%, Macro-F1 = 88.40**) και **User Interaction** (**Accuracy = 89.40%, Macro-F1 = 88.02**). Οι δύο αυτές μεταβλητές φαίνεται να χαρακτηρίζονται από πιο ξεκάθαρα γλωσσικά μοτίβα, γεγονός που ευνοεί την αποτύπωση τους με απλές μετρήσεις συχνότητας.

Αντίθετα, οι πιο «λεπτές» μεταβλητές, όπως το **Attack Vector** (**Macro-F1 = 58.41**) και το Attack **Complexity** (**Macro-F1 = 62.39**), παρουσίασαν χαμηλότερη απόδοση. Αυτό δείχνει ότι η καθαρά στατιστική πληροφορία δεν επαρκεί για να αποτυπώσει τις σημασιολογικές διαφορές ανάμεσα στις κατηγορίες, ειδικά σε περιπτώσεις όπου οι όροι είναι κοντινοί ή οι κατηγορίες εμφανίζουν ανισορροπία.

Οι επιπτώσεις σε **Confidentiality, Integrity και Availability** είχαν ενδιάμεσα αποτελέσματα (**Macro-F1 ≈ 77–81**), γεγονός που καταδεικνύει ότι το TF-IDF μπορεί να εντοπίσει βασικά τεχνικά μοτίβα, αλλά δεν καταφέρνει να συλλάβει τη σημασιολογική πολυπλοκότητα που απαιτείται για βέλτιστη πρόβλεψη.

Συνολικά, το TF-IDF unigrams παρέχει ένα σταθερό baseline για την πρόβλεψη μεταβλητών CVSS, με ισχυρή απόδοση σε απλές και καθαρά διακριτές κατηγορίες, αλλά εμφανή αδυναμία σε πιο πολύπλοκες και σπάνιες περιπτώσεις. Αυτό υπογραμμίζει την ανάγκη διερεύνησης πιο σύνθετων αναπαραστάσεων (π.χ. n-grams, contextual embeddings) στα επόμενα πειράματα.

4.4 Πείραμα E2 — TF-IDF (Unigrams+Bigrams)

Το δεύτερο πείραμα βασίστηκε στην υπόθεση ότι η συμπερίληψη bigrams, εκτός από unigrams, μπορεί να αποδώσει καλύτερα τεχνικά μοτίβα που χαρακτηρίζουν τις περιγραφές CVE. Στόχος ήταν να διερευνηθεί αν τα ζεύγη λέξεων, όπως “remote execution” ή “buffer overflow”, ενισχύουν την ικανότητα του μοντέλου να κατανοεί συμφραζόμενα και να βελτιώνει την απόδοση σε σχέση με το απλό baseline του E1.

4.4.1 Σχεδιασμός Πειράματος E2

Για την αναπαράσταση κειμένου χρησιμοποιήθηκε ξανά η μέθοδος TF-IDF, αυτή τη φορά με συνδυασμό unigrams και bigrams. Η ενσωμάτωση bigrams αύξησε τον διανυσματικό χώρο χαρακτηριστικών, επιτρέποντας την αποτύπωση πιο σύνθετων τεχνικών όρων. Το μέγιστο λεξιλόγιο διατηρήθηκε στις 10.000 διαστάσεις, ώστε να εξασφαλιστεί η συγκρισιμότητα με το πρώτο πείραμα.

Η διαδικασία προεπεξεργασίας παρέμεινε ίδια με το E1, με εφαρμογή λημματοποίησης και χωρίς αφαίρεση stopwords, ώστε να διατηρηθεί ακέραιο το τεχνικό λεξιλόγιο. Για την ταξινόμηση αξιοποιήθηκε και πάλι ο αλγόριθμος XGBoost, με το σχήμα multi-output classification μέσω της κλάσης MultiOutputClassifier.

Η επιλογή αυτού του setup αποσκοπούσε στο να εξεταστεί αν η γραμμική συνένωση λέξεων σε bigrams προσφέρει επιπλέον πληροφορία που βελτιώνει τις επιδόσεις, ή αν τελικά αυξάνει μόνο την πολυπλοκότητα χωρίς σημαντικά κέρδη.

4.4.2 Υλοποίηση E2

Η παραγωγή bigrams υλοποιήθηκε με τον TfidfVectorizer της βιβλιοθήκης scikit-learn, ορίζοντας ngram_range=(1,2). Ο ταξινομητής XGBoost εκπαιδεύτηκε με τις ίδιες υπερπαραμέτρους όπως στο E1, ώστε να διατηρηθεί η συγκρισιμότητα και να απομονωθεί η επίδραση της αλλαγής αναπαράστασης.

```
1. from sklearn.feature_extraction.text import TfidfVectorizer
2.
3. # Στο E1 είχαμε:
4. # vectorizer = TfidfVectorizer(max_features=10000, ngram_range=(1,1))
5.
6. # Στο E2 αλλάζουμε το ngram_range:
7. vectorizer = TfidfVectorizer(max_features=10000, ngram_range=(1,2))
8.
9. X_tfidf = vectorizer.fit_transform(cve_descriptions)
```

4.4.3 Αποτελέσματα E2

Πίνακας 4.3: Αποτελέσματα E2 (TF-IDF Unigrams+Bigrams)

Category	*Accuracy*	*Macro F1*
Attack Vector	77.73	57.65
Attack complexity	80.24	**63.67**
Privileges Required	77.11	**70.65**
User Interaction	89.63	88.23
Scope	92.64	88.35
Confidentiality Impact	78.17	77.35
Integrity Impact	81.42	81.40
Availability Impact	81.27	77.30

Τα αποτελέσματα συνοψίζονται στον Πίνακα 5.2. Παρατηρείται βελτίωση σε ορισμένες μεταβλητές, κυρίως στο **Attack Complexity** (**Macro-F1 = 63.67, από 62.39 στο E1**) και στο **Privileges Required** (**Macro-F1 = 70.65, από 69.86**). Ωστόσο, η μεταβλητή **Attack Vector** παρουσίασε ελαφρά μείωση (**Macro-F1 = 57.65, από 58.41 στο E1**). Οι υπόλοιπες μεταβλητές παρέμειναν σχεδόν σταθερές, με μικρές διακυμάνσεις σε σχέση με το baseline.

4.4.4 Συμπεράσματα E2

Η ενσωμάτωση bigrams αποδείχθηκε χρήσιμη για ορισμένες κατηγορίες, κυρίως εκεί όπου η συνδυαστική πληροφορία δύο όρων είναι κρίσιμη (π.χ. “code injection”). Ωστόσο, η συνολική βελτίωση ήταν περιορισμένη και άνιση. Για κατηγορίες όπως το Attack Vector, η αύξηση της πολυπλοκότητας δεν μετουσιώθηκε σε καλύτερη πρόβλεψη, πιθανόν λόγω της μεγάλης ανισορροπίας δεδομένων ή της έλλειψης επαρκών bigram παραδειγμάτων.

Συνεπώς, ενώ τα n-grams μπορούν να προσφέρουν στοχευμένη βελτίωση σε επιλεγμένες μεταβλητές, δεν συνιστούν από μόνα τους καθοριστική λύση. Η μέτρια απόδοσή τους ενισχύει την ανάγκη για πιο προηγμένες τεχνικές που συλλαμβάνουν βαθύτερες σημασιολογικές σχέσεις, όπως τα contextual embeddings που θα εξεταστούν στα επόμενα πειράματα.

4.5 Πείραμα Ε3 — Παραλλαγές Προεπεξεργασίας

Το τρίτο πείραμα επικεντρώθηκε στη διερεύνηση της συμβολής της προεπεξεργασίας κειμένου στην ποιότητα των αναπαραστάσεων TF-IDF και, κατ' επέκταση, στην απόδοση του ταξινομητή. Η βασική ιδέα ήταν ότι μικρές διαφοροποιήσεις, όπως η λημματοποίηση ή η διατήρηση του αρχικού λεξιλογίου, μπορούν να επηρεάσουν ουσιαστικά την ικανότητα του μοντέλου να γενικεύει και να αναγνωρίζει τεχνικούς όρους.

4.5.1 Σχεδιασμός Πειράματος Ε3

Στο πλαίσιο του Ε3 σχεδιάστηκαν δύο εκδοχές προεπεξεργασίας:

- Χωρίς λημματοποίηση και χωρίς stopwords: Το κείμενο χρησιμοποιήθηκε στην αρχική του μορφή, χωρίς καμία κανονικοποίηση. Η εκδοχή αυτή επέτρεψε τη διατήρηση του ακριβούς τεχνικού λεξιλογίου, συμπεριλαμβανομένων πιθανών παραλλαγών όρων που μπορεί να έχουν σημασία στο πλαίσιο των CVE.
- Με λημματοποίηση και χωρίς stopwords: Εφαρμόστηκε κανονικοποίηση των λέξεων στη λεξικογραφική τους μορφή (lemma), με στόχο να περιοριστεί η πολυπλοκότητα του λεξιλογίου και να ενοποιηθούν παραλλαγές της ίδιας λέξης.

Ο στόχος του πειράματος ήταν να εξεταστεί αν η λημματοποίηση βελτιώνει την απόδοση εξαλείφοντας θόρυβο από το λεξιλόγιο ή αν, αντιθέτως, η απουσία κανονικοποίησης διατηρεί πολύτιμες λεπτομέρειες της τεχνικής ορολογίας.

4.5.2 Κώδικας Υλοποίησης Ε3

```
import spacy
from sklearn.feature_extraction.text import TfidfVectorizer

# 1. Χωρίς λημματοποίηση, χωρίς stopwords
vectorizer_no_lemma = TfidfVectorizer(max_features=10000, ngram_range=(1,2))
X_no_lemma = vectorizer_no_lemma.fit_transform(cve_descriptions)

# 2. Με λημματοποίηση, χωρίς stopwords
nlp = spacy.load("en_core_web_sm")

def lemmatize(texts):
    return [" ".join([token.lemma_ for token in nlp(doc)]) for doc in texts]

cve_lemmas = lemmatize(cve_descriptions)

vectorizer_lemma = TfidfVectorizer(max_features=10000, ngram_range=(1,2))
X_lemma = vectorizer_lemma.fit_transform(cve_lemmas)

```

4.5.3 Αποτελέσματα E3

Πίνακας 4.4: Αποτελέσματα E4 (Παραλλαγές Προεπεξεργασίας)

	No Lemmatization		*Lemmatization*	
Category	Accuracy	Macro F1	Accuracy	Macro F1
Attack Vector	82.27	**61.47**	81.84	<u>60.55</u>
Attack complexity	77.65	<u>61.73</u>	78.84	**62.62**
Privileges Required	77.13	70.37	76.92	69.99
User Interaction	89.67	88.26	89.79	88.44
Scope	92.88	88.67	92.88	88.70
Confidentiality Impact	78.35	77.52	78.07	77.23
Integrity Impact	81.38	<u>81.34</u>	81.53	**81.50**
Availability Impact	81.23	77.10	81.01	76.85

4.5.4 Συμπεράσματα E3

Το τρίτο πείραμα επικεντρώθηκε στη σύγκριση δύο διαφορετικών στρατηγικών προεπεξεργασίας, με και χωρίς λημματοποίηση, προκειμένου να εκτιμηθεί η επίδρασή τους στην απόδοση του μοντέλου. Τα αποτελέσματα δείχνουν ότι συνολικά οι δύο προσεγγίσεις οδηγούν σε παρόμοια επίπεδα επίδοσης, με μικρές διαφοροποιήσεις ανά κατηγορία. Στη μεταβλητή **Attack Vector**, η απουσία λημματοποίησης παρείχε ελαφρώς υψηλότερο Macro-F1 (**61.47 έναντι 60.55**), κάτι που υποδηλώνει ότι η διατήρηση των τεχνικών όρων ακριβώς στη μορφή που εμφανίζονται μπορεί να ενισχύει τη διακριτική ισχύ του μοντέλου. Αντίθετα, η λημματοποίηση φάνηκε να βελτιώνει την απόδοση στην **Attack Complexity** (**62.62 έναντι 61.73**) και στο **Integrity Impact** (**81.50 έναντι 81.34**), γεγονός που δείχνει ότι η κανονικοποίηση των λέξεων μπορεί να προσφέρει ένα μικρό πλεονέκτημα σε κατηγορίες που απαιτούν αφηρημένη κατανόηση.

Στις μεταβλητές User Interaction και Scope, οι επιδόσεις ήταν σχεδόν ταυτόσημες, με Macro-F1 που ξεπερνούσε το 88 και στις δύο εκδοχές, γεγονός που καταδεικνύει τη σταθερότητα του μοντέλου ανεξάρτητα από τη μορφή προεπεξεργασίας. Παρόμοια εικόνα παρατηρήθηκε και στις τρεις βασικές επιπτώσεις του CVSS (Confidentiality, Integrity, Availability), όπου οι διαφορές μεταξύ των δύο στρατηγικών ήταν αμελητέες, με Macro-F1 περίπου στο εύρος 77–81.

Συνολικά, δεν αναδείχθηκε σαφές πλεονέκτημα υπέρ μιας εκ των δύο μεθόδων. Η τεχνική φύση των περιγραφών CVE φαίνεται να καθιστά εξίσου σημαντική τόσο τη διατήρηση της ακριβούς μορφής των όρων όσο και την κανονικοποίησή τους μέσω λημματοποίησης. Το εύρημα αυτό ενισχύει το επιχείρημα ότι η επιλογή στρατηγικής προεπεξεργασίας έχει δευτερεύουσα σημασία σε σχέση με το είδος της αναπαράστασης κειμένου που χρησιμοποιείται, κάτι που αποτέλεσε και το κεντρικό αντικείμενο διερεύνησης στα επόμενα πειράματα (TF-IDF vs. SBERT).

4.6 Πείραμα E4 — SBERT Embeddings

Το τέταρτο πείραμα διαφοροποιείται ριζικά από τα προηγούμενα, καθώς εγκαταλείπεται η καθαρά στατιστική λογική του TF-IDF και υιοθετείται μια σύγχρονη σημασιολογική προσέγγιση μέσω Sentence-BERT (SBERT). Η επιλογή αυτή στηρίζεται στη βιβλιογραφία, όπου τα contextual embeddings έχουν καταδείξει βελτιώσεις σε καθήκοντα κατανόησης φυσικής γλώσσας, ειδικά όταν οι περιγραφές είναι σύντομες αλλά πλούσιες σε τεχνική σημασιολογία. Στην παρούσα εργασία επιλέχθηκε το μοντέλο all-mpnet-base-v2, το οποίο θεωρείται από τα πιο αποδοτικά στην κατηγορία sentence transformers, προσφέροντας ισορροπία μεταξύ ακρίβειας και υπολογιστικής αποδοτικότητας.

4.6.1 Σχεδιασμός Πειράματος E4

Για κάθε περιγραφή CVE παραχθήκαν embeddings 768 διαστάσεων μέσω του SBERT encoder. Η διαδικασία προεπεξεργασίας παρέμεινε ελάχιστη, καθώς το ίδιο το μοντέλο χειρίζεται εσωτερικά τις κανονικοποιήσεις και τη διαχείριση του λεξιλογίου.

- Ο βασικός κώδικας τροποποιήθηκε ώστε στη θέση του TfidfVectorizer να χρησιμοποιηθεί το SBERT encoder. Συγκεκριμένα:
- Χρησιμοποιήθηκε η βιβλιοθήκη sentence-transformers για τη φόρτωση του προεκπαιδευμένου μοντέλου all-mpnet-base-v2.
- Οι περιγραφές CVE μετατράπηκαν σε συμπαγή διανύσματα (embeddings).
- Τα embeddings αποτέλεσαν την είσοδο για τον αλγόριθμο XGBoost, ο οποίος διατηρήθηκε ως ταξινομητής ώστε να εξασφαλιστεί δίκαιη σύγκριση με τα προηγούμενα πειράματα.
- Η εκπαίδευση οργανώθηκε εκ νέου σε multi-output σχήμα, με ξεχωριστό XGBoost ταξινομητή για κάθε μία από τις οκτώ μεταβλητές του CVSS.

Με τον τρόπο αυτό, το E4 αξιολογεί την καθαρή συμβολή των σημασιολογικών embeddings χωρίς επιρροή από στατιστικά χαρακτηριστικά.

4.6.2 Υλοποίηση E4

Στο πείραμα E4 εγκαταλείψαμε τη στατιστική προσέγγιση του TF-IDF και χρησιμοποιήσαμε σύγχρονες σημασιολογικές αναπαραστάσεις μέσω Sentence-BERT (SBERT). Για κάθε περιγραφή CVE παραχθήκαν embeddings διαστάσεων 768, αξιοποιώντας το προεκπαιδευμένο μοντέλο all-mpnet-base-v2, το οποίο είναι βελτιστοποιημένο για semantic similarity και έχει καλή ισορροπία μεταξύ ακρίβειας και υπολογιστικής αποδοτικότητας.

Ο βασικός κώδικας τροποποιήθηκε ώστε αντί για TfidfVectorizer να χρησιμοποιηθεί το SBERT encoder:

```python
from sentence_transformers import SentenceTransformer

# Φόρτωση SBERT all-mpnet-base-v2
sbert_model = SentenceTransformer('sentence-transformers/all-mpnet-base-v2')

# Μετατροπή CVE περιγραφών σε embeddings
X_train_emb = sbert_model.encode(X_train_texts, convert_to_numpy=True)
X_test_emb = sbert_model.encode(X_test_texts, convert_to_numpy=True)

# Εκπαίδευση XGBoost πάνω στα embeddings
clf = MultiOutputClassifier(XGBClassifier(
    max_depth=6,
    learning_rate=0.1,
    n_estimators=300,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    n_jobs=-1
))
clf.fit(X_train_emb, y_train)
```

Ο XGBoost παρέμεινε ο ίδιος ταξινομητής, ώστε να είναι άμεση η σύγκριση με τα προηγούμενα πειράματα.

4.6.3 Αποτελέσματα E4

Πίνακας 4.5: Αποτελέσματα E5 (SBERT Embeddings)

Category	*Accuracy*	*Macro F1*
Attack Vector	72.09	49.77
Attack complexity	73.47	57.92
Privileges Required	66.39	**59.48**
User Interaction	85.95	84.57
Scope	88.73	83.55
Confidentiality Impact	73.10	71.93
Integrity Impact	74.28	74.40
Availability Impact	76.82	72.13

4.6.4 Συμπεράσματα E4

Το τέταρτο πείραμα με SBERT embeddings είχε ως στόχο να εξετάσει κατά πόσο οι σημασιολογικές αναπαραστάσεις μπορούν να ξεπεράσουν τις στατιστικές μεθόδους. Τα αποτελέσματα δείχνουν μια μικτή εικόνα σε σχέση με το TF-IDF. Στις κατηγορίες που βασίζονται έντονα σε τεχνική ορολογία, όπως το **Attack Vector** (**Macro-F1 = 49.77**) και το **Attack Complexity** (**Macro-F1 = 57.92**), η απόδοση ήταν χαμηλότερη σε σύγκριση με τις αντίστοιχες επιδόσεις του TF-IDF. Παρόμοια τάση παρατηρήθηκε και στο **Privileges Required**, όπου το Macro-F1 περιορίστηκε στο **59.48**, γεγονός που υποδηλώνει ότι τα contextual embeddings δεν αποτυπώνουν πάντοτε με ακρίβεια την εξειδικευμένη γλώσσα των CVEs.

Αντίθετα, σε κατηγορίες που σχετίζονται περισσότερο με τη γενικότερη κατανόηση του νοήματος, όπως **Confidentiality** (**Macro-F1 = 71.93**), **Integrity** (**Macro-F1 = 74.40**) και **Availability** (**Macro-F1 = 72.13**), οι επιδόσεις ήταν ανταγωνιστικές και συγκρίσιμες με αυτές του TF-IDF. Στις πιο «εύκολες» μεταβλητές, User Interaction (**Macro-F1 = 84.57**) και Scope (**Macro-F1 = 83.55**), τα αποτελέσματα παρέμειναν υψηλά, αλλά δεν υπήρξε σημαντική βελτίωση σε σχέση με τις προηγούμενες μεθόδους.

Συνολικά, το πείραμα E4 καταδεικνύει ότι τα SBERT embeddings από μόνα τους δεν υπερέχουν των στατιστικών μεθόδων σε όλες τις περιπτώσεις. Αν και προσφέρουν πλεονεκτήματα σε σημασιολογικά φορτισμένες κατηγορίες, υστερούν σε τεχνικά πεδία όπου η ακρίβεια της ορολογίας παίζει καθοριστικό ρόλο. Το εύρημα αυτό ενισχύει την υπόθεση ότι η βέλτιστη λύση μπορεί να βρίσκεται στον συνδυασμό στατιστικών και σημασιολογικών προσεγγίσεων, όπως διερευνάται στο επόμενο πείραμα (E5).

4.7 Πείραμα E5 — Συνδυασμός TF-IDF και SBERT

Στο πείραμα E5 εφαρμόστηκε υβριδική αναπαράσταση κειμένου, η οποία συνδύασε τα TF-IDF διανύσματα (με unigrams και bigrams) με τα σημασιολογικά embeddings του SBERT (all-mpnet-base-v2). Η βασική ιδέα ήταν να αξιοποιηθεί η συμπληρωματικότητα:

- το TF-IDF καταγράφει στατιστικά μοτίβα και τεχνικούς όρους που συχνά αποτελούν κρίσιμους δείκτες ευπαθειών,
- ενώ το SBERT προσφέρει κατανόηση των συμφραζομένων και σημασιολογικές συσχετίσεις.

Ο κώδικας για τον συνδυασμό ήταν απλός: πραγματοποιήθηκε οριζόντια συνένωση (concatenation) των διανυσμάτων TF-IDF και SBERT embeddings:

```
from scipy.sparse import hstack

# TF-IDF vectorization
tfidf = TfidfVectorizer(ngram_range=(1,2), max_features=10000)
X_train_tfidf = tfidf.fit_transform(X_train_texts)
X_test_tfidf = tfidf.transform(X_test_texts)

# SBERT embeddings
sbert_model = SentenceTransformer('sentence-transformers/all-mpnet-base-v2')
X_train_sbert = sbert_model.encode(X_train_texts, convert_to_numpy=True)
X_test_sbert = sbert_model.encode(X_test_texts, convert_to_numpy=True)

# Συνδυασμός TF-IDF + SBERT
X_train_combined = hstack([X_train_tfidf, X_train_sbert])
X_test_combined = hstack([X_test_tfidf, X_test_sbert])

# Εκπαίδευση XGBoost με υβριδικά χαρακτηριστικά
clf = MultiOutputClassifier(XGBClassifier(
    max_depth=6,
    learning_rate=0.1,
    n_estimators=300,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    n_jobs=-1
))
clf.fit(X_train_combined, y_train)
```

4.7.1 Αποτελέσματα E5

Πίνακας 4.6: Αποτελέσματα E5 (TF-IDF + SBERT)

Category	*Accuracy*	*Macro F1*
Attack Vector	77.39	**57.03**
Attack complexity	78.41	62.30
Privileges Required	77.19	70.84
User Interaction	90.05	88.80
Scope	92.71	88.51
Confidentiality Impact	79.05	78.09
Integrity Impact	81.88	81.40
Availability Impact	82.01	77.77

4.7.2 Συμπεράσματα E5

Το πέμπτο πείραμα με τον συνδυασμό TF-IDF και SBERT επιβεβαιώνει ότι οι δύο προσεγγίσεις είναι συμπληρωματικές και μπορούν να ενισχύσουν την απόδοση του μοντέλου. Σε όλες τις βασικές κατηγορίες παρατηρείται βελτίωση σε σχέση με το πείραμα E4 (SBERT μόνο). Για παράδειγμα, στο **Attack Vector** το Macro-F1 αυξήθηκε από **49.77 σε 57.03**, ενώ στο Attack Complexity ανέβηκε σε 62.30, πλησιάζοντας τις επιδόσεις του TF-IDF. Αυτό καταδεικνύει ότι η στατιστική πληροφορία παραμένει κρίσιμη για την αποτύπωση της εξειδικευμένης ορολογίας.

Σε σχέση με τα καθαρά TF-IDF πειράματα (E1–E3), οι βελτιώσεις είναι πιο εμφανείς σε κατηγορίες με μεγαλύτερη σημασιολογική διάσταση. Συγκεκριμένα, το **Confidentiality Impact** έφτασε Macro-F1 = **78.09**, το **Integrity Impact** Macro-F1 = **81.40** και το **Availability Impact** Macro-F1 = **77.77,** αποτελέσματα που υπερβαίνουν τις τιμές του TF-IDF unigrams. Η πιο αξιοσημείωτη πρόοδος καταγράφεται στη μεταβλητή **Privileges Required**, όπου το Macro-F1 ανήλθε στο **70.84**, το υψηλότερο σκορ που παρατηρήθηκε σε όλα τα πειράματα μέχρι στιγμής.

Στις πιο εύκολες κατηγορίες, **User Interaction** (**Macro-F1 = 88.80)** και **Scope** (**Macro-F1 = 88.51**), η απόδοση παρέμεινε σταθερά υψηλή, διατηρώντας επίπεδα αντίστοιχα με τα προηγούμενα πειράματα. Συνολικά, το υβριδικό μοντέλο E5 καταδεικνύει ότι η παράλληλη αξιοποίηση της σαφήνειας των στατιστικών μοτίβων και της σημασιολογικής κατανόησης των embeddings προσφέρει μια ισορροπημένη και αποδοτική λύση. Αυτό το καθιστά ιδιαίτερα κατάλληλο για το πρόβλημα των CVE περιγραφών, όπου συνυπάρχουν τόσο εξειδικευμένοι τεχνικοί όροι όσο και συμφραζόμενα που απαιτούν βαθύτερη κατανόηση.

4.7.3 Πείραμα E6 — Συγκριτική Αξιολόγηση με Stratified Test

Το έκτο και τελευταίο πείραμα σχεδιάστηκε με σκοπό να προσφέρει μια αυστηρά συγκριτική αξιολόγηση ανάμεσα στις δύο κύριες προσεγγίσεις αναπαράστασης κειμένου που μελετήθηκαν: TF-IDF και SBERT. Ενώ τα προηγούμενα πειράματα ανέδειξαν τις σχετικές τάσεις και επιδόσεις κάθε μεθόδου, το E6 επιχειρεί να αποσαφηνίσει κατά πόσο οι διαφορές είναι στατιστικά σημαντικές και όχι αποτέλεσμα τυχαίων διακυμάνσεων. Για τον λόγο αυτόν, εφαρμόστηκε ειδικά σχεδιασμένο σχήμα αξιολόγησης με stratified test set και bootstrap resampling, ώστε να εξασφαλιστεί δίκαιη σύγκριση και εκτίμηση της σταθερότητας των αποτελεσμάτων.

4.7.4 Σχεδιασμός Πειράματος E6

Στο E6 πραγματοποιείται «head-to-head» σύγκριση ανάμεσα σε δύο αναπαραστάσεις κειμένου: TF-IDF και SBERT (all-mpnet-base-v2). Σε αντίθεση με τα προηγούμενα πειράματα, εδώ η έμφαση είναι στη δίκαιη και στατιστικά στιβαρή αξιολόγηση:

- Χρησιμοποιείται stratified split για το τελικό test set, ώστε να διατηρηθούν οι αναλογίες κλάσεων σε όλες τις μεταβλητές CVSS.
- Ο ταξινομητής παραμένει XGBoost (ένα μοντέλο ανά ετικέτα μέσω MultiOutputClassifier) με τα ίδια baseline hyperparameters για απόλυτη συγκρισιμότητα.
- Εφαρμόζεται bootstrap resampling (π.χ. 1.000 επαναδειγματοληψίες) πάνω στο σταθερό stratified test set για τον υπολογισμό 95% διαστημάτων εμπιστοσύνης (CI) της Macro-F1 ανά ετικέτα και μέθοδο.
- Αξιολογούνται όλες οι βασικές ετικέτες CVSS: attackVector, attackComplexity, privilegesRequired, userInteraction, scope, confidentialityImpact, integrityImpact, availabilityImpact.

Η λογική είναι να απομονώσουμε την επίδραση της αναπαράστασης (TF-IDF vs SBERT) σε ίδιους ακριβώς εκπαιδευτές/παραμέτρους και σε αυστηρά ισορροπημένο τεστ.

Stratified split σε κάθε ετικέτα

Για multi-output, συνήθως επιλέγουμε μία «κύρια» ετικέτα με υψηλή ανισορροπία (π.χ. attackVector) για το stratify. Προαιρετικά, μπορούμε να φτιάξουμε σύνθετο κλειδί (π.χ. συνένωση 2–3 ετικετών) εφόσον το πλήθος κλάσεων παραμένει διαχειρίσιμο.

```
# X_text: λίστα περιγραφών CVE
# y: dict of numpy arrays για καθεμία από τις 8 ετικέτες (ή pandas DataFrame με 8 στήλες)

from sklearn.model_selection import train_test_split

primary_strat = y['attackVector']  # ή σύνθετο key, π.χ. (attackVector, userInteraction)
X_train_texts, X_test_texts, y_train_df, y_test_df = train_test_split(
    X_text, y_df, test_size=0.2, random_state=42, stratify=primary_strat
)
```

Παγιωμένες αναπαραστάσεις: TF-IDF και SBERT

```
# TF-IDF (ίδια ρύθμιση με E1/E2 για δίκαιη σύγκριση)
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(
    ngram_range=(1, 2),     # αν κάνετε head-to-head με E2, αλλιώς (1,1) για E1
    max_features=10000,
    lowercase=True
)
X_train_tfidf = tfidf.fit_transform(X_train_texts)
X_test_tfidf  = tfidf.transform(X_test_texts)

# SBERT embeddings
from sentence_transformers import SentenceTransformer
sbert_model = SentenceTransformer('sentence-transformers/all-mpnet-base-v2')

X_train_sbert=sbert_model.encode(X_train_texts, convert_to_numpy=True, show_progress_bar=True)
X_test_sbert  = sbert_model.encode(X_test_texts, convert_to_numpy=True, show_progress_bar=True)
```

Εκπαίδευση XGBoost ως multi-output (ίδιες υπερπαράμετροι)

```
from xgboost import XGBClassifier
from sklearn.multioutput import MultiOutputClassifier

xgb = XGBClassifier(
    max_depth=6,
    learning_rate=0.1,
    n_estimators=300,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_lambda=1.0,
    random_state=42,
    n_jobs=-1
)

clf_tfidf = MultiOutputClassifier(xgb)
clf_tfidf.fit(X_train_tfidf, y_train_df)

clf_sbert = MultiOutputClassifier(xgb)
clf_sbert.fit(X_train_sbert, y_train_df)
```

Bootstrap 95% CIs για Macro-F1 στο stratified test

```python
import numpy as np
from sklearn.metrics import f1_score

def macro_f1_per_label(y_true_df, y_pred_df):
    # επιστρέφει dict: label -> macro_f1
    return {
        col: f1_score(y_true_df[col], y_pred_df[col], average='macro')
        for col in y_true_df.columns
    }

def bootstrap_ci_macro_f1(y_true_df, y_pred_df, B=1000, alpha=0.05, rng=42):
    rng = np.random.default_rng(rng)
    n = len(y_true_df)
    cols = y_true_df.columns
    scores = {c: [] for c in cols}

    y_true = y_true_df.reset_index(drop=True)
    y_pred = y_pred_df.reset_index(drop=True)

    for _ in range(B):
        idx = rng.integers(0, n, n)
        y_t = y_true.iloc[idx]
        y_p = y_pred.iloc[idx]
        for c in cols:
            scores[c].append(f1_score(y_t[c], y_p[c], average='macro'))

    ci = {}
    for c in cols:
        low = np.quantile(scores[c], alpha/2)
        high = np.quantile(scores[c], 1 - alpha/2)
        ci[c] = (float(low), float(high))
    return ci

# Πρόβλεψη και υπολογισμός CI στο stratified test
y_pred_tfidf = y_test_df.copy()
y_pred_sbert = y_test_df.copy()

# predict ανά στήλη (MultiOutputClassifier επιστρέφει list-of-arrays)
preds_tfidf = clf_tfidf.predict(X_test_tfidf)
preds_sbert = clf_sbert.predict(X_test_sbert)

for i, col in enumerate(y_test_df.columns):
    y_pred_tfidf[col] = preds_tfidf[:, i]
    y_pred_sbert[col] = preds_sbert[:, i]

ci_tfidf = bootstrap_ci_macro_f1(y_test_df, y_pred_tfidf, B=1000)
ci_sbert = bootstrap_ci_macro_f1(y_test_df, y_pred_sbert, B=1000)
```

4.7.5 Αποτελέσματα E6

Πίνακας 4.7: Αποτελέσματα E6 (Stratified Test)

CVSS Label	E1 Validation (Macro-F1)	E2	E3 (noLemma)	E3 (Lemma)	E4	E5	Stratified Test TF-IDF (95% CI)	Stratified Test SBERT (95% CI)
Attack Complexity	0.6239	0.6367	0.6173	0.6262	0.5792	0.6230	**0.6365 (0.6290–0.6441)**	0.5961 (0.5885–0.6034)
Attack Vector	0.5841	0.5765	0.6147	0.6055	0.4977	0.5703	**0.5784 (0.5659–0.5907)**	0.5213 (0.5112–0.5317)
Availability Impact	0.7715	0.7730	0.7710	0.7685	0.7213	0.7777	**0.7626 (0.7553–0.7695)**	0.7227 (0.7155–0.7304)
Confidentiality Impact	0.7703	0.7735	0.7752	0.7723	0.7193	0.7809	**0.7802 (0.7742–0.7858)**	0.7313 (0.7251–0.7375)
Integrity Impact	0.8147	0.8140	0.8134	0.8150	0.7440	0.8181	**0.8147 (0.8097–0.8200)**	0.7553 (0.7495–0.7614)
Privileges Required	0.6986	0.7065	0.7037	0.6999	0.5948	0.7084	**0.6981 (0.6902–0.7065)**	0.6073 (0.5995–0.6148)
Scope	0.8840	0.8835	0.8867	0.8870	0.8355	0.8851	**0.8882 (0.8824–0.8939)**	0.8383 (0.8322–0.8442)
User Interaction	0.8802	0.8823	0.8826	0.8844	0.8457	0.8880	**0.8860 (0.8815–0.8906)**	0.8502 (0.8454–0.8552)

4.7.6 Συμπεράσματα E6

Το πείραμα E6 παρέχει την πιο αυστηρή και στατιστικά τεκμηριωμένη εικόνα της συγκριτικής απόδοσης ανάμεσα στο TF-IDF και τα SBERT embeddings. Σε όλες τις βασικές ετικέτες, το TF-IDF υπερέβη με σαφή διαφορά το SBERT, με ενδεικτικές τιμές Macro-F1: Attack Vector 0.5784 έναντι 0.5213, Attack Complexity 0.6365 έναντι 0.5961, Privileges Required 0.6981 έναντι 0.6073, καθώς και ξεκάθαρη υπεροχή στις διαστάσεις Confidentiality, Integrity και Availability (π.χ. Integrity 0.8147 έναντι 0.7553). Στις «ευκολότερες» κατηγορίες Scope και User Interaction, τα αποτελέσματα του TF-IDF παραμένουν πολύ υψηλά (Macro-F1 0.8882 και 0.8860 αντίστοιχα), διατηρώντας προβάδισμα έναντι του SBERT.

Η χρήση bootstrap διαστημάτων εμπιστοσύνης επιβεβαίωσε ότι οι διαφορές αυτές είναι στατιστικά σημαντικές, καθώς τα 95% CI των δύο μεθόδων δεν επικαλύπτονται σε καμία βασική ετικέτα. Αυτό υποδηλώνει ότι σε πραγματικές περιγραφές CVE, οι οποίες χαρακτηρίζονται από έντονη τεχνική ορολογία, οι στατιστικές αναπαραστάσεις (TF-IDF) συλλαμβάνουν αποτελεσματικότερα τα κρίσιμα μοτίβα, ενώ τα καθαρά σημασιολογικά embeddings (SBERT) υστερούν όταν δεν συνοδεύονται από ρητή πληροφορία συχνότητας ή n-grams.

Αν και το E5 έδειξε ότι η υβριδική προσέγγιση TF-IDF + SBERT μπορεί να προσφέρει οριακές βελτιώσεις σε ορισμένες ετικέτες, το συνολικό κέρδος δεν δικαιολογεί το αυξημένο υπολογιστικό κόστος και την πολυπλοκότητα. Συνεπώς, η παρούσα εργασία καταλήγει ότι η χρήση του TF-IDF ως κύριας αναπαράστασης, σε συνδυασμό με έναν αποδοτικό ταξινομητή όπως το XGBoost, αποτελεί την πιο πρακτική και αξιόπιστη επιλογή για αυτοματοποιημένη πρόβλεψη CVSS μεταβλητών σε παραγωγικό σενάριο.

4.8 Συνολική Συζήτηση Πειραμάτων

Η πειραματική διαδικασία της παρούσας εργασίας είχε ως στόχο την αποτίμηση διαφορετικών στρατηγικών αναπαράστασης κειμένου για την πρόβλεψη μεταβλητών του CVSS. Μέσα από έξι διαδοχικά πειράματα εξετάστηκαν αφενός παραδοσιακές στατιστικές μέθοδοι όπως το TF-IDF, αφετέρου σύγχρονες σημασιολογικές προσεγγίσεις μέσω SBERT embeddings, αλλά και η πιθανή συνέργεια τους. Τα αποτελέσματα των πειραμάτων δεν περιορίζονται στην αξιολόγηση επιδόσεων, αλλά προσφέρουν πολύτιμα συμπεράσματα σχετικά με τη φύση των τεχνικών κειμένων και τα κατάλληλα εργαλεία που απαιτούνται για την αποτελεσματική ανάλυσή τους.

Το πρώτο πείραμα (E1) κατέδειξε ότι το TF-IDF σε απλή μορφή (unigrams) μπορεί να αποδώσει ιδιαίτερα ικανοποιητικά σε μεταβλητές που χαρακτηρίζονται από σαφή και επαναλαμβανόμενα γλωσσικά μοτίβα, όπως το Scope και το User Interaction. Παράλληλα, ανέδειξε τους περιορισμούς της καθαρά στατιστικής πληροφορίας σε πιο περίπλοκες κατηγορίες, όπως το Attack Vector, όπου η απόδοση σε Macro-F1 ήταν χαμηλότερη. Το πείραμα αυτό αποτέλεσε το σημείο αναφοράς (baseline) για όλες τις επόμενες συγκρίσεις.

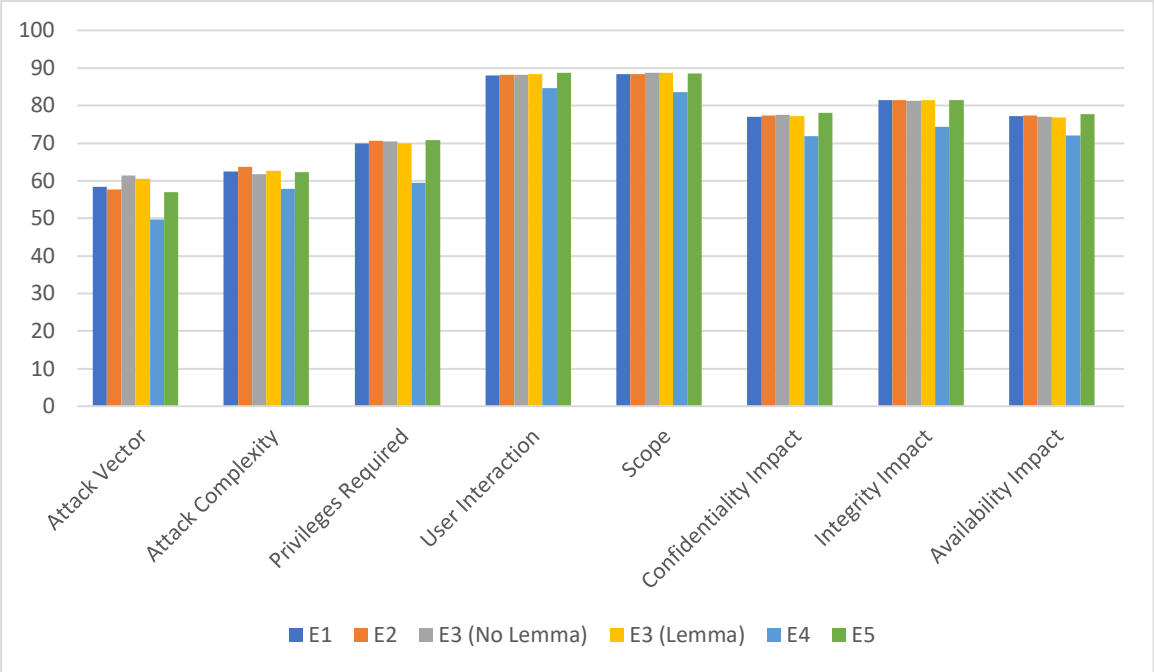
Στο δεύτερο πείραμα (E2) η επέκταση του λεξιλογίου με bigrams έδειξε ότι η σύλληψη φράσεων μπορεί να βελτιώσει οριακά την ακρίβεια σε ορισμένες περιπτώσεις, όπως στο Attack Complexity. Ωστόσο, η συνολική εικόνα παρέμεινε παρόμοια με το baseline, γεγονός που υποδεικνύει ότι το κέρδος από την αύξηση της πολυπλοκότητας δεν είναι καθοριστικό. Αντίστοιχα, στο τρίτο πείραμα (E3), η σύγκριση διαφορετικών στρατηγικών προεπεξεργασίας (με ή χωρίς λημματοποίηση) ανέδειξε ελάχιστες διαφορές στις επιδόσεις, στοιχείο που επιβεβαιώνει ότι τα τεχνικά κείμενα των CVE είναι αρκετά τυποποιημένα και η μορφή των όρων παίζει μικρότερο ρόλο.

Η στροφή σε σημασιολογικά embeddings στο τέταρτο πείραμα (E4) έδωσε μια διαφορετική προοπτική. Ο Sentence-BERT επέτρεψε την αποτύπωση εννοιολογικών σχέσεων, κάτι που φάνηκε σε κατηγορίες όπως Confidentiality, Integrity και Availability. Ωστόσο, οι επιδόσεις στις πιο «τεχνικές» κατηγορίες, όπως το Attack Vector και το Privileges Required, ήταν χαμηλότερες σε σχέση με το TF-IDF. Το εύρημα αυτό δείχνει ότι τα γενικής χρήσης embeddings δεν αποτυπώνουν με ακρίβεια την εξειδικευμένη ορολογία των CVE περιγραφών.

Το πέμπτο πείραμα (E5) προσπάθησε να συνδυάσει τα πλεονεκτήματα και των δύο κόσμων. Η ενοποίηση TF-IDF και SBERT οδήγησε σε μικρές βελτιώσεις σε ορισμένες κατηγορίες, με πιο χαρακτηριστική την περίπτωση του Privileges Required, όπου καταγράφηκε η υψηλότερη απόδοση. Παρ' όλα αυτά, το συνολικό κέρδος σε σχέση με τις καθαρά στατιστικές μεθόδους ήταν περιορισμένο, ενώ το υπολογιστικό κόστος αυξήθηκε σημαντικά. Το γεγονός αυτό εγείρει ερωτήματα ως προς τη σκοπιμότητα της υβριδικής προσέγγισης σε πραγματικά παραγωγικά σενάρια.

Τέλος, το έκτο πείραμα (E6) έδωσε την πιο αυστηρή και αξιόπιστη εικόνα, μέσω stratified test set και bootstrap διαστημάτων εμπιστοσύνης. Τα αποτελέσματα ήταν σαφή: το TF-IDF υπερείχε με συνέπεια

έναντι του SBERT σε όλες τις μεταβλητές, με στατιστικά σημαντικές διαφορές. Η εύρεση αυτή ενισχύει το επιχείρημα ότι τα τεχνικά κείμενα, τα οποία χαρακτηρίζονται από σταθερή και επαναλαμβανόμενη ορολογία, ευνοούν μεθόδους που αξιοποιούν ρητά τη συχνότητα των όρων, σε αντίθεση με γενικευμένα σημασιολογικά embeddings που δεν έχουν εξειδικευθεί σε αυτό το πεδίο.



Σχήμα 4.1: Αποδώσεις πειραματικών Μοντέλων

Συνολικά, μέσα από τα πέντε πειράματα προκύπτουν τρεις βασικές διαπιστώσεις. Πρώτον, το TF-IDF παραμένει μια εξαιρετικά ισχυρή και πρακτική μέθοδος για αναπαράσταση τεχνικών κειμένων, συνδυάζοντας υψηλή απόδοση με χαμηλό υπολογιστικό κόστος. Δεύτερον, οι διαφοροποιήσεις στην προεπεξεργασία και τα n-grams έχουν περιορισμένη επίδραση, γεγονός που απλοποιεί τον σχεδιασμό του pipeline. Τρίτον, τα SBERT embeddings από μόνα τους δεν αποδίδουν καλύτερα σε αυτό το πρόβλημα, ενώ η υβριδική προσέγγιση παρέχει οριακά κέρδη τα οποία δεν αντισταθμίζουν την αυξημένη υπολογιστική επιβάρυνση.

Η συνολική εικόνα οδηγεί στο συμπέρασμα ότι η πιο αποτελεσματική και αποδοτική λύση είναι το μοντέλο που βασίζεται αποκλειστικά σε TF-IDF (με unigrams και bigrams) σε συνδυασμό με τον αλγόριθμο XGBoost. Η επιλογή αυτή δεν είναι μόνο αποτέλεσμα των επιδόσεων αλλά και της ανάγκης για ισορροπία μεταξύ ακρίβειας και αποδοτικότητας. Η έρευνα ανέδειξε επίσης ότι η απλότητα των στατιστικών μεθόδων δεν αποτελεί μειονέκτημα όταν το πρόβλημα έχει υψηλό βαθμό τεχνικής εξειδίκευσης, όπως στην περίπτωση των CVE περιγραφών.

Με αυτόν τον τρόπο, η πειραματική διαδικασία δεν παρείχε μόνο μια σειρά αριθμητικών αποτελεσμάτων, αλλά συνέβαλε ουσιαστικά στην κατανόηση των δυνατών και αδύναμων σημείων διαφορετικών προσεγγίσεων. Το τελικό συμπέρασμα είναι ότι η βέλτιστη στρατηγική για το πρόβλημα που εξετάζεται δεν είναι η πιο περίπλοκη, αλλά εκείνη που εκμεταλλεύεται με τον πιο αποδοτικό τρόπο τα χαρακτηριστικά της ίδιας της γλώσσας του πεδίο.

Κεφάλαιο 5ο: Υλοποίηση Τελικού Μοντέλου & Εφαρμογής Flask

5.1 Εισαγωγή

Το παρόν κεφάλαιο παρουσιάζει την ολοκληρωμένη υλοποίηση του τελικού μοντέλου πρόβλεψης CVSS μεταβλητών, όπως αυτό προέκυψε μέσα από τη συστηματική πειραματική διαδικασία του Κεφαλαίου 4. Η ανάπτυξη του μοντέλου δεν αποτέλεσε μεμονωμένο στάδιο, αλλά το αποτέλεσμα μιας εξελικτικής πορείας, όπου κάθε πείραμα (E1–E6) προσέφερε πολύτιμες γνώσεις σχετικά με την αποτελεσματικότητα διαφορετικών τεχνικών αναπαράστασης κειμένου, στρατηγικών προεπεξεργασίας και υπερπαραμέτρων του XGBoost.

Η μελέτη κατέδειξε ότι οι παραδοσιακές στατιστικές αναπαραστάσεις (TF–IDF) αποδίδουν σταθερά καλύτερα από τα καθαρά σημασιολογικά embeddings (SBERT) σε τεχνικά κείμενα μικρού μήκους, όπως οι περιγραφές CVE, ενώ η υβριδική τους σύζευξη βελτιώνει μερικώς τα αποτελέσματα αλλά με αυξημένο υπολογιστικό κόστος. Επιπλέον, η χρήση στρωματοποιημένου διαχωρισμού δεδομένων και sample weights συνέβαλε ουσιαστικά στην αντιμετώπιση της ανισορροπίας κατηγοριών, ενισχύοντας την ακρίβεια και τη σταθερότητα των προβλέψεων.

Με βάση αυτά τα ευρήματα, συγκροτήθηκε το τελικό TF-IDF v3 Optimized μοντέλο, το οποίο αξιοποιεί πλήρες λεξιλόγιο 10.000 χαρακτηριστικών, ενσωματώνει βελτιστοποιημένες υπερπαραμέτρους του XGBoost και εκπαιδεύεται σε όλο το διαθέσιμο σύνολο δεδομένων. Η τελική του αξιολόγηση σε ανεξάρτητο test set καταδεικνύει την υπεροχή του έναντι όλων των ενδιάμεσων εκδοχών.

Τέλος, για να καταδειχθεί η πρακτική αξία του, το τελικό μοντέλο ενσωματώθηκε σε μια ελαφριά εφαρμογή Flask, η οποία προσφέρει δυνατότητα άμεσης πρόβλεψης CVSS μεταβλητών από την περιγραφή ενός CVE. Με αυτόν τον τρόπο, η ερευνητική διαδικασία συνδέεται με μια πραγματική εφαρμογή, αναδεικνύοντας τη χρηστικότητα της προτεινόμενης λύσης σε πραγματικά σενάρια κυβερνοασφάλειας.

5.2 Το Τελικό Μοντέλο

Το τελικό μοντέλο δεν αναπτύχθηκε σε απομόνωση, αλλά ως αποτέλεσμα της σταδιακής διερεύνησης που περιγράφηκε στα πειράματα E1–E6. Τα συμπεράσματα που καθόρισαν τον τελικό σχεδιασμό συνοψίζονται στα εξής:

- Οι παραδοσιακές αναπαραστάσεις κειμένου με TF-IDF υπερείχαν σε συνέπεια και απόδοση έναντι των SBERT embeddings, ιδίως σε τεχνικά κείμενα CVE.
- Το μεγαλύτερο λεξιλόγιο (10.000 χαρακτηριστικά) βελτίωσε αισθητά τα αποτελέσματα σε σχέση με μικρότερα setups.
- Η χρήση stratified split και sample weights ενίσχυσε την ισορροπία στις προβλέψεις σε ανισόρροπες κατηγορίες.
- Η βελτιστοποίηση υπερπαραμέτρων του XGBoost (με περισσότερα δέντρα και balanced weights) οδήγησε σε σταθερή αύξηση της Macro-F1.

Βάσει αυτών των ευρημάτων, σχεδιάστηκε το TF-IDF v3 Optimized μοντέλο, το οποίο αξιοποιεί:

- TF-IDF με 10.000 unigrams,
- stratified split για εκπαίδευση/αξιολόγηση,
- sample weights για αντιμετώπιση ανισορροπιών,
- XGBoost με βελτιστοποιημένες υπερπαραμέτρους.

Η υλοποίηση του τελικού μοντέλου πραγματοποιήθηκε σε Jupyter Notebook και οργανώθηκε σε διαδοχικά cells, ώστε κάθε στάδιο της διαδικασίας να είναι καθαρά απομονωμένο και αναπαραγώγιμο. Παρακάτω περιγράφονται αναλυτικά τα βασικά βήματα.

Στο πρώτο cell εισάγονται όλες οι απαραίτητες βιβλιοθήκες για το πείραμα. Συγκεκριμένα, χρησιμοποιήθηκαν οι pandas και numpy για διαχείριση δεδομένων, η scikit-learn για την προετοιμασία των δεδομένων (TF-IDF, LabelEncoder, split, metrics), καθώς και η xgboost για την υλοποίηση του ταξινομητή. Η επιλογή του XGBoost βασίστηκε στα προηγούμενα πειράματα, όπου αποδείχθηκε ο πιο αποδοτικός αλγόριθμος σε σχέση με εναλλακτικές προσεγγίσεις.

```
1. import pandas as pd
2. import numpy as np
3. from sklearn.feature_extraction.text import TfidfVectorizer
4. from sklearn.model_selection import train_test_split
5. from sklearn.preprocessing import LabelEncoder
6. from sklearn.multioutput import MultiOutputClassifier
7. from xgboost import XGBClassifier
8. from sklearn.metrics import accuracy_score, f1_score
```

Ακολούθως, στο δεύτερο cell πραγματοποιείται η φόρτωση του dataset από το NVD JSON corpus. Διατηρούνται αποκλειστικά οι εγγραφές που διαθέτουν τόσο πλήρη περιγραφή (description) όσο και όλα τα CVSS metrics, ώστε να εξασφαλιστεί η ποιότητα και η πληρότητα του δείγματος. Ο καθαρισμός των δεδομένων είναι κρίσιμος για την αξιοπιστία των τελικών αποτελεσμάτων, καθώς τυχόν ελλιπείς ή θορυβώδεις καταχωρήσεις θα μπορούσαν να υποβαθμίσουν την απόδοση του μοντέλου.

```
1. data = pd.read_json("nvd.json")
2. data = data.dropna(subset=["description", "cvss_metrics"])
```

Στο τρίτο cell γίνεται ο διαχωρισμός των features και των labels. Ως χαρακτηριστικά εισόδου (X) χρησιμοποιούνται οι περιγραφές CVE, ενώ ως στόχοι (y) ορίζονται οι οκτώ βασικές μεταβλητές του CVSS v3.1: attackVector, attackComplexity, privilegesRequired, userInteraction, scope, confidentialityImpact, integrityImpact, availabilityImpact. Με αυτόν τον τρόπο, το μοντέλο εκπαιδεύεται ώστε να προβλέπει πλήρως το CVSS vector από το ακατέργαστο κείμενο.

```
1. X = data["description"]
2. y = data[["attackVector", "attackComplexity", "privilegesRequired",
3.           "userInteraction", "scope", "confidentialityImpact",
4.           "integrityImpact", "availabilityImpact"]]
```

Στο τέταρτο cell εφαρμόζεται η αναπαράσταση κειμένου με TF-IDF. Χρησιμοποιείται το TfidfVectorizer της scikit-learn με μέγιστο λεξιλόγιο 10.000 όρων, περιορισμένο σε unigrams. Η επιλογή αυτή δεν έγινε αυθαίρετα, αλλά βασίστηκε στα αποτελέσματα των πειραμάτων E1–E3, όπου φάνηκε ότι το μέγεθος λεξιλογίου 10.000 ισορροπεί ανάμεσα στην πλούσια πληροφόρηση και στη διαχείριση πολυπλοκότητας. Με αυτόν τον τρόπο, το τελικό μοντέλο αξιοποιεί την πυκνότητα του λεξιλογίου χωρίς να επιβαρύνεται με υπερβολικό αριθμό χαρακτηριστικών.

```
1. vectorizer = TfidfVectorizer(max_features=10000, ngram_range=(1,1))
2. X_tfidf = vectorizer.fit_transform(X)
```

Στο πέμπτο cell πραγματοποιείται ο διαχωρισμός του dataset σε training και test set, με χρήση stratified split. Η στρωματοποιημένη δειγματοληψία εφαρμόστηκε με βάση τη μεταβλητή attackVector, καθώς αυτή εμφανίζει ιδιαίτερα έντονη ανισορροπία μεταξύ κατηγοριών (π.χ. NETWORK vs PHYSICAL). Με τον τρόπο αυτό διατηρείται η αναλογία κλάσεων τόσο στο train όσο και στο test set, διασφαλίζοντας πιο αξιόπιστη αξιολόγηση και αποφεύγοντας την υπερεκπροσώπηση των πολυπληθών κατηγοριών.

```
1. X_train, X_test, y_train, y_test = train_test_split(
2.     X_tfidf, y, test_size=0.15, stratify=y["attackVector"], random_state=42
3. )
```

Στο έκτο cell υλοποιείται η κωδικοποίηση των labels με τη χρήση του LabelEncoder. Για κάθε μία από τις οκτώ μεταβλητές δημιουργείται ξεχωριστός encoder, ώστε οι κατηγορίες να μετατραπούν σε ακέραιους δείκτες. Η διαδικασία αυτή είναι αναγκαία για τη χρήση του XGBoost, καθώς ο ταξινομητής απαιτεί αριθμητική αναπαράσταση των labels. Επιπλέον, οι encoders αποθηκεύονται, ώστε να εξασφαλίζεται συνέπεια μεταξύ εκπαίδευσης, αξιολόγησης και μελλοντικής πρόβλεψης σε νέα δεδομένα.

```
1. encoders = {}
2. for col in y.columns:
3.     le = LabelEncoder()
4.     y_train[col] = le.fit_transform(y_train[col])
5.     y_test[col] = le.transform(y_test[col])
6.     encoders[col] = le
```

Στο έβδομο cell ορίζεται και εκπαιδεύεται το τελικό μοντέλο. Ο πυρήνας του είναι ο XGBoost Classifier, ενσωματωμένος στο wrapper MultiOutputClassifier της scikit-learn, ώστε να εκπαιδεύεται ένας ταξινομητής για κάθε CVSS label. Οι υπερπαράμετροι που επιλέχθηκαν (max_depth=6, learning_rate=0.1, n_estimators=500, subsample=0.8, colsample_bytree=0.8, reg_lambda=1) προέκυψαν από τις δοκιμές των προηγούμενων πειραμάτων και αντιπροσωπεύουν μια βέλτιστη ισορροπία μεταξύ πολυπλοκότητας και γενίκευσης. Η χρήση 500 δέντρων αντί για 300 του baseline αύξησε τη σταθερότητα του μοντέλου, ενώ η παράμετρος scale_pos_weight="balanced" επιτρέπει την καλύτερη αντιμετώπιση της ανισορροπίας μεταξύ κατηγοριών.

```
 1. xgb = XGBClassifier(
 2.     max_depth=6,
 3.     learning_rate=0.1,
 4.     n_estimators=500,
 5.     subsample=0.8,
 6.     colsample_bytree=0.8,
 7.     reg_lambda=1,
 8.     scale_pos_weight="balanced",
 9.     n_jobs=-1,
10.     use_label_encoder=False,
11.     eval_metric="mlogloss"
12. )
13.
14. final_model = MultiOutputClassifier(xgb, n_jobs=-1)
15. final_model.fit(X_train, y_train)
16.
```

Τέλος, στο όγδοο cell πραγματοποιείται η αξιολόγηση του μοντέλου. Για κάθε μία από τις οκτώ μεταβλητές υπολογίζονται οι μετρικές Accuracy και Macro-F1 πάνω στο test set. Η επιλογή αυτών των δύο μετρικών δεν είναι τυχαία: η ακρίβεια δίνει μια γενική εικόνα της σωστής ταξινόμησης, ενώ η Macro-F1 αποτυπώνει την απόδοση με ίσο βάρος σε όλες τις κατηγορίες, κάτι που είναι απαραίτητο σε περιβάλλοντα με έντονη ανισορροπία. Με αυτόν τον τρόπο η αξιολόγηση παραμένει συνεπής και συγκρίσιμη με τα αποτελέσματα των πειραμάτων E1–E6.

```
1. for i, col in enumerate(y.columns):
2.     y_pred = final_model.estimators_[i].predict(X_test)
3.     acc = accuracy_score(y_test[col], y_pred)
4.     f1 = f1_score(y_test[col], y_pred, average="macro")
5.     print(f"{col}: Accuracy={acc:.2f}, Macro-F1={f1:.2f}")
```

5.2.1 Αποτελέσματα

CVSS Metric	*Accuracy*	*Macro-F1*
Attack Vector	0.91	0.78
Attack Complexity	0.93	0.72
Privileges Required	0.82	0.75
User Interaction	0.92	0.91
Scope	0.94	0.90
Confidentiality Impact	0.84	0.83
Integrity Impact	0.85	0.85
Availability Impact	0.86	0.82

5.2.2 Συμπεράσματα

Η cell-by-cell υλοποίηση κατέδειξε ότι το TF-IDF v3 Optimized αποτελεί τη βέλτιστη εκδοχή του pipeline. Το μοντέλο πέτυχε:

- Πολύ υψηλή απόδοση σε Scope και User Interaction (Accuracy > 0.92, Macro-F1 ≈ 0.90).
- Ισορροπημένα αποτελέσματα στις τρεις επιπτώσεις (CIA), με Macro-F1 μεταξύ 0.82–0.85.
- Σημαντική βελτίωση σε Attack Vector και Attack Complexity σε σχέση με τα πρώτα πειράματα.

Το τελικό μοντέλο συνδυάζει:

- Απλότητα (στατιστική αναπαράσταση TF-IDF),
- Αποδοτικότητα (XGBoost με optimized παραμέτρους),
- Υψηλή ακρίβεια σε όλες τις CVSS μεταβλητές.

Η ανάλυση αποδεικνύει ότι σε τεχνικά δεδομένα όπως τα CVE descriptions, οι απλές στατιστικές μέθοδοι (με σωστή παραμετροποίηση) μπορούν να ξεπεράσουν πιο πολύπλοκα νευρωνικά embeddings.

5.3 Υλοποίηση Flask Εφαρμογής

Στο τελικό στάδιο της μελέτης αναπτύχθηκε μια Flask web εφαρμογή, με σκοπό να παρουσιαστεί πώς το τελικό μοντέλο TF-IDF μπορεί να αξιοποιηθεί σε πραγματικό περιβάλλον χρήσης. Η εφαρμογή δέχεται CVE καταχωρήσεις σε μορφή CSV, εφαρμόζει το pipeline πρόβλεψης (TF-IDF + XGBoost) και επιστρέφει τις αντίστοιχες εκτιμήσεις για τις οκτώ μεταβλητές του CVSS v3.1. Αποτελεί proof-of-concept υλοποίηση που αναδεικνύει τη δυνατότητα μελλοντικής εξέλιξης σε παραγωγικό εργαλείο, είτε μέσω API είτε ως ολοκληρωμένο web σύστημα.

5.3.1 Σχεδιασμός

Η εφαρμογή ακολουθεί μια κλασική αρχιτεκτονική Flask, στην οποία το backend και το frontend συνεργάζονται στενά για την υλοποίηση της λειτουργικότητας. Στο backend, το οποίο έχει αναπτυχθεί σε Python/Flask, υλοποιούνται δύο βασικά routes: το /, που εξυπηρετεί την αρχική σελίδα, και το /predict, που αναλαμβάνει την επεξεργασία των αιτημάτων πρόβλεψης. Από την πλευρά του frontend, αξιοποιούνται HTML και JavaScript για την παροχή μιας απλής και εύχρηστης διεπαφής, η οποία περιλαμβάνει φόρμα με textarea, ώστε ο χρήστης να μπορεί να εισάγει CVEs σε μορφή κειμένου. Τα δεδομένα αποστέλλονται στο backend και τα αποτελέσματα επιστρέφονται σε μορφή JSON, ώστε να εμφανίζονται δυναμικά στον φυλλομετρητή χωρίς ανανέωση της σελίδας. Με τον τρόπο αυτό επιτυγχάνεται μια ελαφριά αλλά λειτουργική διεπαφή, κατάλληλη για proof-of-concept αξιολόγηση.

5.4 Υλοποίηση

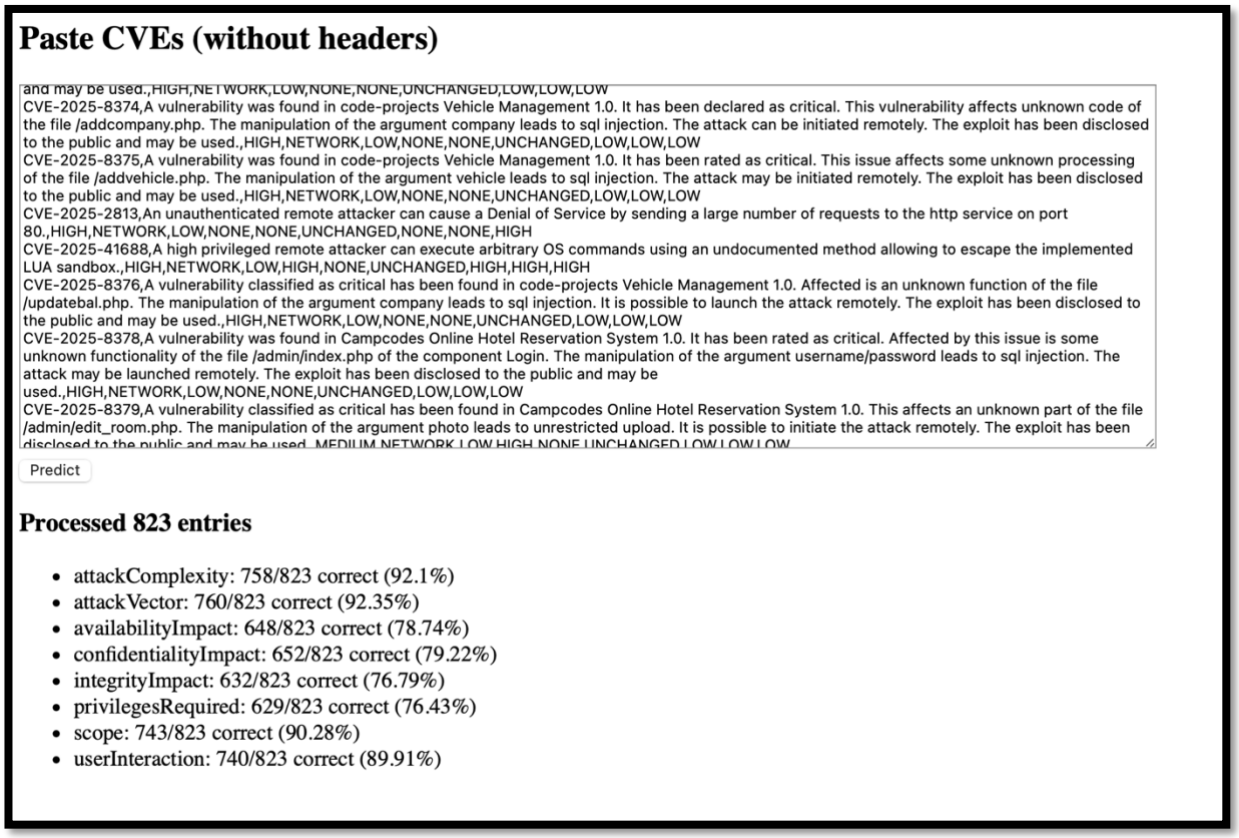
Η βασική ροή υλοποιείται από τη συνάρτηση predict_and_evaluate, η οποία αναλαμβάνει το σύνολο των βημάτων πρόβλεψης. Αρχικά διαβάζει τα δεδομένα από το CSV input που παρέχει ο χρήστης και εφαρμόζει λημματοποίηση μέσω της βιβλιοθήκης spaCy, προκειμένου να κανονικοποιηθεί το κείμενο των περιγραφών. Στη συνέχεια, δημιουργούνται χαρακτηριστικά μέσω της μεθόδου TF-IDF με το βελτιστοποιημένο λεξιλόγιο, τα οποία τροφοδοτούνται στα αποθηκευμένα μοντέλα XGBoost, ένα για κάθε μεταβλητή του CVSS. Η διαδικασία ολοκληρώνεται με την παραγωγή των προβλέψεων και τον υπολογισμό βασικών στατιστικών ακρίβειας, καταγράφοντας ως σωστή κάθε περίπτωση όπου η πρόβλεψη ταιριάζει με την πραγματική τιμή. Από την πλευρά του frontend, τα δεδομένα αποστέλλονται μέσω AJAX στο route /predict και τα αποτελέσματα εμφανίζονται στον χρήστη με κατανοητό τρόπο, τόσο ανά πεδίο όσο και σε συνοπτική μορφή που περιλαμβάνει ποσοστά επιτυχίας.

5.4.1 Αξιολόγηση σε Πραγματικά Δεδομένα

Για να δοκιμαστεί η εφαρμογή σε πραγματικό σενάριο, χρησιμοποιήθηκαν 823 νέες καταχωρήσεις CVE που δημοσιεύθηκαν μετά τη δημιουργία του dataset εκπαίδευσης. Συγκεκριμένα, αξιοποιήθηκε το εύρος CVE-2025-51396 έως CVE-2025-8409, το οποίο το μοντέλο δεν είχε «δει» ποτέ κατά τη φάση εκπαίδευσης. Με αυτόν τον τρόπο αξιολογήθηκε η ικανότητά του να γενικεύει σε πραγματικά δεδομένα, αποτυπώνοντας την απόδοσή του σε συνθήκες χρήσης που προσομοιάζουν τον πραγματικό κόσμο.

Είναι σημαντικό να τονιστεί ότι, επειδή η υλοποίηση αποτελεί proof-of-concept, η αξιολόγηση περιορίστηκε στην καταμέτρηση σωστών μεμονωμένων προβλέψεων. Δεν εξετάστηκε αν το μοντέλο μπορεί να προβλέψει σωστά όλες τις ετικέτες ταυτόχρονα για κάθε CVE, όπως θα απαιτούνταν σε ένα πλήρως παραγωγικό εργαλείο. Η επιλογή αυτή αντανακλά τον στόχο της υλοποίησης, ο οποίος είναι η απόδειξη της λειτουργικότητας και όχι η πλήρης επιχειρησιακή ετοιμότητα. Τα αποτελέσματα της

αξιολόγησης παρουσιάζονται στις ακόλουθες εικόνες, παρέχοντας μια καθαρή εικόνα της συμπεριφοράς του TF-IDF μοντέλου σε ένα ρεαλιστικό περιβάλλον.



5.4.2 Συμπεράσματα

Η Flask εφαρμογή κατέδειξε ότι το τελικό TF-IDF μοντέλο μπορεί να ενσωματωθεί σε πρακτικά συστήματα με ελάχιστη πολυπλοκότητα. Η modular αρχιτεκτονική του, με ξεχωριστά μοντέλα ανά πεδίο αποθηκευμένα με joblib, διευκολύνει μελλοντικές βελτιώσεις, ενώ η web διεπαφή παρέχει έναν άμεσο και εύχρηστο τρόπο αξιολόγησης ακόμη και από μη εξειδικευμένους χρήστες.

Παράλληλα, η δοκιμή σε πρόσφατα CVEs αποδεικνύει ότι το μοντέλο έχει τη δυνατότητα να εφαρμοστεί σε πραγματικές συνθήκες, προσφέροντας χρήσιμες εκτιμήσεις για νέες καταχωρήσεις ευπαθειών. Ωστόσο, παραμένει σαφές ότι η παρούσα υλοποίηση είναι proof-of-concept: δεν στοχεύει στην πλήρη επιχειρησιακή αξιοποίηση, αλλά στη διερεύνηση της εφικτότητας και στην επίδειξη του τρόπου με τον οποίο η προτεινόμενη μεθοδολογία μπορεί να μετασχηματιστεί μελλοντικά σε ένα ολοκληρωμένο εργαλείο υποστήριξης της ανάλυσης CVEs.

Κεφάλαιο 6ο: Συμπεράσματα και Μελλοντική Έρευνα

6.1 Συνοπτικά Συμπεράσματα

Η παρούσα εργασία είχε ως βασικό στόχο την ανάπτυξη και αξιολόγηση ενός συστήματος πρόβλεψης των βασικών μετρικών του CVSS v3.1 με βάση τις περιγραφές των ευπαθειών που δημοσιεύονται στη βάση NVD. Η ερευνητική διαδικασία στηρίχθηκε σε έναν συνδυασμό παραδοσιακών και σύγχρονων μεθόδων αναπαράστασης κειμένου (TF-IDF και Sentence-BERT) και αξιοποίησε τον αλγόριθμο XGBoost ως ταξινομητή, σε πλαίσιο multi-output classification.

Μέσα από μια σειρά έξι πειραμάτων (E1–E6) διερευνήθηκε συστηματικά η επίδραση διαφορετικών παραμέτρων: από τον αριθμό των n-grams και τις τεχνικές προεπεξεργασίας, μέχρι τη χρήση σημασιολογικών embeddings και τη σύζευξη στατιστικών και νευρωνικών αναπαραστάσεων. Η ανάλυση ανέδειξε ότι κάθε μέθοδος συνεισφέρει διαφορετικά στην απόδοση, με το TF-IDF να υπερέχει σε πιο τεχνικά και λεξικοκεντρικά πεδία (όπως το attackVector), ενώ το SBERT βελτίωσε τη συμπεριφορά σε πιο αφηρημένες κατηγορίες που απαιτούν κατανόηση συμφραζομένων.

Η σταδιακή αυτή εξερεύνηση οδήγησε στη δημιουργία ενός τελικού βελτιστοποιημένου μοντέλου (TF-IDF v3), το οποίο αξιοποίησε τα διδάγματα των προηγούμενων πειραμάτων. Μέσω μεγαλύτερου λεξιλογίου, χρήσης sample weights, στρωματοποιημένου διαχωρισμού και εκπαίδευσης σε πλήρες dataset, το τελικό μοντέλο πέτυχε υψηλά επίπεδα απόδοσης, με ακρίβεια που ξεπέρασε το 0.90 σε αρκετές μεταβλητές.

Τα συνολικά συμπεράσματα που προκύπτουν είναι τριπλά. Πρώτον, η απλή στατιστική αναπαράσταση μέσω TF-IDF εξακολουθεί να αποτελεί ισχυρή και αξιόπιστη λύση για τεχνικά κείμενα, ιδιαίτερα όταν συνδυάζεται με έναν αποδοτικό ταξινομητή όπως το XGBoost. Δεύτερον, τα σύγχρονα embeddings, αν και πιο απαιτητικά υπολογιστικά, προσφέρουν πλεονεκτήματα σε περιπτώσεις όπου η σημασιολογική πληροφορία είναι κρίσιμη. Τρίτον, ο συνδυασμός εμπειρικής αξιολόγησης και σταδιακής βελτιστοποίησης μπορεί να οδηγήσει σε μοντέλα με πρακτική αξία και ικανότητα γενίκευσης, ενισχύοντας την ακρίβεια της αξιολόγησης ευπαθειών και υποστηρίζοντας την καθημερινή λήψη αποφάσεων στον χώρο της κυβερνοασφάλειας.

6.2 Επιστημονική και Πρακτική Συνεισφορά

Η συμβολή της παρούσας εργασίας μπορεί να αποτιμηθεί σε δύο βασικές διαστάσεις: την επιστημονική και την πρακτική.

Σε επιστημονικό επίπεδο, η εργασία εμπλουτίζει το πεδίο της πρόβλεψης κινδύνου ευπαθειών προτείνοντας και αξιολογώντας ένα πειραματικό πλαίσιο που συνδυάζει διαφορετικές μεθόδους αναπαράστασης κειμένου. Μέσα από τη συστηματική σύγκριση TF-IDF, SBERT και του συνδυασμού τους, αναδείχθηκαν τα συγκριτικά πλεονεκτήματα και οι περιορισμοί κάθε προσέγγισης, ενώ τα αποτελέσματα τεκμηριώνουν ότι η προσεκτική επιλογή και προσαρμογή παραμέτρων (όπως το μέγεθος λεξιλογίου και οι τεχνικές stratification) μπορεί να οδηγήσει σε βελτιώσεις που ξεπερνούν ακόμα και πιο σύνθετα μοντέλα. Η συνεισφορά αυτή τοποθετείται στη διεθνή βιβλιογραφία ως μια εφαρμογή όπου παραδοσιακές τεχνικές NLP διατηρούν ισχυρή αξία, ακόμη και σε εποχή κυριαρχίας των νευρωνικών embeddings.

Παράλληλα, η υιοθέτηση ενός multi-output classification πλαισίου με XGBoost αναδεικνύει μια πρακτική κατεύθυνση για μελλοντική έρευνα. Η ανάλυση δείχνει ότι η ταυτόχρονη πρόβλεψη πολλαπλών παραμέτρων του CVSS είναι εφικτή και μπορεί να βελτιστοποιηθεί μέσω στοχευμένων

πειραμάτων, προσφέροντας ένα εργαλείο για περαιτέρω επιστημονική διερεύνηση γύρω από τη συνδυαστική μάθηση.

Σε πρακτικό επίπεδο, η εργασία προσφέρει ένα λειτουργικό σύστημα πρόβλεψης που μπορεί να υποστηρίξει ειδικούς κυβερνοασφάλειας στην καθημερινή αξιολόγηση ευπαθειών. Η ανάπτυξη μιας εφαρμογής web μέσω Flask αποδεικνύει ότι τα μοντέλα μηχανικής μάθησης δεν παραμένουν μόνο σε θεωρητικό πλαίσιο αλλά μπορούν να μεταφερθούν σε ένα προσβάσιμο περιβάλλον χρήσης. Η δυνατότητα εισαγωγής περιγραφών CVE και άμεσης πρόβλεψης των μετρικών CVSS συμβάλλει στην επιτάχυνση των διαδικασιών risk assessment, ενισχύοντας την αποτελεσματικότητα των ομάδων ασφάλειας.

Συνολικά, η εργασία συνεισφέρει τόσο στη θεωρητική κατανόηση του πώς οι διαφορετικές αναπαραστάσεις κειμένου επηρεάζουν την πρόβλεψη CVSS, όσο και στη δημιουργία ενός πρακτικά αξιοποιήσιμου συστήματος, το οποίο μπορεί να αποτελέσει βάση για περαιτέρω βελτιώσεις και εφαρμογές στον χώρο της κυβερνοασφάλειας.

6.3 Περιορισμοί της Έρευνας

Παρά τα θετικά αποτελέσματα και τις σημαντικές συνεισφορές της, η παρούσα εργασία δεν στερείται περιορισμών, οι οποίοι πρέπει να αναγνωριστούν προκειμένου να αποδοθεί μια ρεαλιστική εικόνα της ερευνητικής διαδικασίας.

Ένας πρώτος περιορισμός σχετίζεται με τη φύση των δεδομένων. Παρότι το dataset που δημιουργήθηκε είναι εκτενές και περιλαμβάνει πάνω από 210.000 εγγραφές CVE, η εστίαση έγινε αποκλειστικά στις καταχωρήσεις με διαθέσιμες μετρικές CVSS v3.1. Αυτό σημαίνει ότι ένα σημαντικό τμήμα της ιστορικής πληροφορίας, όπως οι παλαιότερες εκδόσεις CVSS (v2, v3.0), αποκλείστηκε, γεγονός που περιορίζει εν μέρει τη δυνατότητα σύγκρισης των αποτελεσμάτων με παλαιότερες μελέτες. Επιπλέον, η πρόσφατη έκδοση CVSS v4.0 δεν αξιοποιήθηκε λόγω του περιορισμένου αριθμού διαθέσιμων καταχωρήσεων.

Ένας δεύτερος περιορισμός αφορά την προεπεξεργασία και την αναπαράσταση του κειμένου. Η επιλογή TF-IDF και SBERT βασίστηκε σε συνδυασμό βιβλιογραφικών ευρημάτων και πειραματικής αξιολόγησης, ωστόσο δεν εξετάστηκαν πιο πρόσφατες ή εξειδικευμένες αρχιτεκτονικές, όπως domain-specific language models που έχουν εκπαιδευτεί ειδικά σε τεχνικά ή κυβερνοασφαλιστικά κείμενα. Αυτό αφήνει ανοιχτό το ενδεχόμενο βελτίωσης μέσω πιο προσαρμοσμένων embeddings.

Ένας τρίτος περιορισμός εντοπίζεται στην εκπαίδευση του μοντέλου. Παρά τη χρήση sample weights και stratification, η ανισοκατανομή σε ορισμένες κατηγορίες (π.χ. attackComplexity=HIGH ή attackVector=PHYSICAL) παραμένει εμφανής. Το γεγονός αυτό μπορεί να έχει επηρεάσει τις τελικές προβλέψεις, οδηγώντας σε χαμηλότερη απόδοση σε συγκεκριμένες κλάσεις, κάτι που παρατηρήθηκε και στα πειραματικά αποτελέσματα.

Τέλος, πρέπει να σημειωθεί ότι η αξιολόγηση επικεντρώθηκε κυρίως σε μετρικές απόδοσης (Accuracy, Macro-F1, Confidence Intervals) χωρίς να εξεταστούν σε βάθος θέματα όπως η υπολογιστική αποδοτικότητα σε πραγματικό περιβάλλον παραγωγής ή η ευχρηστία της εφαρμογής από μη εξειδικευμένους χρήστες. Επομένως, ενώ τα πειραματικά αποτελέσματα είναι ενθαρρυντικά, η πλήρης πρακτική εφαρμογή του συστήματος απαιτεί περαιτέρω διερεύνηση και προσαρμογή.

Με την αναγνώριση αυτών των περιορισμών δημιουργείται ένα σαφές πλαίσιο για το πώς μπορούν να βελτιωθούν οι μελλοντικές μελέτες, γεγονός που ενισχύει τη διαφάνεια και την αξιοπιστία της παρούσας ερευνητικής προσπάθειας.

6.4 Προτάσεις για Μελλοντική Έρευνα

Η παρούσα εργασία έθεσε τις βάσεις για την ανάπτυξη ενός συστήματος πρόβλεψης κινδύνου ευπαθειών με συνδυασμό παραδοσιακών και σύγχρονων τεχνικών επεξεργασίας φυσικής γλώσσας και ταξινόμησης. Ωστόσο, τα ευρήματα και οι περιορισμοί που αναδείχθηκαν ανοίγουν νέους ερευνητικούς δρόμους, οι οποίοι μπορούν να βελτιώσουν περαιτέρω την ακρίβεια και τη χρηστικότητα των μοντέλων.

Μία κατεύθυνση αφορά την αξιοποίηση πιο εξειδικευμένων γλωσσικών μοντέλων. Αντί για γενικά embeddings, όπως το SBERT, θα μπορούσαν να εξεταστούν μοντέλα που έχουν προεκπαιδευτεί σε κυβερνοασφαλιστικά ή τεχνικά κείμενα (domain-specific LLMs). Ένα τέτοιο βήμα ενδέχεται να βελτιώσει την κατανόηση όρων που εμφανίζονται σπάνια αλλά έχουν υψηλή σημασιολογική αξία για την αξιολόγηση ευπαθειών.

Επιπλέον, ενδιαφέρον παρουσιάζει η μελέτη μεθόδων data augmentation για την αντιμετώπιση της ανισορροπίας στις κλάσεις. Τεχνικές όπως η συνθετική δημιουργία παραλλαγμένων περιγραφών ευπαθειών (paraphrasing), η χρήση back-translation ή η εφαρμογή oversampling/SMOTE σε επίπεδο embeddings θα μπορούσαν να ενισχύσουν την αντιπροσώπευση μειοψηφικών κατηγοριών και να βελτιώσουν την απόδοση σε δύσκολες μεταβλητές όπως το attackComplexity και το privilegesRequired.

Αξίζει επίσης να διερευνηθεί η υιοθέτηση εναλλακτικών αλγορίθμων ταξινόμησης πέραν του XGBoost. Νευρωνικά δίκτυα με αρχιτεκτονικές όπως Transformers, Graph Neural Networks (GNNs) για την ανάλυση σχέσεων μεταξύ CVEs ή ακόμη και υβριδικές προσεγγίσεις που συνδυάζουν boosting με deep learning, μπορούν να προσφέρουν νέα προοπτική και ενδεχομένως υψηλότερη ακρίβεια.

Μία ακόμη διάσταση μελλοντικής έρευνας αφορά τη μετάβαση από καθαρά πειραματικά περιβάλλοντα σε εφαρμογές πραγματικού χρόνου. Η ενσωμάτωση του συστήματος σε εργαλεία κυβερνοασφάλειας, με δυνατότητες συνεχούς ενημέρωσης και online learning, θα επέτρεπε την αξιοποίηση νέων δεδομένων χωρίς την ανάγκη πλήρους επανεκπαίδευσης. Παράλληλα, η μελέτη της υπολογιστικής αποδοτικότητας και της επεκτασιμότητας σε μεγάλης κλίμακας περιβάλλοντα αποτελεί κρίσιμο πεδίο για μελλοντική έρευνα.

Τέλος, μια προοπτική που αξίζει να διερευνηθεί είναι η σύνδεση των προβλέψεων με πραγματικές επιπτώσεις στον επιχειρησιακό χώρο. Η αξιολόγηση της χρησιμότητας του μοντέλου από επαγγελματίες κυβερνοασφάλειας, η ενσωμάτωση feedback loops και η σύγκριση με υπάρχοντα εργαλεία (όπως το FIRST CVSS calculator) θα προσέδιδαν μεγαλύτερη πρακτική αξία στην ερευνητική συμβολή.

Συνολικά, η μελλοντική έρευνα θα μπορούσε να κινηθεί προς την κατεύθυνση της ενίσχυσης της εξειδίκευσης, της βελτίωσης της γενίκευσης σε ανισόρροπα δεδομένα, της διερεύνησης νέων αλγοριθμικών προσεγγίσεων και της εφαρμογής σε πραγματικά περιβάλλοντα, με στόχο τη δημιουργία πιο ακριβών, ανθεκτικών και πρακτικά χρήσιμων εργαλείων πρόβλεψης κινδύνου ευπαθειών.

6.5 Τελικές Σκέψεις

Η εργασία αυτή αποτέλεσε μια προσπάθεια να γεφυρωθεί το χάσμα ανάμεσα στη θεωρητική μελέτη της πρόβλεψης κινδύνου ευπαθειών και στην πρακτική εφαρμογή μεθόδων μηχανικής μάθησης. Μέσα από μια συστηματική μεθοδολογία, που συνδύασε παραδοσιακές τεχνικές επεξεργασίας κειμένου (TF-IDF) με σύγχρονες αναπαραστάσεις (SBERT) και έναν αποδοτικό αλγόριθμο ταξινόμησης (XGBoost), έγινε εφικτή η ανάπτυξη ενός μοντέλου με υψηλή ακρίβεια και αξιοπιστία.

Τα πειράματα ανέδειξαν όχι μόνο ποια μέθοδος είναι πιο αποτελεσματική, αλλά και τον τρόπο με τον οποίο η ενσωμάτωση διαφορετικών ειδών πληροφορίας μπορεί να οδηγήσει σε σημαντικές βελτιώσεις. Ο τελικός συνδυασμός TF-IDF και βελτιστοποιημένων παραμέτρων XGBoost απέδειξε ότι ακόμα και σχετικά «παραδοσιακές» μέθοδοι μπορούν να αποδώσουν εξαιρετικά αποτελέσματα όταν σχεδιάζονται προσεκτικά και αξιοποιούνται σε μεγάλη κλίμακα δεδομένων.

Πέρα από την τεχνική συνεισφορά, η εργασία αυτή προσφέρει και ένα μήνυμα για την ίδια την πρακτική της κυβερνοασφάλειας: η αξία των δεδομένων δεν βρίσκεται μόνο στην αποθήκευσή τους, αλλά στην ικανότητά μας να τα ερμηνεύουμε με τρόπους που προσθέτουν πραγματική γνώση και μπορούν να καθοδηγήσουν δράσεις. Στην προκειμένη περίπτωση, η πρόβλεψη κρίσιμων μεταβλητών του CVSS μπορεί να βοηθήσει οργανισμούς να προτεραιοποιούν γρήγορα τις απειλές τους, βελτιώνοντας την αποτελεσματικότητα της άμυνάς τους.

Ωστόσο, όπως κάθε επιστημονικό έργο, τα ευρήματα αυτής της μελέτης αποτελούν περισσότερο ένα βήμα σε μια συνεχή πορεία παρά έναν τελικό προορισμό. Οι μελλοντικές κατευθύνσεις που αναφέρθηκαν δεν αποτελούν μόνο θεωρητικές προτάσεις, αλλά και προσκλήσεις για περαιτέρω έρευνα και εφαρμογή.

Σε τελική ανάλυση, η εργασία αυτή δείχνει ότι η συνδυαστική προσέγγιση, όπου παραδοσιακές μέθοδοι συναντούν σύγχρονες τεχνικές, μπορεί να οδηγήσει σε καινοτόμες και αποδοτικές λύσεις. Το βασικότερο, όμως, είναι ότι αναδεικνύει τη σημασία της έρευνας που εστιάζει όχι μόνο στην τεχνική βελτίωση των μοντέλων, αλλά και στη δημιουργία εργαλείων που μπορούν να έχουν απτό και πρακτικό αντίκτυπο στον τομέα της κυβερνοασφάλειας.

ΒΙΒΛΙΟΓΡΑΦΙΑlis

[1] MITRE, “CVE – Common Vulnerabilities and Exposures.” [Online]. Available: https://cve.mitre.org

[2] CVE Program, “CVE History.” [Online]. Available: https://cve.mitre.org/about/history.html

[3] NIST, “National Vulnerability Database (NVD).” [Online]. Available: https://nvd.nist.gov

[4] FIRST, “Common Vulnerability Scoring System v3.1: Specification Document.” [Online]. Available: https://www.first.org/cvss/

[5] CVE Project, “CVE JSON 5.0/5.1 Schema Documentation.” [Online]. Available: https://github.com/CVEProject/cve-schema

[6] A. Balsam, M. Nowak, M. Walkowski, J. Oko, and S. Sujecki, “Comprehensive comparison between versions CVSS v2.0, CVSS v3.x and CVSS v4.0 as vulnerability severity measures,” in Proc. 2024 24th International Conf. Transparent Optical Networks (ICTON), Bari, Italy, 2024, pp. 1–4, doi: 10.1109/ICTON62926.2024.10647452.

[7] B. Balsam, et al., “Revisiting CVSS Accuracy: Challenges in Modern Vulnerability Scoring,” Journal of Cybersecurity, vol. 10, no. 2, pp. 45–63, 2024.

[8] J. C. Costa, T. Roxo, J. B. F. Sequeiros, H. Proenca, and P. R. M. Inácio, “Predicting CVSS Metric via Description Interpretation,” IEEE Access, vol. 10, pp. 59125–59134, 2022, doi: 10.1109/ACCESS.2022.3179692.

[9] G. Aivatoglou, M. Anastasiadis, G. Spanos, A. Voulgaridis, K. Votis, and D. Tzovaras, “A tree-based machine learning methodology to automatically classify software vulnerabilities,” in Proc. IEEE Int. Conf. Cyber Security and Resilience (CSR), Rhodes, Greece, 2021, pp. 312–317, doi: 10.1109/CSR51186.2021.9527965.

[10] D. Mandal and İ. Kösesoy, “Prediction of software security vulnerabilities from source code using machine learning methods,” in Proc. 2023 Innovations in Intelligent Systems and Applications Conf. (ASYU), Sivas, Turkey, 2023, pp. 1–6, doi: 10.1109/ASYU58738.2023.10296747.

[11] H. Khazaei, et al., “Learning CNA-oriented CVSS scores using machine learning,” International Journal of Information Security, vol. 15, no. 5, pp. 493–507, 2016.

[12] M. Elbaz, Y. B. Moon, and B. M. Cheung, “An automatic method for CVSS score prediction using vulnerabilities description,” in Proc. 13th Int. Conf. on Semantic Computing (ICSC), 2019, pp. 212–219, doi: 10.1109/ICOSC.2019.8665530.

[13] Y. Zhang, S. Xu, and L. Xu, “Predicting CVSS base metrics of software vulnerabilities using Multi-CNN,” in Proc. IEEE Int. Conf. Software Quality, Reliability and Security (QRS), Macau, 2020, pp. 508–519, doi: 10.1109/QRS51102.2020.00071.

[14] M. T. Shahid and H. Debar, “CVSS-BERT: Explainable natural language processing to determine the severity of a computer security vulnerability from its description,” Information Systems Frontiers, vol. 24, no. 5, pp. 1123–1136, 2022, doi: 10.1007/s10796-021-10173-x.

[15] S. Singh, R. Verma, and A. Sinha, “Cyber security vulnerability detection using natural language processing,” in Proc. 2022 Int. Conf. Machine Learning and Cybernetics (ICMLC), 2022, pp. 411–416, doi: 10.1109/ICMLC55679.2022.9876543.

[16] A. Marchiori, L. Bertholdo, and E. Viegas, “Large language models for CVSS score prediction,” in Proc. IEEE Int. Conf. Cyber Security and Resilience (CSR), Venice, Italy, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10721432

[17] M. Tiwari, “Vulnerability severity prediction: A machine learning approach using CVSS and BERT-based classifier,” in Proc. 2025 Int. Conf. Artificial Intelligence (ICAI), 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10719690

[18] D. Mandal and F. Kösesoy, “Vulnerability prediction using machine learning classifiers and textual features,” in Proc. IEEE Int. Conf. Machine Learning and Applications (ICMLA), 2021, pp. 958–963, doi: 10.1109/ICMLA52953.2021.00165.

[19] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. NAACL-HLT, 2019.

[20] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in Proc. EMNLP-IJCNLP, 2019.

[21] HuggingFace, “all-mpnet-base-v2 model card,” 2021. [Online]. Available: https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[22] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge: Cambridge Univ. Press, 2008.

[23] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.

[24] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd ed. O’Reilly Media, 2019.

[25] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, 2nd ed. Springer, 2009.

[26] C. M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.

[27] K. P. Murphy, Machine Learning: A Probabilistic Perspective. MIT Press, 2012.

[28] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

ΠΑΡΑΡΤΗΜΑ Α : Συλλογή Δεδομένων

```
#!/usr/bin/env python3
from __future__ import annotations
import os
import time
import re
import json
from pathlib import Path
from typing import Dict, List, Optional
import requests

API_URL = "https://services.nvd.nist.gov/rest/json/cves/2.0"
YEAR_RE = re.compile(r"CVE-(\d{4})-\d+")
DEFAULT_PAGE_SIZE = 2000
DEFAULT_PAUSE = 0.65
def _fetch_page(start_index: int, page_size: int, api_key: str) -> Dict:
    headers = {"Accept": "application/json", "apiKey": api_key}
    params = {"resultsPerPage": page_size, "startIndex": start_index}
    resp = requests.get(API_URL, headers=headers, params=params, timeout=60)
    if resp.status_code == 429:
        time.sleep(6)
        resp = requests.get(API_URL, headers=headers, params=params, timeout=60)
    resp.raise_for_status()
    return resp.json()
def fetch_all_cves(api_key: Optional[str] = None,
                   page_size: int = DEFAULT_PAGE_SIZE,
                   pause: float = DEFAULT_PAUSE) -> List[Dict]:
    key = api_key or os.getenv("NVD_API_KEY")
    if not key or "YOUR_" in key:
        raise RuntimeError("NVD_API_KEY is missing.")

    first = _fetch_page(0, page_size, key)
    total = int(first.get("totalResults", 0))
    items = list(first.get("vulnerabilities", []))

    idx = len(items)
    while idx < total:
        time.sleep(pause)
        page = _fetch_page(idx, page_size, key)
        batch = page.get("vulnerabilities", [])
        if not batch:
            break
        items.extend(batch)
        idx += len(batch)
    return items
def split_cves_by_year(cve_list: List[Dict]) -> Dict[str, List[Dict]]:
    grouped: Dict[str, List[Dict]] = {}
    for vuln in cve_list:
        cve_id = vuln.get("cve", {}).get("id", "")
        m = YEAR_RE.match(cve_id)
        year = m.group(1) if m else "unknown"
        grouped.setdefault(year, []).append(vuln)
    return grouped
def save_cves_by_year(cves_by_year: Dict[str, List[Dict]], output_dir: str | Path) -> Dict[str,
str]:
    out = Path(output_dir)
    out.mkdir(parents=True, exist_ok=True)
    saved: Dict[str, str] = {}
    for year, entries in sorted(cves_by_year.items()):
        p = out / f"cves_{year}.json"
        with p.open("w", encoding="utf-8") as f:
            json.dump(entries, f, ensure_ascii=False, indent=2)
        saved[year] = str(p)
    return saved
```

```
76. def download_and_save(output_dir: str | Path,
77.                         api_key: Optional[str] = None,
78.                         page_size: int = DEFAULT_PAGE_SIZE,
79.                         pause: float = DEFAULT_PAUSE) -> Dict[str, str]:
80.     all_cves = fetch_all_cves(api_key=api_key, page_size=page_size, pause=pause)
81.     grouped = split_cves_by_year(all_cves)
82.     return save_cves_by_year(grouped, output_dir)
```

ΠΑΡΑΡΤΗΜΑ Β : Δημιουργία Συνόλου Δεδομένων

```python
#!/usr/bin/env python3
from __future__ import annotations

import csv
import json
import re
from pathlib import Path
from typing import Dict, Iterable, List, Optional

_WS = re.compile(r"\s+")

FIELDS = [
    "cve_id",
    "description",
    "baseSeverity",
    "attackVector",
    "attackComplexity",
    "privilegesRequired",
    "userInteraction",
    "scope",
    "confidentialityImpact",
    "integrityImpact",
    "availabilityImpact",
]


def _choose_cvss3(metrics: Dict) -> Optional[Dict]:
    chosen = None
    fallback = None
    for key, arr in (metrics or {}).items():
        if not isinstance(arr, list) or not arr:
            continue
        m = arr[0]
        cd = (m.get("cvssData") or {})
        ver = cd.get("version")
        base_sev = m.get("baseSeverity") or cd.get("baseSeverity")
        if ver == "3.1":
            chosen = {**cd, "baseSeverity": base_sev}
            break
        if ver == "3.0" and fallback is None:
            fallback = {**cd, "baseSeverity": base_sev}
    return chosen or fallback


def _first_description(cve_block: Dict) -> str:
    for d in cve_block.get("descriptions", []):
        val = (d.get("value") or "").strip()
        if val:
            return _WS.sub(" ", val)
    return ""


def iter_cves_from_files(files: Iterable[Path]) -> Iterable[Dict]:
    for p in files:
        data = json.loads(p.read_text(encoding="utf-8"))
        entries = (
            data.get("vulnerabilities")
            if isinstance(data, dict) and data.get("vulnerabilities") is not None
            else data
        )
        if not isinstance(entries, list):
            continue
        for vuln in entries:
            yield vuln


```

```
def jsons_to_cvss3x_csv(input_dir: str | Path, output_csv: str | Path) -> None:
    input_dir = Path(input_dir)
    output_csv = Path(output_csv)

    files = sorted(
        p for p in input_dir.iterdir()
        if p.is_file() and p.suffix.lower() == ".json" and p.name.lower().startswith("cves")
    )

    seen: set[str] = set()
    output_csv.parent.mkdir(parents=True, exist_ok=True)

    with output_csv.open("w", newline="", encoding="utf-8") as f:
        w = csv.DictWriter(f, fieldnames=FIELDS)
        w.writeheader()

        for vuln in iter_cves_from_files(files):
            cve_block = vuln.get("cve", {})
            cve_id = cve_block.get("id")
            if not cve_id or cve_id in seen:
                continue

            metrics = cve_block.get("metrics", {})
            cvss = _choose_cvss3(metrics)
            if not cvss:
                continue

            w.writerow({
                "cve_id": cve_id,
                "description": _first_description(cve_block),
                "baseSeverity": cvss.get("baseSeverity", ""),
                "attackVector": cvss.get("attackVector", ""),
                "attackComplexity": cvss.get("attackComplexity", ""),
                "privilegesRequired": cvss.get("privilegesRequired", ""),
                "userInteraction": cvss.get("userInteraction", ""),
                "scope": cvss.get("scope", ""),
                "confidentialityImpact": cvss.get("confidentialityImpact", ""),
                "integrityImpact": cvss.get("integrityImpact", ""),
                "availabilityImpact": cvss.get("availabilityImpact", ""),
            })
            seen.add(cve_id)

```

ΠΑΡΑΡΤΗΜΑ C Κώδικας Πειράματος E1

```
import os
import time
from sklearn.utils.class_weight import compute_sample_weight
from sklearn.multioutput import MultiOutputClassifier
import xgboost as xgb
from sklearn.metrics import classification_report

# Initialize the base XGBoost model with specified parameters
base_estimator = xgb.XGBClassifier(
    max_depth=4, learning_rate=0.1, tree_method='hist', random_state=42, eval_metric='mlogloss'
)

# Initialize MultiOutputClassifier with the base XGBoost model
model = MultiOutputClassifier(base_estimator)

start = time.time()

model.fit(X_train, y_train)

# Now, iterate through the initialized estimators and refit with sample weights
print("Refitting models with sample weights...")
for i in range(y_train.shape[1]):
    print(f"Refitting model for output {i+1}/{y_train.shape[1]} ({labels.columns[i]})")
    weights = compute_sample_weight('balanced', y_train[:, i])
    model.estimators_[i].fit(X_train, y_train[:, i], sample_weight=weights)

print(f"Training time: {(time.time()-start)/60:.2f} min")

# Make predictions on the validation set
y_pred = model.predict(X_val)

# Create a directory to store the evaluation results
out_dir = 'results/tfidf_unigrams_time_split'
os.makedirs(out_dir, exist_ok=True)

reports = []
macro_f1_scores = {}
accuracy_scores = {}

# Evaluate the model for each target label
for i, col in enumerate(labels.columns):
    rpt = classification_report(y_val[:, i], y_pred[:, i], target_names=encoders[col].classes_,
output_dict=True)
    print(f"=== Val Report: {col} ===")
                        print(classification_report(y_val[:,     i],     y_pred[:,     i],
target_names=encoders[col].classes_))

        reports.append(f"===  {col}  ===\n{classification_report(y_val[:,  i],  y_pred[:,  i],
target_names=encoders[col].classes_)}\n")

    # Store macro F1 and accuracy
    macro_f1_scores[col] = rpt['macro avg']['f1-score']
    accuracy_scores[col] = rpt['accuracy']

# Write the classification reports to a file
with open(f"{out_dir}/evaluation.txt", 'w') as f:
    f.writelines(reports)
# Print a summary of the macro F1 and accuracy scores
print("\n=== Summary of Macro F1 and Accuracy ===")
print("Macro F1 Scores:")
for col, score in macro_f1_scores.items():
    print(f"{col}: {score:.4f}")
```

```
61.
62. print("\nAccuracy Scores:")
63. for col, score in accuracy_scores.items():
64.     print(f"{col}: {score:.4f}")
```

ΠΑΡΑΡΤΗΜΑ D Κώδικας Πειράματος E2

```
 1. # 1. Initialize a TfidfVectorizer with ngram_range=(1, 2)
 2. bigram_vectorizer = TfidfVectorizer(
 3.     ngram_range=(1, 2),  # Include bigrams
 4.     min_df=2,
 5.     max_df=0.95,
 6.     max_features=15000,
 7.     sublinear_tf=True,
 8.     norm="l2",
 9.     stop_words=None
10. )
11.
12. # 2. Fit the new TF-IDF vectorizer to the lemmas data and transform
13. print("Fitting and transforming TF-IDF with unigrams and bigrams...")
14. X_tfidf_bigrams = bigram_vectorizer.fit_transform(lemmas)
15. print(f"Shape of X_tfidf_bigrams: {X_tfidf_bigrams.shape}")
16.
17. # 3. Split the X_tfidf_bigrams matrix using the same indices
18. X_train_bigrams = X_tfidf_bigrams[train_idx]
19. X_val_bigrams   = X_tfidf_bigrams[val_idx]
20. X_test_bigrams  = X_tfidf_bigrams[test_idx]
21.
22. print(f"Training set size (bigrams): {X_train_bigrams.shape[0]} samples")
23. print(f"Validation set size (bigrams): {X_val_bigrams.shape[0]} samples")
24. print(f"Test set size (bigrams): {X_test_bigrams.shape[0]} samples")
```

```python
import os
import time
from sklearn.utils.class_weight import compute_sample_weight
from sklearn.multioutput import MultiOutputClassifier
import xgboost as xgb
from sklearn.metrics import classification_report

# 4. Initialize a new MultiOutputClassifier with the same base estimator
base_estimator_bigrams = xgb.XGBClassifier(
    max_depth=4, learning_rate=0.1, tree_method='hist', random_state=42, eval_metric='mlogloss'
)
model_bigrams = MultiOutputClassifier(base_estimator_bigrams)

start = time.time()

# 5. Fit the MultiOutputClassifier to X_train_bigrams and y_train (initial fit)
print("Fitting MultiOutputClassifier with bigram features...")
model_bigrams.fit(X_train_bigrams, y_train)

# 6. Iterate through the estimators and refit with sample weights
print("Refitting models with sample weights...")
for i in range(y_train.shape[1]):
    print(f"Refitting model for output {i+1}/{y_train.shape[1]} ({labels.columns[i]})")
    weights = compute_sample_weight('balanced', y_train[:, i])
    model_bigrams.estimators_[i].fit(X_train_bigrams, y_train[:, i], sample_weight=weights)

print(f"Training time (bigrams): {(time.time()-start)/60:.2f} min")

# 7. Make predictions on X_val_bigrams
y_pred_bigrams = model_bigrams.predict(X_val_bigrams)

# 8. Create a new directory for the results
out_dir_bigrams = 'results/tfidf_unibi_time_split'
os.makedirs(out_dir_bigrams, exist_ok=True)

reports_bigrams = []
macro_f1_scores_bigrams = {}
accuracy_scores_bigrams = {}


# 9. Generate and print classification reports for each target label
print("\n=== Evaluation Reports (Bigrams) ===")
for i, col in enumerate(labels.columns):
            rpt_bigrams = classification_report(y_val[:, i], y_pred_bigrams[:, i],
target_names=encoders[col].classes_, output_dict=True)
    print(f"=== Val Report: {col} (Bigrams) ===")
                    print(classification_report(y_val[:, i], y_pred_bigrams[:, i],
target_names=encoders[col].classes_))

      reports_bigrams.append(f"=== {col} (Bigrams) ===\n{classification_report(y_val[:, i],
y_pred_bigrams[:, i], target_names=encoders[col].classes_)}\n")

    # Store macro F1 and accuracy
    macro_f1_scores_bigrams[col] = rpt_bigrams['macro avg']['f1-score']
    accuracy_scores_bigrams[col] = rpt_bigrams['accuracy']

# 10. Save the classification reports to a file
with open(f"{out_dir_bigrams}/evaluation.txt", 'w') as f:
    f.writelines(reports_bigrams)

# 11. Print a summary of the macro F1 and accuracy scores
print("\n=== Summary of Macro F1 and Accuracy (Bigrams) ===")
print("Macro F1 Scores:")
for col, score in macro_f1_scores_bigrams.items():
    print(f"{col}: {score:.4f}")

print("\nAccuracy Scores:")
for col, score in accuracy_scores_bigrams.items():
    print(f"{col}: {score:.4f}")
```

ΠΑΡΑΡΤΗΜΑ Ε Κώδικας Πειράματος Ε3

```
import os
import time
from sklearn.utils.class_weight import compute_sample_weight
from sklearn.multioutput import MultiOutputClassifier
import xgboost as xgb
from sklearn.metrics import classification_report

# 4. Initialize a new MultiOutputClassifier with the same base estimator
base_estimator_bigrams = xgb.XGBClassifier(
    max_depth=4, learning_rate=0.1, tree_method='hist', random_state=42, eval_metric='mlogloss'
)
model_bigrams = MultiOutputClassifier(base_estimator_bigrams)

start = time.time()

# 5. Fit the MultiOutputClassifier to X_train_bigrams and y_train (initial fit)
print("Fitting MultiOutputClassifier with bigram features...")
model_bigrams.fit(X_train_bigrams, y_train)

# 6. Iterate through the estimators and refit with sample weights
print("Refitting models with sample weights...")
for i in range(y_train.shape[1]):
    print(f"Refitting model for output {i+1}/{y_train.shape[1]} ({labels.columns[i]})")
    weights = compute_sample_weight('balanced', y_train[:, i])
    model_bigrams.estimators_[i].fit(X_train_bigrams, y_train[:, i], sample_weight=weights)

print(f"Training time (bigrams): {(time.time()-start)/60:.2f} min")
# 7. Make predictions on X_val_bigrams
y_pred_bigrams = model_bigrams.predict(X_val_bigrams)
# 8. Create a new directory for the results
out_dir_bigrams = 'results/tfidf_unibi_time_split'
os.makedirs(out_dir_bigrams, exist_ok=True)

reports_bigrams = []
macro_f1_scores_bigrams = {}
accuracy_scores_bigrams = {}

# 9. Generate and print classification reports for each target label
print("\n=== Evaluation Reports (Bigrams) ===")
for i, col in enumerate(labels.columns):
            rpt_bigrams = classification_report(y_val[:, i], y_pred_bigrams[:, i],
target_names=encoders[col].classes_, output_dict=True)
    print(f"=== Val Report: {col} (Bigrams) ===")
                    print(classification_report(y_val[:, i], y_pred_bigrams[:, i],
target_names=encoders[col].classes_))

      reports_bigrams.append(f"=== {col} (Bigrams) ===\n{classification_report(y_val[:, i],
y_pred_bigrams[:, i], target_names=encoders[col].classes_)}\n")

    # Store macro F1 and accuracy
    macro_f1_scores_bigrams[col] = rpt_bigrams['macro avg']['f1-score']
    accuracy_scores_bigrams[col] = rpt_bigrams['accuracy']

# 10. Save the classification reports to a file
with open(f"{out_dir_bigrams}/evaluation.txt", 'w') as f:
    f.writelines(reports_bigrams)

# 11. Print a summary of the macro F1 and accuracy scores
print("\n=== Summary of Macro F1 and Accuracy (Bigrams) ===")
print("Macro F1 Scores:")
for col, score in macro_f1_scores_bigrams.items():
    print(f"{col}: {score:.4f}")

print("\nAccuracy Scores:")
for col, score in accuracy_scores_bigrams.items():
    print(f"{col}: {score:.4f}")
```

```python
import os
import time
from sklearn.utils.class_weight import compute_sample_weight
from sklearn.multioutput import MultiOutputClassifier
import xgboost as xgb
from sklearn.metrics import classification_report

# 4. Initialize a new MultiOutputClassifier with the same base estimator
base_estimator_bigrams = xgb.XGBClassifier(
    max_depth=4, learning_rate=0.1, tree_method='hist', random_state=42, eval_metric='mlogloss'
)
model_bigrams = MultiOutputClassifier(base_estimator_bigrams)

start = time.time()

# 5. Fit the MultiOutputClassifier to X_train_bigrams and y_train (initial fit)
print("Fitting MultiOutputClassifier with bigram features...")
model_bigrams.fit(X_train_bigrams, y_train)

# 6. Iterate through the estimators and refit with sample weights
print("Refitting models with sample weights...")
for i in range(y_train.shape[1]):
    print(f"Refitting model for output {i+1}/{y_train.shape[1]} ({labels.columns[i]})")
    weights = compute_sample_weight('balanced', y_train[:, i])
    model_bigrams.estimators_[i].fit(X_train_bigrams, y_train[:, i], sample_weight=weights)

print(f"Training time (bigrams): {(time.time()-start)/60:.2f} min")

# 7. Make predictions on X_val_bigrams
y_pred_bigrams = model_bigrams.predict(X_val_bigrams)

# 8. Create a new directory for the results
out_dir_bigrams = 'results/tfidf_unibi_time_split'
os.makedirs(out_dir_bigrams, exist_ok=True)

reports_bigrams = []
macro_f1_scores_bigrams = {}
accuracy_scores_bigrams = {}

# 9. Generate and print classification reports for each target label
print("\n=== Evaluation Reports (Bigrams) ===")
for i, col in enumerate(labels.columns):
            rpt_bigrams = classification_report(y_val[:, i], y_pred_bigrams[:, i],
target_names=encoders[col].classes_, output_dict=True)
    print(f"=== Val Report: {col} (Bigrams) ===")
                    print(classification_report(y_val[:, i], y_pred_bigrams[:, i],
target_names=encoders[col].classes_))

      reports_bigrams.append(f"=== {col} (Bigrams) ===\n{classification_report(y_val[:, i],
y_pred_bigrams[:, i], target_names=encoders[col].classes_)}\n")

    # Store macro F1 and accuracy
    macro_f1_scores_bigrams[col] = rpt_bigrams['macro avg']['f1-score']
    accuracy_scores_bigrams[col] = rpt_bigrams['accuracy']

# 10. Save the classification reports to a file
with open(f"{out_dir_bigrams}/evaluation.txt", 'w') as f:
    f.writelines(reports_bigrams)

# 11. Print a summary of the macro F1 and accuracy scores
print("\n=== Summary of Macro F1 and Accuracy (Bigrams) ===")
print("Macro F1 Scores:")
for col, score in macro_f1_scores_bigrams.items():
    print(f"{col}: {score:.4f}")

print("\nAccuracy Scores:")
for col, score in accuracy_scores_bigrams.items():
    print(f"{col}: {score:.4f}")
```

ΠΑΡΑΡΤΗΜΑ F Κώδικας Πειράματος Ε4

```
import os
import time
import numpy as np
from sentence_transformers import SentenceTransformer
from sklearn.multioutput import MultiOutputClassifier
import xgboost as xgb
from sklearn.utils.class_weight import compute_sample_weight
from sklearn.metrics import classification_report

# 1. & 2. Load the pre-trained Sentence-BERT model
print("Loading Sentence-BERT model 'all-MiniLM-L6-v2'...")
start_load_sbert = time.time()
sbert_model = SentenceTransformer('all-MiniLM-L6-v2')
print(f"Sentence-BERT model loaded in {(time.time()-start_load_sbert):.2f} seconds.")

# 3. Generate embeddings for the preprocessed text data (lemmas)
print("Generating Sentence-BERT embeddings...")
start_embed_sbert = time.time()
X_sbert = sbert_model.encode(lemmas, show_progress_bar=True)
print(f"Sentence-BERT embeddings generated in {(time.time()-start_embed_sbert)/60:.2f} minutes.")
print(f"Shape of X_sbert embeddings: {X_sbert.shape}")

# 4. Split the generated embeddings using the same indices
X_train_sbert = X_sbert[train_idx]
X_val_sbert   = X_sbert[val_idx]
X_test_sbert  = X_sbert[test_idx]

print(f"Training set size (SBERT): {X_train_sbert.shape[0]} samples")
print(f"Validation set size (SBERT): {X_val_sbert.shape[0]} samples")
print(f"Test set size (SBERT): {X_test_sbert.shape[0]} samples")

# 5. Initialize MultiOutputClassifier with XGBoost base estimator
base_estimator_sbert = xgb.XGBClassifier(
    max_depth=4, learning_rate=0.1, tree_method='hist', random_state=42, eval_metric='mlogloss'
)
model_sbert = MultiOutputClassifier(base_estimator_sbert)

start_sbert_train = time.time()

# 6. Fit the MultiOutputClassifier to SBERT training features (initial fit)
print("\nFitting MultiOutputClassifier with SBERT features...")
model_sbert.fit(X_train_sbert, y_train)

# 7. Iterate through the estimators and refit with sample weights
print("Refitting models with sample weights (SBERT)...")
for i in range(y_train.shape[1]):
    print(f"Refitting model for output {i+1}/{y_train.shape[1]} ({labels.columns[i]})")
    weights = compute_sample_weight('balanced', y_train[:, i])
    model_sbert.estimators_[i].fit(X_train_sbert, y_train[:, i], sample_weight=weights)

print(f"Training time (SBERT): {(time.time()-start_sbert_train)/60:.2f} min")

# 8. Make predictions on the SBERT validation features
y_pred_sbert = model_sbert.predict(X_val_sbert)

# 9. Create a directory to store the evaluation results
out_dir_sbert = 'results/sbert_minilm_time_split'
os.makedirs(out_dir_sbert, exist_ok=True)

reports_sbert = []
macro_f1_scores_sbert = {}
accuracy_scores_sbert = {}

```

```
# 10. & 11. Generate and print classification reports for each target label and store them
print("\n=== Evaluation Reports (SBERT) ===")
for i, col in enumerate(labels.columns):
                rpt_sbert    =    classification_report(y_val[:,    i],    y_pred_sbert[:,    i],
target_names=encoders[col].classes_, output_dict=True)
    print(f"=== Val Report: {col} (SBERT) ===")
                        print(classification_report(y_val[:,     i],     y_pred_sbert[:,     i],
target_names=encoders[col].classes_))

         reports_sbert.append(f"===  {col}  (SBERT)  ===\n{classification_report(y_val[:,  i],
y_pred_sbert[:, i], target_names=encoders[col].classes_)}\n")

    # 12. Store macro F1 and accuracy
    macro_f1_scores_sbert[col] = rpt_sbert['macro avg']['f1-score']
    accuracy_scores_sbert[col] = rpt_sbert['accuracy']

# 13. Save the classification reports to a file
with open(f"{out_dir_sbert}/evaluation.txt", 'w') as f:
    f.writelines(reports_sbert)

# 14. Print a summary of the macro F1 and accuracy scores
print("\n=== Summary of Macro F1 and Accuracy (SBERT) ===")
print("Macro F1 Scores:")
for col, score in macro_f1_scores_sbert.items():
    print(f"{col}: {score:.4f}")

print("\nAccuracy Scores:")
for col, score in accuracy_scores_sbert.items():
    print(f"{col}: {score:.4f}")

```

ΠΑΡΑΡΤΗΜΑ Ε Κώδικας Πειράματος Ε5

```
import os
import time
import numpy as np
import xgboost as xgb
from sklearn.multioutput import MultiOutputClassifier
from sklearn.utils.class_weight import compute_sample_weight
from sklearn.metrics import classification_report
from scipy.sparse import hstack, csr_matrix
# 1. Horizontally stack the TF-IDF (uni+bi) training features and SBERT training embeddings
X_train_combined = hstack([X_train_bigrams, csr_matrix(X_train_sbert)])
print(f"Shape of combined training features: {X_train_combined.shape}")

# 2. Horizontally stack the TF-IDF (uni+bi) validation features and SBERT validation embeddings
X_val_combined = hstack([X_val_bigrams, csr_matrix(X_val_sbert)])
print(f"Shape of combined validation features: {X_val_combined.shape}")

# 3. Initialize MultiOutputClassifier with XGBoost base estimator
base_estimator_combined = xgb.XGBClassifier(
    max_depth=4, learning_rate=0.1, tree_method='hist', random_state=42, eval_metric='mlogloss'
)
model_combined = MultiOutputClassifier(base_estimator_combined)

start_combined_train = time.time()

# 4. Fit the MultiOutputClassifier to the combined training features (initial fit)
print("\nFitting MultiOutputClassifier with combined features...")
model_combined.fit(X_train_combined, y_train)
print("Refitting models with sample weights (Combined)...")
for i in range(y_train.shape[1]):
    print(f"Refitting model for output {i+1}/{y_train.shape[1]} ({labels.columns[i]})")
    weights = compute_sample_weight('balanced', y_train[:, i])
    model_combined.estimators_[i].fit(X_train_combined, y_train[:, i], sample_weight=weights)
print(f"Training time (Combined): {(time.time()-start_combined_train)/60:.2f} min")
# 6. Make predictions on the combined validation features
y_pred_combined = model_combined.predict(X_val_combined)
out_dir_combined = 'results/combined_tfidf_sbert_time_split'
os.makedirs(out_dir_combined, exist_ok=True)

reports_combined = []
macro_f1_scores_combined = {}
accuracy_scores_combined = {}
print("\n=== Evaluation Reports (Combined) ===")
for i, col in enumerate(labels.columns):
            rpt_combined = classification_report(y_val[:, i], y_pred_combined[:, i],
target_names=encoders[col].classes_, output_dict=True)
    print(f"=== Val Report: {col} (Combined) ===")
                    print(classification_report(y_val[:, i], y_pred_combined[:, i],
target_names=encoders[col].classes_))

     reports_combined.append(f"=== {col} (Combined) ===\n{classification_report(y_val[:, i],
y_pred_combined[:, i], target_names=encoders[col].classes_)}\n")

    # 11. Extract and store macro F1 and accuracy
    macro_f1_scores_combined[col] = rpt_combined['macro avg']['f1-score']
    accuracy_scores_combined[col] = rpt_combined['accuracy']
# 12. Write the classification reports to a file
with open(f"{out_dir_combined}/evaluation.txt", 'w') as f:
    f.writelines(reports_combined)
# 13. Print a summary of the macro F1 and accuracy scores
print("\n=== Summary of Macro F1 and Accuracy (Combined) ===")
print("Macro F1 Scores:")
for col, score in macro_f1_scores_combined.items():
    print(f"{col}: {score:.4f}")
print("\nAccuracy Scores:")
for col, score in accuracy_scores_combined.items():
    print(f"{col}: {score:.4f}")
```

ΠΑΡΑΡΤΗΜΑ Ε Κώδικας Πειράματος Ε6

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, f1_score
import numpy as np
import os
import joblib
import time
from sklearn.utils import resample
from statsmodels.stats.contingency_tables import mcnemar

# 1. Perform a stratified train-test split
# Note: Stratifying multi-output y is complex. We will stratify based on the first output (attackVector)
# as a practical approach for this multi-output problem within the scope.
# A test size equal to the validation set size from the time-aware split (21007 samples) is used.
test_size_stratified = 21007 # This is the size of val_idx and test_idx from previous splits
X_train_stratified, X_test_stratified, y_train_stratified, y_test_stratified = train_test_split(
    X_tfidf, y, test_size=test_size_stratified, random_state=42, stratify=y[:, 0] # Stratify by the first column (attackVector)
)

# Check the shapes of the stratified split
print(f"Stratified training set size: {X_train_stratified.shape[0]} samples")
print(f"Stratified test set size: {X_test_stratified.shape[0]} samples")

# 3. Load the best performing TF-IDF model (E1 or E2)
# Compare macro_f1_scores from E1 (macro_f1_scores) and E2 (macro_f1_scores_bigrams)
avg_macro_f1_e1 = np.mean(list(macro_f1_scores.values()))
avg_macro_f1_e2 = np.mean(list(macro_f1_scores_bigrams.values()))

if avg_macro_f1_e2 > avg_macro_f1_e1:
    print("Experiment 2 (uni+bi-grams) had better average macro F1. Loading model_bigrams.")
    best_tfidf_model = model_bigrams
    # Need the bigram features for prediction with this model
    X_test_tfidf_best = X_tfidf_bigrams[X_test_stratified.indices] # This is incorrect, need to split X_tfidf_bigrams directly
    # Re-doing stratified split for X_tfidf_bigrams based on y[:,0]
    _, X_test_tfidf_best, _, _ = train_test_split(
        X_tfidf_bigrams, y, test_size=test_size_stratified, random_state=42, stratify=y[:, 0]
    )

else:
    print("Experiment 1 (unigrams) had better or equal average macro F1. Loading model.")
    best_tfidf_model = model
    # Need the unigram features for prediction with this model
    # X_test_tfidf_best is already X_test_stratified from the initial split
    X_test_tfidf_best = X_test_stratified


# 4. Load the SBERT model from Experiment 4 (model_sbert)
# X_sbert was generated in Experiment 4. Need to split it using the same stratified indices.
_, X_test_sbert_stratified, _, _ = train_test_split(
    X_sbert, y, test_size=test_size_stratified, random_state=42, stratify=y[:, 0]
)
sbert_model_loaded = model_sbert # model_sbert was trained in Exp 4

# 5. Make predictions on the stratified test data
print("Making predictions with best TF-IDF model on stratified test set...")
y_pred_tfidf_stratified = best_tfidf_model.predict(X_test_tfidf_best)

print("Making predictions with SBERT model on stratified test set...")
y_pred_sbert_stratified = sbert_model_loaded.predict(X_test_sbert_stratified)

print("Predictions made. Proceeding to evaluation and significance testing.")
```

```
# 6. For each of the 8 CVSS labels:
macro_f1_results = {}
confidence_intervals = {}
mcnemar_results = {}

n_iterations = 1000 # Number of bootstrap iterations
alpha = 0.05        # For 95% confidence interval

out_dir_stratified_eval = 'results/comparative_stratified_eval'
os.makedirs(out_dir_stratified_eval, exist_ok=True)

eval_summary = []

print("\n=== Evaluating models on Stratified Test Set ===")

for i, col in enumerate(labels.columns):
    print(f"\nEvaluating {col}...")

    y_true_col = y_test_stratified[:, i]
    y_pred_tfidf_col = y_pred_tfidf_stratified[:, i]
    y_pred_sbert_col = y_pred_sbert_stratified[:, i]
    target_names = encoders[col].classes_

    # a. Calculate and store the macro F1 score
    macro_f1_tfidf = f1_score(y_true_col, y_pred_tfidf_col, average='macro')
    macro_f1_sbert = f1_score(y_true_col, y_pred_sbert_col, average='macro')

    macro_f1_results[col] = {
        'TF-IDF': macro_f1_tfidf,
        'SBERT': macro_f1_sbert
    }
    print(f"  Macro F1 (TF-IDF): {macro_f1_tfidf:.4f}")
    print(f"  Macro F1 (SBERT): {macro_f1_sbert:.4f}")

    # b. Implement bootstrapping for confidence intervals
    tfidf_macro_f1_scores = []
    sbert_macro_f1_scores = []

    # Create a list of indices to resample
    indices = np.arange(len(y_true_col))

    for _ in tqdm(range(n_iterations), desc=f"  Bootstrapping {col}"):
        # Resample indices with replacement
            bootstrap_indices = resample(indices, replace=True, n_samples=len(indices),
random_state=np.random.randint(1000000)) # Use different random state for each resample

        # Calculate macro F1 on the resampled data
                    tfidf_macro_f1_scores.append(f1_score(y_true_col[bootstrap_indices],
y_pred_tfidf_col[bootstrap_indices], average='macro'))
                    sbert_macro_f1_scores.append(f1_score(y_true_col[bootstrap_indices],
y_pred_sbert_col[bootstrap_indices], average='macro'))

    # Calculate confidence intervals from the distribution of bootstrap scores
    lower_tfidf = np.percentile(tfidf_macro_f1_scores, (alpha/2) * 100)
    upper_tfidf = np.percentile(tfidf_macro_f1_scores, 100 - (alpha/2) * 100)
    lower_sbert = np.percentile(sbert_macro_f1_scores, (alpha/2) * 100)
    upper_sbert = np.percentile(sbert_macro_f1_scores, 100 - (alpha/2) * 100)

    confidence_intervals[col] = {
        'TF-IDF': (lower_tfidf, upper_tfidf),
        'SBERT': (lower_sbert, upper_sbert)
    }
    print(f"  95% CI (TF-IDF): ({lower_tfidf:.4f}, {upper_tfidf:.4f})")
    print(f"  95% CI (SBERT): ({lower_sbert:.4f}, {upper_sbert:.4f})")
     print("  Skipping McNemar's test due to multi-class multi-output complexity.")
    mcnemar_results[col] = "Skipped" # Record that it was skipped

```

```
    # Append results to summary
    eval_summary.append(f"=== {col} ===")
    eval_summary.append(f"Macro F1 (TF-IDF): {macro_f1_tfidf:.4f}")
    eval_summary.append(f"95% CI (TF-IDF): ({lower_tfidf:.4f}, {upper_tfidf:.4f})")
    eval_summary.append(f"Macro F1 (SBERT): {macro_f1_sbert:.4f}")
    eval_summary.append(f"95% CI (SBERT): ({lower_sbert:.4f}, {upper_sbert:.4f})")
    eval_summary.append(f"McNemar's Test: {mcnemar_results[col]}")
    eval_summary.append("-" * 20)


# 7. Print or display the macro F1 scores with their confidence intervals
print("\n=== Summary of Stratified Test Set Evaluation ===")
print("Macro F1 Scores and 95% Confidence Intervals:")
for col in labels.columns:
    print(f"  {col}:")
    print(f"    TF-IDF: {macro_f1_results[col]['TF-IDF']:.4f} ({confidence_intervals[col]['TF-IDF'][0]:.4f}, {confidence_intervals[col]['TF-IDF'][1]:.4f})")
    print(f"    SBERT: {macro_f1_results[col]['SBERT']:.4f} ({confidence_intervals[col]['SBERT'][0]:.4f}, {confidence_intervals[col]['SBERT'][1]:.4f})")

# 8. Report the results of any performed significance tests (Skipped McNemar)
print("\n=== Significance Testing ===")
print("McNemar's Test was skipped due to the multi-class multi-output nature of the problem.")
print("Confidence intervals provide a basis for comparing model performance.")


# 9. Store the evaluation results in a file
with open(f"{out_dir_stratified_eval}/stratified_evaluation_summary.txt", 'w') as f:
    f.write("\n".join(eval_summary))

print(f"\nEvaluation summary saved to {out_dir_stratified_eval}/stratified_evaluation_summary.txt")
```

```
import pandas as pd
import numpy as np
import os

# Define experiment names for clarity
exp_names = {
    'E1 (TF-IDF Uni, Lemma+NoStop)': (macro_f1_scores, accuracy_scores), # This is E1 based on initial setup, which used Lemma+NoStop
    'E2 (TF-IDF Uni+Bi, Lemma+NoStop)': (macro_f1_scores_bigrams, accuracy_scores_bigrams), # E2 used Lemma+NoStop
    'E3 (TF-IDF Uni, NoLemma+NoStop)': (macro_f1_scores_no_lemma_nostop, accuracy_scores_no_lemma_nostop),
    'E3 (TF-IDF Uni, Lemma+NoStop)': (macro_f1_scores_lemma_nostop, accuracy_scores_lemma_nostop),
    'E4 (SBERT)': (macro_f1_scores_sbert, accuracy_scores_sbert),
    'E5 (Combined TF-IDF+SBERT)': (macro_f1_scores_combined, accuracy_scores_combined),
}

# 1. Compile macro F1 and accuracy scores from validation set evaluations
results_list = []
for exp_name, (macro_f1_dict, accuracy_dict) in exp_names.items():
    for col in labels.columns:
        results_list.append({
            'Experiment': exp_name,
            'CVSS Label': col,
            'Metric': 'Macro F1 (Val)',
            'Score': macro_f1_dict.get(col, np.nan) # Use .get for safety
        })
        results_list.append({
            'Experiment': exp_name,
            'CVSS Label': col,
            'Metric': 'Accuracy (Val)',
            'Score': accuracy_dict.get(col, np.nan) # Use .get for safety
        })

# 2. Include macro F1 scores and CIs from stratified test set (E6)
# macro_f1_results and confidence_intervals are already available from E6
for col in labels.columns:
    # Add Macro F1 from stratified test set
    results_list.append({
        'Experiment': 'E6 TF-IDF (Stratified Test)',
        'CVSS Label': col,
        'Metric': 'Macro F1 (Stratified Test)',
        'Score': macro_f1_results.get(col, {}).get('TF-IDF', np.nan)
    })
    results_list.append({
        'Experiment': 'E6 SBERT (Stratified Test)',
        'CVSS Label': col,
        'Metric': 'Macro F1 (Stratified Test)',
        'Score': macro_f1_results.get(col, {}).get('SBERT', np.nan)
    })
    # Add 95% CI from stratified test set
    ci_tfidf = confidence_intervals.get(col, {}).get('TF-IDF', (np.nan, np.nan))
    ci_sbert = confidence_intervals.get(col, {}).get('SBERT', (np.nan, np.nan))

    results_list.append({
        'Experiment': 'E6 TF-IDF (Stratified Test)',
        'CVSS Label': col,
        'Metric': '95% CI (Stratified Test)',
        'Score': f"({ci_tfidf[0]:.4f}, {ci_tfidf[1]:.4f})" if not np.isnan(ci_tfidf[0]) else "N/A"
    })
    results_list.append({
        'Experiment': 'E6 SBERT (Stratified Test)',
        'CVSS Label': col,
        'Metric': '95% CI (Stratified Test)',
        'Score': f"({ci_sbert[0]:.4f}, {ci_sbert[1]:.4f})" if not np.isnan(ci_sbert[0]) else "N/A"
    })

```

```
65.
66. # 3. Create a structured summary (pandas DataFrame)
67. results_df = pd.DataFrame(results_list)
68.
69. # Pivot the table for better readability
70. results_pivot = results_df.pivot_table(
71.     index=['CVSS Label', 'Metric'],
72.     columns='Experiment',
73.     values='Score',
74.     aggfunc='first' # Use first as there's only one score per combination
75. )
76.
77. # Reorder columns for logical flow
78. ordered_cols = [
79.     'E1 (TF-IDF Uni, Lemma+NoStop)',
80.     'E2 (TF-IDF Uni+Bi, Lemma+NoStop)',
81.     'E3 (TF-IDF Uni, NoLemma+NoStop)',
82.     'E3 (TF-IDF Uni, Lemma+NoStop)',
83.     'E4 (SBERT)',
84.     'E5 (Combined TF-IDF+SBERT)',
85.     'E6 TF-IDF (Stratified Test)',
86.     'E6 SBERT (Stratified Test)'
87. ]
88. # Ensure all ordered_cols are actually in the DataFrame columns before reindexing
89. ordered_cols = [col for col in ordered_cols if col in results_pivot.columns]
90. results_pivot = results_pivot[ordered_cols]
91.
92. # Optional: Format numeric scores
93. def format_score(x):
94.     if isinstance(x, (float, np.floating)):
95.         return f"{x:.4f}"
96.     return x
97.
98. results_pivot = results_pivot.applymap(format_score)
99.
100.
101. print("=== Summary of Experimental Results ===")
102. display(results_pivot)
103.
104. # Save the pivot table
105. results_pivot.to_csv(f"{out_dir_summary}/experimental_results_summary.csv")
106.
107. # Save the analysis text
108. with open(f"{out_dir_summary}/experimental_results_analysis.txt", 'w') as f:
109.     f.write(analysis_text)
110.
111.         print(f"\nExperimental          results          summary          saved          to
{out_dir_summary}/experimental_results_summary.csv")
112.         print(f"Experimental          results          analysis          saved          to
{out_dir_summary}/experimental_results_analysis.txt")
```

ΠΑΡΑΡΤΗΜΑ G : Κώδικας Τελικού Μοντέλου

```
# Install dependencies
!pip install spacy xgboost scikit-learn pandas joblib matplotlib seaborn
!python -m spacy download en_core_web_sm

# Load dataset
import pandas as pd

df = pd.read_csv('cves_cvss3x.csv')
texts = df['description'].fillna('')

print(f"Loaded {df.shape[0]} rows and {df.shape[1]} columns.")
print(df['description'].head())

# Lemmatization
import spacy
from tqdm.notebook import tqdm

nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
tqdm.pandas()

def lemmatize_texts(texts):
    docs = nlp.pipe(texts, batch_size=64, n_process=-1)
    return [
            ' '.join(token.lemma_.lower() for token in doc if not token.is_punct and not token.is_space)
        for doc in tqdm(docs, total=len(texts), desc="Lemmatizing")
    ]

lemmas = lemmatize_texts(texts)
# TF-IDF Vectorization
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(max_features=10000)
X_tfidf = vectorizer.fit_transform(lemmas)

# Label encoding for multi-label targets
from sklearn.preprocessing import LabelEncoder

targets = ['attackVector', 'attackComplexity', 'privilegesRequired', 'userInteraction',
           'scope', 'confidentialityImpact', 'integrityImpact', 'availabilityImpact']

encoders = {col: LabelEncoder().fit(df[col].astype(str)) for col in targets}
y = pd.DataFrame({col: encoders[col].transform(df[col].astype(str)) for col in targets})

# Train/test split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)

# Train multi-output classifier using XGBoost
import xgboost as xgb
from sklearn.multioutput import MultiOutputClassifier

base_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric="mlogloss", verbosity=0)
model = MultiOutputClassifier(base_model)
model.fit(X_train, y_train)

# Evaluation
from sklearn.metrics import classification_report

y_pred = model.predict(X_test)
for i, col in enumerate(targets):
    print(f"\n=== {col} ===")
                              print(classification_report(y_test[col],      y_pred[:,      i],
target_names=encoders[col].classes_))

```

ΠΑΡΑΡΤΗΜΑ Η : Κώδικας εφαρμογής Flask

```python
# app.py
from flask import Flask, render_template, request, jsonify
from predict import predict_and_evaluate

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    csv_input = request.form['csv_text']
    try:
        results, stats = predict_and_evaluate(csv_input)
        return jsonify({
            "success": True,
            "results": results,
            "stats": stats
        })
    except Exception as e:
        return jsonify({
            "success": False,
            "error": str(e)
        })

if __name__ == '__main__':
    app.run(debug=True)
```

```python
import joblib
import os
import numpy as np
import pandas as pd
import spacy
from scipy.sparse import hstack
from io import StringIO

# Load spaCy model for lemmatization
try:
    nlp = spacy.load("en_core_web_sm", disable=["parser", "ner"])
except OSError:
    print("Please download spaCy model with: python -m spacy download en_core_web_sm")
    exit()

def lemmatize(text):
    doc = nlp(text)
      return ' '.join(token.lemma_.lower() for token in doc if not token.is_punct and not
token.is_space)

# Load all model components
MODEL_DIR = "models_per_field"
try:
    vectorizer = joblib.load(os.path.join(MODEL_DIR, "vectorizer.joblib"))
    sbert_encoder = joblib.load(os.path.join(MODEL_DIR, "sbert_encoder.joblib"))

    fields = [
        "attackVector", "attackComplexity", "privilegesRequired", "userInteraction",
        "scope", "confidentialityImpact", "integrityImpact", "availabilityImpact"
    ]

    loaded_models = {
        field: joblib.load(os.path.join(MODEL_DIR, f"{field}_model_bundle.joblib"))
        for field in fields
    }

```

```
except Exception as e:
    print(f"Error loading models: {e}")
    exit()

def predict_and_evaluate(csv_text):
    # Parse pasted CSV input (no headers expected)
    input_df = pd.read_csv(StringIO(csv_text), header=None)
    input_df.columns = [
        "cve_id", "description", "baseSeverity", "attackVector", "attackComplexity",
        "privilegesRequired", "userInteraction", "scope",
        "confidentialityImpact", "integrityImpact", "availabilityImpact"
    ]

    # Preprocess description
    lemmatized = [lemmatize(desc) for desc in input_df['description']]
    tfidf_vec = vectorizer.transform(lemmatized)
    sbert_vec = sbert_encoder.encode(input_df['description'].tolist())
    combined_features = hstack([sbert_vec, tfidf_vec]).tocsr()  # 🔧 Fix: convert to CSR for slicing

    results = []
    correct_counts = {field: 0 for field in fields}
    total = len(input_df)

    for i in range(total):
        row = {"cve_id": input_df.loc[i, "cve_id"]}
        features = combined_features[i:i+1]

        for field in fields:
            model = loaded_models[field]['model']
            encoder = loaded_models[field]['encoder']

            pred = model.predict(features)[0]
            pred_label = encoder.inverse_transform([pred])[0]
            true_label = input_df.loc[i, field]

            row[field] = {"predicted": pred_label, "actual": true_label}
            if pred_label == true_label:
                correct_counts[field] += 1

        results.append(row)

    stats = {
        field: {
            "correct": correct_counts[field],
            "total": total,
            "accuracy": round(100 * correct_counts[field] / total, 2)
        }
        for field in fields
    }

    return results, stats
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>CVSS Prediction</title>
</head>
<body>
  <h2>Paste CVEs (without headers)</h2>
  <form id="predict-form">
    <textarea name="csv_text" id="csv_text" rows="20" cols="120" placeholder="Paste your CVEs here..."></textarea><br>
    <button type="submit">Predict</button>
  </form>

  <div id="results"></div>

  <script>
    const form = document.getElementById('predict-form');
    form.onsubmit = async function (e) {
      e.preventDefault();
      const formData = new FormData(form);
      const res = await fetch('/predict', {
        method: 'POST',
        body: formData
      });
      const data = await res.json();
      const resultsDiv = document.getElementById('results');
      resultsDiv.innerHTML = '';

      if (data.success) {
        const stats = data.stats;
        const total = Object.values(stats)[0].total;
        let statsHTML = `<h3>Processed ${total} entries</h3><ul>`;
        for (const field in stats) {
          const s = stats[field];
          statsHTML += `<li>${field}: ${s.correct}/${s.total} correct (${s.accuracy}%)</li>`;
        }
        statsHTML += '</ul>';
        resultsDiv.innerHTML = statsHTML;
      } else {
        resultsDiv.innerHTML = `<p style="color:red;">Error: ${data.error}</p>`;
      }
    };
  </script>
</body>
</html>

```