National Technical University of Athens
IPSP Data Science and Machine Learning

RiskML: Data-Driven Cyber-Risk Predictions Leveraging State-of-the-Art Machine Learning Operations in the Healthcare Domain

Diploma Thesis

Sotiris Pelekis

Supervision:
Dimitris Askounis
Professor - Department of Electrical and Computer Engineering, NTUA

Approved by the three-member comittee the 29th of December 2021.

(Signature)	(Signature)	(Signature)
Dimitris Askounis	George Stamou	John Psarras
Professor at NTUA	Professor at NTUA	Professor at NTUA

Athens, December 2021

Εθνικο Μετσοβιο Πολυτεχνειο

ΔΠΜΣ Επιστημη Δεδομενων και Μηχανικη Μαθηση

RiskML: Εφαρμογή Προβλέψεων Ρίσκου Κυβερνοασφάλειας με Αξιοποίηση Τεχνολογιών Διαδικασιών Μηχανικής Μάθησης στον Κλάδο της Υγείας

Διπλωματικη Εργασια

Σωτηρης Πελεκης

Επιβλέπων:
Δημήτρης Ασκούνης
Καθηγητής Ε.Μ.Π. - Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Δεκεμβρίου 2021.

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
Δημήτρης Ασκούνης	Γιώργος Στάμου	Ιωάννης Ψαρράς
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2021

Solemn Declaration
I certify that I am the author of this dissertation, and that any assistance I have received in preparing it is fully acknowledged and referred to in the dissertation. I have also cited any sources from which I have used data, ideas or words, whether they are exact or paraphrased. I also certify that this dissertation was prepared personally for the requirements of the curriculum of the Department of Electrical and Computer Engineering of the National Technical University of Athens.

(Signature)

Sotiris Pelekis
Graduate of the Department of Electrical and Computer Engineering of the National Technical University of Athens

Εκτεταμένη Περίληψη

Τα δεδομένα αποτελούν σημαντικό περιουσιακό στοιχείο σε κάθε σύγχρονη επιχείρηση και οργανισμό. Τα πολύτιμα δεδομένα ενός οργανισμού μπορεί να περιλαμβάνουν ιδιωτικές πληροφορίες όπως ιατρικά αρχεία, αριθμούς πιστωτικών καρτών, δεδομένα πελατών αποθηκευμένα στο cloud, εμπορικά μυστικά ή και δημόσιες πληροφορίες όπως για παράδειγμα ο ιστότοπος μιας εταιρείας διαδικτυακού εμπορίου. Οποιοδήποτε περιστατικό κυβερνοασφάλειας σχετιζόμενο με τέτοια δεδομένα, είτε σκόπιμο (στοχευμένες επιθέσεις), είτε ακούσιο (εσωτερικά σφάλματα), μπορεί να διαταράξει μια επιχείρηση και να προκαλέσει ζημιά στα περιουσιακά στοιχεία και τη φήμη της. Ως εκ τούτου, ένα μέρος των πόρων ενός οργανισμού συχνά αφιερώνεται στην προστασία του από τέτοια συμβάντα ασφαλείας. Σχετικά προληπτικά μέτρα αποτελούν η διατήρηση τακτικών αντιγράφων ασφαλείας, η ενημέρωση του λογισμικού και η εκπαίδευση των εργαζομένων προκειμένου να μειωθούν πιθανά ανθρώπινα λάθη. Ωστόσο, ο καθορισμός του τρόπου κατανομής πόρων, καθώς και ο καθορισμός του βέλτιστου επιπέδου επένδυσης σε προληπτικά μέτρα, είναι σημαντική διαδικασία, καθώς οι μέθοδοι επίθεσης ποικίλουν που διαρκώς εξελίσσονται. Στο πλαίσιο αυτό, μεταξύ άλλων, έχουν δημιουργηθεί βάσεις δεδομένων, όπως η Veris Community Database (VCDB) [1] ή το Hackmaggedon [2] οι οποίες καταγράφουν τέτοια περιστατικά. Ταυτόχρονα μια μεγάλη ερευνητική προσπάθεια διεξάγεται στην ανάπτυξη μεθοδολογιών εκτίμησης ρίσκου κυβερνοασφάλειας βασισμένων σε τέτοιου είδους δεδομένα. Άλλες έρευνες εστιάζουν στον εντοπισμό απειλών και συχνά στον μετριασμό του κινδύνου. Σε όλες τις ανωτέρω περιπτώσεις, η μηχανική μάθηση βρίσκεται πλέον στο επίκεντρο των προσπαθειών. Ωστόσο, το μεγαλύτερο μέρος της τρέχουσας έρευνας στον τομέα αυτό έχει επικεντρωθεί κυρίως στην ανάπτυξη και εκπαίδευση μοντέλων, αγνοώντας ζητήματα που σχετίζονται με τη ροή δεδομένων (data flow), το σερβίρισμα (model serving), την ενημέρωση μοντέλων καθώς και τη διαχείριση ολόκληρου του κύκλου ζωής της μηχανικής μάθησης [3].
Η διπλωματική εργασία αυτή αρχικά επιχειρεί μια βιβλιογραφική ανασκόπηση ερευνητικών προσεγγίσεων πρόβλεψης ρίσκου κυβερνοασφάλειας που βασίζονται στη μηχανική μάθηση και είναι οι εξής:

1. Αναφορές και σύνολα δεδομένων οργανισμών
2. Σύνολα δεδομένων εκτελέσιμου κώδικα
3. Σύνολα δεδομένων δικτύου
4. Συνθετικά σύνολα δεδομένων
5. Δεδομένα ιστοσελίδων
6. Δεδομένα μέσων κοινωνικής δικτύωσης
7. Σύνολα δεδομένων μικτού τύπου

Στη συνέχεια η ανάλυση επικεντρώνεται στην πρώτη κατηγορία καθώς σε αυτά τα σύνολα συγκαταλέγεται και η βάση δεδομένων VCDB. Οι σημαντικότερες ερευνητικές εργασίες που αξιοποιούν τις αναφορές οργανισμών ή τα σύνολα δεδομένων ως πηγή δεδομένων είναι [4], [5], [6]. Καμία από αυτές ωστόσο δεν έχει καταπιαστεί με

διαδικασίες μηχανικής μάθησης (MLOps).
Ακολούθως, εγκαθιδρύεται αναλυτικά το υπόβαθρο του ταχέως αναπτυσσόμενου κλάδου της πληροφορικής και της μηχανικής μάθησης δηλαδή αυτό των διαδικασιών μηχανικής μάθησης μαζί με τις τρέχουσες μεθοδολογίες και εργαλεία λογισμικού του τομέα αυτού. Οι διαδικασίες μηχανικής μάθησης καλύπτουν τις ανάγκες i) διατήρησης της ιστορικότητας του τρίπτυχου: κώδικας, δεδομένα, μοντέλα ii) κατανεμημένης επεξεργασίας και ανάλυσης δεδομένων iii) εκπαίδευσης κατανεμημένων μοντέλων iv) οργάνωσης και διαχείρισης αρχιτεκτονικών σωλήνωσης μηχανικής μάθησης v) σερβίρισματος μοντέλων μηχανικής μάθησης στην παραγωγή. Οι διαδικασίες μηχανικής μάθησης διευκολύνουν σημαντικά την ανάπτυξη και τη διατήρηση σχετικών εφαρμογών αυτοματοποιώντας τα διάφορα στάδια του κύκλου ζωής της μηχανικής μάθησης (machine learning lifecycle), επιταχύνοντας μαζικά τις διαδικασίες ανάπτυξης και συντήρησης. Υπό ένα πρίσμα πλήρους αυτοματοποίησης, οι ομάδες μπορούν να συμβαδίζουν με την αιχμή της τεχνολογίας ML και να αναπτύσσουν γρήγορα νέα μοντέλα. Γι' αυτό και τέτοιου τύπου τεχνολογίες μελετήθηκαν και υιοθετήθηκαν στο πλαίσιο της διπλωματικής εργασίας αυτής.
Στη συνέχεια, δεδομένου ότι η Veris Community Database (VCDB) βρίσκεται στον πυρήνα αυτής της εργασίας, διεξάγεται μια εκτενής διερευνητική ανάλυση δεδομένων σε αυτή και πραγματοποιείται μια αναλυτική περιγραφή της ιεραρχικής δομής της. Η VCDB στοχεύει στη συλλογή και τη διάδοση πληροφοριών παραβίασης δεδομένων για όλα τα συμβάντα κυβερνοασφάλειας που αποκαλύπτονται δημόσια. Τα δεδομένα κωδικοποιούνται στη μορφή VERIS [7] και διατίθενται για δημόσια χρήση. Η ανάλυση δεδομένων, επιτρέπει την εξαγωγή χρήσιμων συσχετισμών και αναλυτικών στοιχείων μεταξύ των χαρακτηριστικών του συνόλου δεδομένων καθώς και την εξερεύνηση του τρέχοντος status quo στην κυβερνοασφάλεια. Περαιτέρω έμφαση δίνεται στον τομέα της υγειονομικής περίθαλψης καθώς το κύριο παράδειγμα χρήσης της εφαρμογής αναπτύχθηκε σε σχετικό περιβάλλον. Τα κύρια συμπεράσματα της ανάλυσης αυτής είναι τα εξής:

- Ο αριθμός καταγεγραμμένων συμβάντων στη VCDB παρουσιάζει ένα μειούμενο ρυθμό, χωρίς αυτό να συνεπάγεται απαραίτητα πραγματική μείωση παγκοσμίως, αλλα μείωση των καταχωρήσεων στη συγκεκριμένη βάση. Ωστόσο η πανδημία του Covid-19 οδήγησε σε σημαντική αύξηση εντός του 2020.

- Ο αριθμός συμβάντων παραβίασης κυβερνοασφάλειας δε συνάδει απαραίτητα με τον αριθμό παραβιασμένων εγγραφών ανά έτος. Αντιθέτως, αυτός ο δείκτης δείχνει να αυξάνεται διαρκώς με ολική κορύφωση το 2020. Αυτό σημαίνει ότι πλέον ο αντίκτυπος ανά συμβάν / κυβερνοεπίθεση έχει αυξηθεί ραγδαία.

- Ο δημόσιος τομέας αλλά και ο τομέας υγείας φαίνεται να είναι οι πλέον πληττόμενοι, κυρίως μάλιστα εξ αιτίας ανθρώπινων λαθών. Εικάζεται ότι αυτό συμβαίνει επειδή ο πρώτος χαρακτηρίζεται συνήθως από περιορισμένα πρωτόκολλα ασφαλείας και εκπαίδευση προσωπικού. Αντίστοιχα ο δεύτερος φαίνεται να είναι ο πιο ελκυστικός για τους επιτιθέμενους καθώς τα ηλεκτρονικά συστήματα εκεί κατακλύζονται από ιδιαίτερα ευαίσθητα και μεγάλης αξίας δεδομένα.

Αναφορικά με το πειραματικό σκέλος της διπλωματικής εργασίας αυτής, εκπονείται μια προσπάθεια υλοποίησης μιας αρχιτεκτονικής σωλήνωσης (pipeline) για την διαμόρφωση και επίλυση πολλαπλών διαφορετικών προβλημάτων εκτίμησης ρίσκου αξιοποιώντας την προαναφερθείσα βάση δεδομένων VCDB. Τα προβλήματα

πρόβλεψης κινδύνου αναλύονται σε 3 ομάδες πιθανοτικών εργασιών δυαδικής ταξινόμησης που σχετίζονται με τα παραβιασμένα περιουσιακά στοιχεία, τις ενέργειες απειλών και τα δημογραφικά στοιχεία του θύματος. Οι εργασίες διαμορφώνονται σε πιθανοτικό επίπεδο ως εξής:

- Πιθανότητα εμφάνισης κυβερνοαπειλών όπως διαφορετικοί τύποι hacking και κακόβουλου λογισμικού, ηλεκτρονικό ψαρέμα (phishing) και ανθρώπινα λάθη:

$$\mathbb{P}(\text{είδος απειλής} \mid \text{κλάδος οργανισμού, μέγεθος οργανισμού, είδος περιουσιακού στοιχείου})$$

- Πιθανότητα να γίνει ένα περιουσιακό στοιχείο στόχος μιας συγκεκριμένης απειλής στην οποία το σύστημα έχει ήδη εκτεθεί:

$$\mathbb{P}(\text{περιουσιακού στοιχείου} \mid \text{κλάδος οργανισμού, μέγεθος οργανισμού, είδος δράσης})$$

- Πιθανότητα ένα περιστατικό να επηρεάσει κάποιο από τα συστατικά του τρίπτυχου της ασφάλειας πληροφορίας, δηλαδή εμπιστευτικότητα, ακεραιότητα, διαθεσιμότητα:

$$\mathbb{P}(\text{χαρακτηριστικού} \mid \text{κλάδος οργανισμού, μέγεθος οργανισμού, είδος απειλής, είδος περιουσιακού στοιχείου})$$

Η αρχιτεκτονική αυτή υιοθετεί την τρέχουσα τεχνολογία αιχμής στον κλάδο των διαδικασιών μηχανικής μάθησης (MLOps). Στο επίκεντρο της υλοποίησης βρίσκεται το προγραμματιστικό εργαλείο ανοιχτού κώδικα MLflow [8]. Η σωλήνωση είναι διαμορφωμένη σε βήματα ETL (εξαγωγή, μετατροπή και φόρτωση), προεπεξεργασίας-εκπαίδευσης και αξιολόγησης τα οποία διεξαγονται σειριακά, αλλα με παραμετροποιήσιμο τρόπο, εξυπηρετώντας έτσι τους συνδυασμούς μεταβλητών εισόδου-εξόδου της εκάστοτε διαδικασίας πρόβλεψης. Σε τελικό στάδιο παράγονται αποτελέσματα αξιολόγησης των επιλεχθέντων αλγορίθμων μηχανικής μάθησης. Σε ότι αφορά στα επιλεχθέντα προς εκπαίδευση μοντέλα έγιναν πειράματα σε πολλαπλούς αλγορίθμους μηχανικής μάθησης (Κ-πλησιέστεροι γείτονες [9], LightGBM [10], Τυχαία Δάση [11], Μηχανές Διανυσμάτων Υποστήριξης [12], Λογιστική Παλινδρόμηση [13], Αφελής Μπεϋζιανή Ταξινόμηση [14]). Ενδεικτικά, παρουσιάζονται δύο πειράματα:

- **Μηχανές διανυσμάτων υποστήριξης (Support Vector Machines) για την πρόβλεψη κινδύνου ασφαλείας διακομιστή.** Στο πείραμα αυτό ο αλγόριθμος με πυρήνα rbf, παράμετρο ποινής του όρου σφάλματος (C) ίση με 100 και αυτοματη εκτίμηση παραμέτρου καμπυλότητας (γ) υπερτερεί όλων των άλλων μοντέλων της οικογένειας.
- **Πολλαπλοί αλγόριθμοι για την πρόβλεψη του κινδύνου απώλειας διαθεσιμότητας δεδομένων**. Στην περίπτωση αυτή, ο αλγόριθμος LightGBM φαίνεται να είναι ο πιο ισορροπημένος μεταξύ άλλων καθώς παρουσιάζει ικανοποιητική βαθμονόμηση (σε επίπεδο πιθανοτικών προβλέψεων - Hosmer Lemeshow p-value) σε συνδυασμό με ικανοποιητικές επιδόσεις ταξινόμησης (F1-Score).

Το τελικό ”προϊόν” της διπλωματικής εργασίας αυτής αποτέλεί μια διαδικτυακή εφαρμογή που ονομάζεται RiskML. Το RiskML επιτρέπει την αυτόματη διεξαγωγή, τροποποίηση, παρακολούθηση και εγκατάσταση στην παραγωγή πολλαπλων μοντέλων μηχανικής μάθησης για εκτίμηση ρίσκου κυβερνοασφάλειας, καλύπτοντας έτσι το κενό που παρατηρείται στον κλάδο αυτή τη στιγμή. Η εφαρμογή αυτή στηρίζεται στην υλοποιημένη αρχιτεκτονική σωλήνωσης του πειραματικού σταδίου και βασίζεται σε τεχνολογίες λογισμικού αιχμής όπως για παράδειγμα τα MLflow [8], Docker [15], Swagger / OpenAPI [16] και MinIO [17]. Για κάθε εργασία μηχανικής μάθησης, καθοριζόμενη από τον τελικό χρήστη, δημιουργούνται ειδικά αρχεία ρυθμίσεων στο Mlflow που διαχειρίζονται διαφορετικές εκπαιδευτικές εργασίες. Για κάθε εργασία, οι διαφορετικοί αλγόριθμοι μηχανικής μάθησης που προαναφέρθηκαν (ή υποσύνολο αυτών, βάσει επιλογής του χρήστη) εκπαιδεύονται με αυτόματο και μαζικό τρόπο και καταληκτικά επιλέγεται για ”σερβίρισμα” στην παράγωγή το μοντέλο με την καλύτερη απόδοση. Σε περίπτωση ενημέρωσης του συνόλου δεδομένων VCDB, τα μοντέλα δύνανται να επανεκπαιδευτούν κατ’ απαίτηση του χρήστη, παρέχοντας έτσι πιο ενημερωμένες και ακριβείς εκτιμήσεις πιθανότητας εμφάνισης απειλών. Τα μοντέλα που τελικώς ”σερβίρονται” μπορούν να καταναλωθούν από άλλους τελικούς χρήστες ή προγραμματιστικούς πράκτορες του περιβάλλοντος. Αναφορικά με την επικύρωση του εργαλείου, το RiskML έχει εγκατασταθεί ήδη σε παραγωγικό περιβάλλον στο πλαίσιο του ευρωπαϊκου έργου SPHINX [18] για να συνεισφέρει με γνώση προερχόμενη από τη VCDB σε ένα ολιστικό εργαλείο αξιολόγησης ρίσκου κυβερνοασφάλειας πραγματικού χρόνου εγκατεστημένου σε νοσοκομειακή υποδομή.

Περίληψη

Ο πυρήνας της παρούσας διπλωματικής εργασίας είναι μια εφαρμογή λογισμικού μηχανικής μάθησης που ονομάζεται RiskML. Το RiskML διεξάγει προβλέψεις ρίσκου κυβερνοασφάλειας με χρήση της βάσης δεδομένων της κοινότητας Veris [1]. Η εφαρμογή απευθύνεται σε όλα τα στάδια του κύκλου ζωής του ML και βασίζεται σε εργαλεία αιχμής του κλάδου των διαδικασιών μηχανικής μάθησης (MLOps). Κατα αυτόν τον τρόπο, μέσω αρχιτεκτονικών σωλήνωσης (pipelines), φέρνει τις μεθοδολογίες πρόβλεψης ρίσκου κυβερνοασφάλειας στην παραγωγή, πράγμα ιδιαίτερα καινοτόμο για τον κλάδο.

Ένα σημαντικό κομμάτι της ερευνητικής προσπάθειας της παρούσας διπλωματικής εργασίας αποτελεί η ανάλυση της βάσης VCDB ως βασική πηγή δεδομένων για πειραματισμό και υλοποιήση μοντέλων. Η VCDB στοχεύει στη συλλογή και τη διάδοση πληροφοριών παραβίασης δεδομένων για όλα τα σχετικά συμβάντα που αποκαλύπτονται δημόσια. Τα δεδομένα κωδικοποιούνται υπό το λεξιλόγιο VERIS [7] και είναι διαθέσιμα για δημόσια χρήση. Προκειμένου να εξερευνηθεί το status quo της επιχειρησιακής κυβερνοασφάλειας και να αναδειχθούν οι πιθανές προσεγγίσεις μηχανικής μάθησης, διεξάγεται μια εκτενής διερευνητική ανάλυση στο σύνολο δεδομένων, εστιάζοντας κυρίως στον τομέα της υγειονομικής περίθαλψης, ο οποίος βρίσκεται στο επίκεντρο αυτής της εργασίας.

Όσον αφορά την πειραματική φάση, δίνεται έμφαση στην επίλυση πολλαπλων εργασιών εκτίμησης ρίσκου στον κυβερνοχώρο που σχετίζονται κυρίως με τα περιουσιακά στοιχεία των επιχειρήσεων, χρησιμοποιώντας μια αρχιτεκτονική σωλήνωσης μηχανικής μάθησης. Τέτοιες εργασίες μπορεί να είναι χρήσιμες για πολλαπλα ενδιαφερόμενα μέρη (όπως ερευνητές, επιστήμονες δεδομένων και αναλυτές κυβερνοασφάλειας) και ειδικά στο πλαίσιο υποδομών ζωτικής σημασίας, όπως είναι οι οργανισμοί υγειονομικής περίθαλψης. Πιο συγκεκριμένα, στη φάση αυτή αναπτύσεται μια αρχιτεκτονική σωλήνωσης μηχανικής μάθησης, που βασίζεται στο MLflow [8], και έχει δημιουργηθεί για την αυτοματοποίηση πειραμάτων μηχανικής μάθησης που αποτελούνται από ένα βήμα μετασχηματισμών, ένα βήμα εκπαίδευσης και ένα βήμα αξιολόγησης. Αναλύονται δύο ενδεικτικές περιπτώσεις χρήσης εκτίμησης ρίσκου, ως proof-of-concept για το προτεινόμενο πλαίσιο πειραματισμού MLOps και τις δυνατότητες αυτοματοποίησής τους. Η πρώτη αναφέρεται στο ρίσκο απώλειας διαθεσιμότητας δεδομένων και η δεύτερη στο ρίσκο στοχοποίησης ενός διακομιστή κατά τη διάρκεια ενός συμβάντος κυβερνοασφάλειας. Στο στάδιο αυτό, εκπαιδεύονται και αξιολογούνται πολλαπλοί αλγόριθμοι μηχανικής μάθησης. Σε ότι αφορά την υλοποίηση και την αρχιτεκτονική της υπό ανάπτυξης εφαρμογής, το RiskML επιτρέπει στους τελικούς χρήστες να εγκαταστήσουν τα μοντέλα πρόβλεψης κινδύνου της προτίμησής τους στην παραγωγή προκειμένου να «λύσουν» τα αντίστοιχα προβλήματα μηχανικής μάθησης με γρήγορο και αυτοματοποιημένο τρόπο. Οι προβλέψεις που παράγονται από το RiskML είναι πλήρως παραμετροποιήσιμες από τον τελικό χρήστη και μπορούν να καταναλωθούν μέσω διεπαφής προγραμματισμού εφαρμογών (API) από οποιονδήποτε προγραμματικό πράκτορα ανάλυσης και αξιολόγησης ρίσκου. Τεχνολογίες αιχμής όπως το MLflow, το Docker [15] και το Swagger [16] χρησιμοποιούνται από την εφαρμογή. Σε επίπεδο επίδειξης, το RiskML συμπεριλήφθηκε σε ένα πλαισίο / εργαλείο αξιολόγησης ρίσκου κυβερνοασφάλειας σε πραγματικό χρόνο στο πλαίσιο του ερευνητικού ευρωπαϊκού έργου SPHINX [18],

ως κομμάτι μιας εργαλειοθήκης κυβερνοασφάλειας που στοχεύει στο νοσοκομειακό τομέα. Ο κώδικας υλοποίησης του RiskML είναι δημόσια διαθέσιμος στο Github [19].

Λέξεις κλειδιά

Διαδικασίες Μηχανικής Μάθησης, Δυαδική Ταξινόμηση, Επιτηρούμενη Μάθηση, Εφαρμογή, Κλάδος Υγείας, Κυβερνοασφάλεια, Μηχανική Μάθηση στην Παραγωγή, Πιθανοτική Ταξινόμηση, Προβλέψη Ρίσκου

Abstract

The main output of the current project is a machine learning software application, namely RiskML, that conducts cyber-risk predictions based on the Veris Community Database [1]. RiskML addresses all the stages of the ML lifecycle and is based on state-of-the-art Machine Learning Operations (MLOps) tools. Such a technological approach, that puts cyber-risk prediction methodologies in practice, by deploying them within machine learning pipelines in production, is beyond the current state-of-the-art in the risk assessment research landscape.
A significant part of the research refers to the analysis of VCDB as the core data source of the research and implementation work. VCDB aims to collect and disseminate data breach information for all publicly disclosed data breaches. The data are coded into the VERIS [7] format and are available for public use. To shed light on the status quo of corporate cybersecurity and unlock potential machine learning challenges an extensive data exploratory analysis is conducted on the dataset, mainly focusing on the healthcare domain, which is in the spotlight of this dissertation.
Regarding the experimental phase of the dissertation, it focuses on the conceptualization of multiple asset-related cyber-risk estimation tasks, based on self-developed machine learning pipelines, that can be useful for various corporate stakeholders (such as researchers, data scientists, security analysts, IT staff) especially within critical infrastructures, such as healthcare organizations. A machine learning pipeline, based on MLflow [8], is built for automating machine learning experiments consisting of an ETL step, a training step and an evaluation step. Two indicative risk prediction use cases are analysed, as proof-of-concept for the MLOps experimentation framework, and its automation capabilities. One of them refers to the risk of loss of data availability and the other to the risk of a server being compromised during an incident. Several machine learning algorithms are trained and evaluated.
In practice, the RiskML application allows end-users to deploy the implemented MLOps pipelines in production to ”solve” the risk prediction tasks of their preference in a quick and automated manner and then serve the best models in production. The predictions produced by RiskML are fully configurable by the end-user and can be consumed through API by any programmatical risk assessment agent and be used for holistic and even real-time risk assessment of infrastructures. State-of-the-art technologies such as MLflow, Docker [15], and Swagger [16] empower the deployment. As proof-of-concept, RiskML has also been deployed as part of a real-time cyber-risk assessment agent in the context of the SPHINX [18] EU project, within a cybersecurity toolkit aimed at the healthcare sector. The implementation code of RiskML is publicly available on Github [19].

Keywords

Application, Deployment, Binary Classification, Cybersecurity, Healthcare Domain, Machine Learning Operations, MLflow, Pipeline, Probabilistic Classification, Risκ-ML, Risk Prediction, Supervised Learning

Acknowledgements

I would like to express my very great appreciation to my supervisors Professor Dimitiris Askounis and Professor George Stamou for their patient guidance, enthusiastic encouragement and useful critiques of this research work. Furthermore, I would like to offer my special thanks to my friends George Bouritsas and Orestis Plevrakis, John Pappas and Alexandros Vythoulkas whose advice has been a great help in conceptualizing and mathematically structuring and formalizing the machine learning modeling frameworks as well as contributing to the final review of this document.

Contents

List of Figures

List of Tables

Nomenclature

AI Artificial Intelligence

API Application Programming Interface

AUC Area Under the Curve

CIA Confidentiality Integrity Availability

CLI Command Line Interface

CSV Comma Separated Value

DAG Directed Acyclic Graph

$DDoS$ Distributed Denial of Service

$DevOps$ Software Development IT operations

DL Deep Learning

DoS Denial of Service

EC European Comission

ETL Extract Transform Load

$HTTP$ HyperText Transfer Protocol

IDS Intrusion Detection System

IG Influence Graph

IPS Intrusion Prevention System

IT Information Technology

$JSON$ JavaScript Object Notation

ML Machine Learning

$MLOps$ Machine Learning Operations

PCA Principal Component Analysis

$REST$ Representational state transfer

$SIEM$ Security Information Management System

SVM Support Vector Machine

TPR True Positive Rate

$TSNE$ T-Distributed Stochastic Neighbor Embedding

UI User Interface

$VCDB$ VERIS Community Database

Chapter 1

Introduction

Data are an important asset in every business; the valuable data of an organization may include private information such as medical records, credit card numbers, private customer data stored on the cloud, or even trade secrets, as well as public information such as the website of an online commerce company. Any incident involving such data, whether intentional (targeted attacks) or unintentional (internal errors), can disrupt a business and inflict damage on its assets and reputation. Therefore, a portion of an organization's resources should be dedicated to protecting itself from security incidents; preventive measures include maintaining regular backups, keeping software up to date, and employee education in order to reduce miscellaneous errors. Nonetheless, determining how to allocate resources in protecting one's assets, as well as choosing an optimal level of investment in each preventive measure, is not a trivial task, as there is a wide variety of ever-changing attack methods. To help identify common forms of data incidents, a number of projects have been created to collect information about incidents that involve some sort of data loss. Some of these projects, such as Hackmaggedon [2], focus exclusively on hacking attacks, while others, like the Veris Community Database (VCDB) [1], cover a broader range of incidents, including human errors, and physical loss of data due to theft. Using these reports, organizations are able to identify prevalent incident vectors, and invest in self-protection in a more optimal way. However, a point that should not be overlooked is that not all businesses should be treated the same, as each business is prone to different forms of incidents. For instance, a cloud hosting company might be more likely to suffer from hacking or denial of service attacks, while a medical institution with a large number of personnel runs a relatively higher risk of data loss through human error. Specifically, healthcare environments are critical infrastructures and thus are always in the spotlight and create substantial concerns over the exposure of sensitive medical records, or even the loss of human lives [20], [21]. Consequently, for the prevention of data breaches and the protection of the triptych of Confidentiality, Integrity, Availability (CIA) of information in the business world various tools and methodologies are being developed every day ranging from honeypots [22], signature and anomaly-based intrusion detection systems (IDS) [23], security information and event management systems (SIEM) [24], [25], sandbox environments for experimentation with potential malicious files and devices [26], [27] to cyber-risk assessment methodologies which rely on scientific concepts very adjacent to the ones to be studied within this dissertation and therefore are presented in detail in section 2. In this context, a variety of works and software have attempted to tackle cybersecurity issues with machine learning (ML) and deep learning (DL) techniques, [28] mainly focusing to intrusion detection [29] and honeypot data analysis [30], [31] workflows. Additionally, most of the current effort in the area of ML/DL has mainly focused on model development and training while disregarding issues relating data flow, model serving and updating, and entire ML lifecycle [3] management. This indicates an obvious scientific and business deficit in the domain of cybersecurity and specifically risk assessment and its coupling with the landscape of ML operations (MLOps) [32].

1.1 Purpose of this project - Problem Statement

The current dissertation proposes a ML-based software architecture named RiskML, aiming to fill the unaddressed gap of risk assessment to adopt MLOps. RiskML combines the solution of major a cybersecurity problem, that is cyber-risk prediction in the business world with the employment of state-of-the-art MLOps tools and methodologies. It thus involves multiple stages of the ML lifecycle, such as model-serving and monitoring instead of simple standard scripting procedures that usually only serve the ML model development stage.

In this context, an analysis on the Veris Community Database (VCDB) is conducted by employing Statistical and ML techniques. Its main contribution is an MLOps pipeline [33], that based on different feature subsets (different input / target combinations) of the VCDB dataset it provides long-term estimations for the following 3 risk predition tasks using appropriately trained data-driven models:

- Likelihood of occurrence of cyberthreats such as different types of hacking and malware, social engineering, and human errors:

$$\mathbb{P}(\text{threat type} \mid \text{organisation industry, organization size, asset type}) \tag{1.1}$$

- Likelihood of an asset to be victim of a particular threat that the system has already been exposed to:

$$\mathbb{P}(\text{asset type} \mid \text{organisation industry, organization size, threat type}) \tag{1.2}$$

- Likelihood of an incident to affect Confidentiality, Integrity, Availability, i.e. each one of the attributes of the Information Security Triad:

$$\mathbb{P}(\text{attribute} \mid \text{organisation industry, organization size, threat type, asset type}) \tag{1.3}$$

For each one of these estimations different training tasks are defined. For each task, different machine learning algorithms (i.e. K-nearest neighbours [9], LightGBM [10], Random Forest [11], SVM [12], Logistic Regression [13]) are trained and the one with the best performance is selected and deployed for inference. The VCDB dataset can be updated, and the models can be re-trained on-demand, thus providing more up-to-date and accurate threat occurrence likelihood estimations. Models are served for inference and can be consumed from other agents / end-users of the environment such as the RCRA-core in order to reinforce the final risk assessment tasks.

1.2 Structure of the document

Chapter 1 forms the introduction of this dissertation, illustrating the status quo of cybersecurity in the business world today and the shortcoming of ML-based frameworks in solving relating risk assessment problems. Additionally, it provided a brief introduction of the RiskML MLOps framework as the proposed solution that aims to address specific risk estimation tasks based on VCDB. Chapter 2 presents a literature review relating to ML-based cyber-risk prediction along with an overview of the MLOps landscape that is tools, software and practices. Chapter 3 provides a detailed exploratory analysis of the VCDB database which is the central source of knowledge for RiskML. Following, the experimental model development procedure along with the developed MLOps pipeline in MLflow is presented in Chapter 4 followed by the full technical specifications of the developed RiskML application in Chapter 5.

1.3 Contribution

The current project's output initially includes a research work on the forming and resolution multiple asset-related cyber-risk estimation tasks, based on self-developed ML pipelines, that can be useful for various corporate stakeholders (such as researchers, data scientists, security analysts, IT staff) especially within critical infrastructures, such as healthcare institutions. However the contribution of this project is more than a data analysis task and extends to a fully functional software application, namely RiskML. RiskML addresses all stages of the ML lifecycle leveraging on state-of-the-art DevOps and MLOps tools, unlike what is common in the research landscape of the cyber-risk assessment domain, successfully putting cyber-risk assessment methodologies in practice by deploying them within production pipelines. RiskML has also been integrated in a holistic real-time cyber-risk assessment application that was developed in the context of the SPHINX [18] EU project, providing multiple risk predictions within a cybersecurity toolkit aimed at the healthcare sector. Finally, the implementation code of RiskML is publicly available on Github [19].

Chapter 2

Current State-of-the-Art in Research and Software

This chapter dives into the literature of cyber-risk prediction in order to establish a robust theoretical background for the current research work. Following, it dives into the concepts of today's state-of-the-art MLOps.

2.1 Literature Review in Data-Driven Cyber Risk Prediction

The cyclical diagram, shown in Figure 2.1 enumerates the common steps generally followed by researchers to predict and discover cybersecurity incidents. The model comprises six steps and forms a circle as a continuous and incremental process: (1) cybersecurity incident analysis; (2) security problem modeling; (3) data collection and processing; (4) feature engineering/ representation learning; (5) model customization; (6) evaluation.

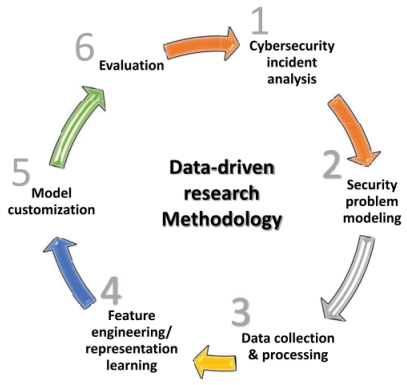

Figure 2.1: Methodology of cyber-incident risk prediction[1]

When trying to predict cybersecurity incidents, data plays a crucial role in the process of analyzing cyberthreats, modeling prediction problems and discovering security incidents. Driven by more and more publicly available data, predicting security trends and discovering indicators of cyber incidents seem to be more feasible than ever. According to [34], data sources can be categorized as follows:

1. **Organization reports and datasets**: Some organizations regularly publish reports or datasets to with security-related information. For example, VCDB [1], which is central in this project, records security incidents under a common format. Important examples that leverage organization reports or datasets as their data source include [4], [5], [6].

2. **Executables datasets**: The content of executable code that can be run by a computer is used as the training dataset. Representative research is [35], [36], [37].

3. **Network datasets**: Network datasets typically record the structure, properties, traffic or symptoms of a network. The main categories of network datasets are

[1] source: [34]

log files [38], network mismanagement symptoms [39], temporal networks from different domains [40] and network traffic [41].

4. **Synthetic datasets**: Prediction models are built and evaluated on synthetic data according to the needs of the use case. [40] and [42] are indicative examples of such work.

5. **Webpage data**: Crawled webpages' contents can also be used as data source as shown in [43], [44], and [45].

6. **Social media data**: Social media is a platform that covers up-to-date insights and information from users around the world. Existing studies in [46], [47], [48], [49], [50] collected data from Tweets, articles and reviews.

7. **Mixed-type datasets**: Some research studies such as [40], [49], and [50] made use of two or more of the above data sources as mixed-type datasets to collect ground truth, and build and validate ML models.

2.1.1 Risk Prediction Based on Organization Reports and Datasets

Having mentioned the above six types of data sources, and taking into account the nature of the current study, which focuses on VCDB [1], the following sections, that are mainly based on the research of Sun et al. in [34], emphasize on the thorough description of the first category referring to organization reports, providing references to seminal work for each kind of dataset.

Predicting Data Breach Incidents Data breaches have been a severe problem in the security field lately. Many researchers and communities are devoted to their detection. Normally, it is already too late when the data breach is detected because the severe damage may have already occurred. However, if such an incident can be forecasted in advance, the organization may survive in, instead of suffering from, it. Liu et al. [4] presented a method to proactively predict an organization's breach incidents based on the externally observed organization's network symptoms data. Firstly, the authors analyzed cybersecurity incidents referenced by VCDB and characterized the extent to which cybersecurity incidents could be predicted. Standing on observations on cyber-incidents, the authors framed the research problem as a binary prediction problem of identifying whether an organization will encounter a data breach incident in the near future based upon externally observed organizations' Internet data instead of data from internal workings of an organization's network. The authors adopted machine learning methods to train and test the classifiers by utilizing organizations' reports data, including security incident data and security posture data. On one hand, security incident data originating from VCDB, Hackmageddon [2] and the Web Hacking Incidents Database [51], serve as ground truth. These three datasets cover the cyber incident events ranging from mid-2013 to 2014. On the other hand, the security posture is quantitatively measured as the level of malicious activities related to an organization along with five mismanagement symptoms. The malicious activities are measured regarding their volume as well as their dynamic behaviors. Mining such information took place by investigating various reputation lists and blacklists, such as CBL [52], SBL [53], SpamCop [54], WPBL [55], PhishTank [56]. Moreover, mismanagement symptoms are obtained based on observations from open Recursive Resolvers, DNS Source Port Randomization, BGP misconfiguration, Untrusted HTTPS Certificates, and

Open SMTP Mail Relays [39] by using databases that record and assess an organization's network. After preprocessing and mapping, 258 externally measurable features are extracted from security posture data. Also, each organization is labeled as "victim" or "non-victim" according to the security incident reports. There are two prediction scenarios proposed in [4], namely short-term prediction and long-term prediction. In terms of ML modeling, the random forest algorithm was utilized. Training datasets comprised a random subset of victim and non-victim organizations. In the short-term prediction scenario, features are extracted from the most up-to-date time ahead of an incident. In the long-term prediction scenario, features are extracted from the periods prior to the first incident happened in the testing dataset. Regarding model evaluation, the prediction performance is assessed with traditional evaluation metrics (including accuracy, true positive, false positive and ROC curve) accompanied by an analysis of the top data breaches in 2014. In this context, the predictive model reaches a combination of 90% TPR and 10% FPR while it accurately forecasts the top 3 data breach incidents [57], namely JP Morgan Chase, eBay, Home Depot. Unfortunately, such estimators can be misleaded by attacker networks reporting fake incidents characterized by "healthy" victim behaviours, or vice versa. The subfield of ML that tries to keep up with such tactics is referred to as adversarial machine learning. However, [4], Liu et al. assume that the data are uncompromisingly real, ignoring the noise and errors in the dataset and leaving this task as potential future work.

Predicting Risk Distributions Over Fine-Grained Data Breach Types Nowadays, businesses are facing various kinds of security incidents, including targeted attacks and internal errors. Once a security incident occurs, a leakage of business data - involving private, as well as public information - is extremely probable. Furthermore, the consequences extend both to the business' assets and reputation. Therefore, organizations tend to invest in protecting themselves from ever-changing security incidents. If the risk distributions can be assessed and predicted, organizations can prioritize the protection and therefore achieve more effective protection with less resources. Sarabi et al. [5] leveraged the business details to train and test a sequence of predictors, which can help organizations prioritize the preventive resource allocation. When the authors analyzed security incidents, they found that no business relates to a single sort of incident. Meanwhile, they noticed that incident reports usually provide security recommendations based exclusively on business sector information. Hence, the ultimate goal of the paper is to employ business details about an organization to predict a sparser set of incidents types compared with [4], so as to provide protection and resource allocation recommendations to an arbitrary organization. For the study, the incidents happened in 2013 and 2014, including 1729 and 592, entries were collected respectively from the VCDB as ground truth. In order to achieve fine-grained cyber incident prediction, each incident was labeled from three fields. The first field is the type of the cyberincident, including environmental, error, hacking, malware, misuse, physical or social. The second is the responsible actor for the attack, labeled as the external, internal or partner. The last is the compromised assets in the incident, containing kiosk/terminal, media, social, network, people, server and device. Additional features gathered for training and testing the predictors are business details from the organization's profiles and websites, which combine information obtained from the VCDB and Alexa Web Information Service (AWIS) [58]. The industry code, number of employees, and the region of operation of the victim organization are three business profiles features extracted from VCDB. AWIS provides the organization's website and statistics information, including

the traffic volume of the website, number of visitors, speed, number of pages linking to the website, and information about the organization that maintains the website (e.g., address, contact information and stock ticker symbol). To forecast the risk of various kinds of data breaches, the authors designed multi-label classifiers and employed the random forest algorithm. Specifically, each binary classifier predicted a field of the incident signature, which is namely action type, actor type or asset type. Regarding the evaluation, predictors were trained on the 2013 incidents data and tested on 2014. The prediction model was evaluated from the risk profiles of the company and the accuracy of the risk assessment model. The result showed that an organization can evade 90% incidents by using 70% of incident types on average. It is worth mentioning that the prediction results seem to be too ambiguous to operate in practice. However, the risk profiles can be potentially used to provide practical security recommendations for security officers of organizations.

Discovering Previous Unknown Malware With Downloader Graph Analytics Malware, has been a vital threat to cybersecurity for a long time. Reported by the PandaLabs [59], 18 million new malware samples were captured in the third quarter of 2016, an average of 200,000 each day. Due to human error, zero-day exploits or other factors, infection is imminent even for the most protected and state-of-the-art systems. Hence, detecting the malware which are previously unknown to the public as early as possible is an effective way to minimize stress, time cost, and damage as well as minimizing incident consequences from their very beginning. However, several malware are almost undetectable by traditional malware detection methods that focus on the analysis of software signatures and behavioural signs. Downloader graph analytics, discussed by Kwon et al. in [6] have the potential of providing indicators of malicious activities and discovering the vast majority of the malware download activities that may otherwise remain undetected. The authors present a malware early-detection system based on the insights from the analysis of downloader graphs. They argue that, due to social engineering or drive-by attacks, users may download additional malware even if they are downloading benign applications. Hence, they propose a graph-based abstraction model to describe the download activities on end-hosts. Based on the abstraction model, they perform a large-scale measurement to investigate the differences in the growth patterns among malicious and benign downloader graphs. Lastly, they employ features extracted from measurements to build a malware early-detection system. The dataset used to develop the malware early-detection system consists of malicious and benign download activities graphs. The graphs are generated by reconstructing download events obtained by anti-virus (AV) telemetry and Symantec's [60] intrusion prevention systems (IPS). To represent the download activities, 19 million influence graphs (IGs), were included in the dataset. Specifically, the nodes of a graph indicate Portable Executable (PE) files (including benign downloaders and malicious downloaders), and the edges of the graph represent download events. According to the ground truth of malicious and benign downloaders, 15 million IGs were labeled as benign, and 0.25 million as malicious. The ground truth was checked by three data sources, containing the downloader records from VirusTotal [61], the National Software Reference Library [62] (NSRL), and Symantec. To explore the differences between malicious and benign downloaders' IGs, the authors conducted an extensive measurement, providing the features to be utilized by the detection system. According to the analysis, there are four apparent indicators of malicious activities: (1) IGs with a large diameter are mostly malicious. (2) IGs

with slow growth rates are primarily malicious. (3) URL access patterns can be distinguished between malicious and benign downloaders. (4) Malware is more prone to download fewer files per domain. Based on the observation, 16 features (including four internal dynamic features, three domain properties features, two down- loader score properties features, four life cycle features and three globally behavior features) were calculated from the IGs. The malware early detection system was developed based on random forest classifiers and leveraging the features obtained from the measurements. To evaluate how early the detection system can detect the previously unknown malicious executables, the authors defined as "early detection" the flagging of unknown executables as malicious before their first submission to VirusTotal. On average, it is shown that this early-detection system can detect unknown malware 9.24 days on average earlier than the VirusTotal anti-virus product. Besides, the authors attempted to perform online detection experiments, simulating the early detection system employed operationally. In the experiment, the training dataset contained data collected before 2014, including 21,543 malicious and 21,755 benign data. For the testing dataset, 12,299 malicious and 12,594 benign data were gathered from the year 2014. The resultant 99.8% TPR and 1.9% FPR highlight the robustness of the system. Although there is a limitation that trojan malwares with rootkit functionality would escape this technique, this method still provides a novel signal and complementary to the current antivirus mechanism.

As already mentioned the current dissertation conducts an analysis and ML modeling procedure, based exclusively on the VCDB dataset and therefore it can be clearly classified as a work of "Predicting Risk Distributions Over Fine-Grained Data Breach Types". However what is novel about this work is that, contrary to similar works such as [5] and [4] a real-world applcation is developed for realising the detection tasks leveraging state-of-the-art MLOps methodologies and tools that are briefly described in the sections to follow.

2.2 The Concept of Machine Learning Operations

Creating ML solutions to various problems can be quite the arduous task due to its multi-stage and complex nature of ML tasks. At this point a reference to the ML lifecycle will be helpful to make this concept clearer.

2.2.1 The Machine Learning Lifecycle

The standard cyclical process followed by ML specialists to resolve ML problems can be defined as the ML lifecycle. The ML lifecycle defines each step that an organisation needs to follow to take advantage of machine learning and artificial intelligence to derive practical business value. The ML lifecycle can be divided in 7 separate stages, as follows:

1. **Problem Definition**
   The first stage of a machine learning project is the definition and understanding of the problem to be solved along with its business value and application. This phase starts with the analysis of goals and the reasons behind a particular problem statement. The ultimate goal is the understanding of the power of data and its usefulness in solving the specific problem and drive results. Asking the right questions is always a great start.

2. **Data Collection**
   Once the goal is clearly defined, it is necessary to acquire the data that is needed from various available data sources. At this stage, some of the questions worth considering are

   - What data is needed for the project?
   - Where is that data available?
   - How can it be obtained?
   - What is the most efficient way to store and access all of it?

   There are various ways to collect data for machine learning projects such as focus groups, interviews, surveys, and internal usage & user data. Most importantly, public data can be another source which is usually free (open data). These include research and trade associations such as banks, publicly-traded corporations, and others. If data is not publicly available, web scraping can be used to obtain it, however there exist several legal restrictions relevant to this method of collection.

3. **Data Wrangling**. The type and quality of data that is used in a machine learning model affects its output considerably. Therefore data wrangling is the crucial and time-consuming process of cleaning and converting raw data into usable format. During this phase data specialists explore, pre-process, condition, and transform data prior to modeling and analysis. Missing values, duplicate data, invalid data, inappropriate formatting, noise are some of the factors that a data scientists will need to handle throughout during this preparation stage and before proceeding to the extraction of insights from the data.

4. **Visualisation and Exploratory Data Analysis**. This stage is essential for the specialist as well for the rest of stakeholders to gain insight on data. Multiple visualisations are created highlighting patterns and trends in the data, thus revealing the relationships / correlations amongst variables of the problem. Generally, in case of large volumes of data, building graphs is the best way to explore and communicate findings, which in turn enables the faster and more meaningful implementation of the latter stages of the machine learning lifecycle. EDA generally leads to faster decision making both at analysis and based on the revealed statistics of the dataset. Common ways of visualisation include area charts, bar plots, density plots, box plots, dot distribution map, histograms, network diagrams, time series plots, word clouds and heatmaps, Sankey diagrams.

5. **Model Development**. This stage includes ML model selection and training. ML helps discover patterns in data and depending on the nature of the problem supervised or unsupervised learning tasks such regression, classification, forecasting and clustering respectively can be applied. During this phase, mathematical, computer science, and business knowledge need to be combined to train an ML algorithm that will make predictions or estimations based on the provided data. Throughout the ML modeling process, appropriate machine learning or deep learning algorithms (DL) need to be selected based on multiple criteria (matching with the nature of the particular dataset, training times, necessity for high accuracy, necessity for interpretability etc.). Finally, often ML algorithms help to identify key features with high predictive value leading to better understanding of the problem and its variables.

6. **Model Evaluation**. It is a crucial step that will determine the quality and accuracy of future predictions in new situations. ML models need to be evaluated according to evaluation metrics (e.g. accuracy, precision, recall, F1-score, for classification, mean squared error, mean absolute error for regression, various similarity metrics for clustering etc.) on previously unseen subsets of the dataset (in case of supervised learning). Optionally, hyperparameter tuning is extremely useful, but often also computationally intensive, in order to discover the best hyperparameter combinations that lead to the best evaluation results for a selected model family. K-folds cross validation [63] is one of the most popular methods to leverage during this process.

7. **Model Deployment and Monitoring**. The last stage is about putting a ML model into a production environment to make data-driven decisions in an automated way. Robustness, compatibility, and scalability are important factors that should be tested and evaluated before deploying a model. There are various ways such as Platform as a Service (PaaS) or Infrastructure as a Service (IaaS). For containerized applications, orchestration platforms such as Kubernetes[64] can be used to rapidly scale the number of containers as demand shifts. Another important part of the last stage is iteration and interpretation. It is critical to constantly optimize the model and monitor the quality of results. Dips in model performance possibly indicate that the entire cycle may need to be repeated to update the model to understand new trends. Operational teams are also responsible for reporting any bugs and unexpected model predictions to the data science team, feedback that also contributes to the start of this whole cycle as the the model needs to be fixed. However, as it is a huge field of study with many components taken into account during the development phase of these models, their integration in production environments faces many difficulties, requiring the continuous development and updating of installed applications. Therefore, the question arises as to what is the right way, if any, to complete these processes successfully, quickly and in full transparency as to the groups responsible for them.

2.2.2 Machine Learning Operations

Many of the challenges faced by developers throughout the deployment of ML applications in production are similar to common software and application development. The solution there was provided by DevOps that is a set of practices aiming at shortening the systems development life cycle and providing continuous delivery with high software quality and maintainability. DevOps is complementary to Agile software development as several DevOps aspects came from the Agile methodology. However, DevOps processes are not enough for the needs of ML due to its complex nature. Thus, there have been developed processes, called MLOps (machine learning operations), that cover the following needs:

- Preserving the history of the triptych: code, data, models
- Distributed processing and analysis of data
- Distributed model training
- Organization and management of ML pipelines
- Model serving (as a service)

MLOps makes it significantly easier to deploy and maintain ML solutions by automating the various ML lifecycle stages, massively expediting the development and maintenance processes. With a fully automated setup, teams can keep up with the latest in ML technology and deploy new models quickly. According to [33], there are 3 different setups representing the different levels of automation:

- **Manual implementation**. It refers to a setup where there are no MLOps principles applied and everything is manually implemented. The steps discussed above in the description of ML lifecycle are all manually performed. Software engineering teams must manually integrate the models into the application, and operational teams must help ensure all functionality is preserved along with collecting data and performance metrics of the model.

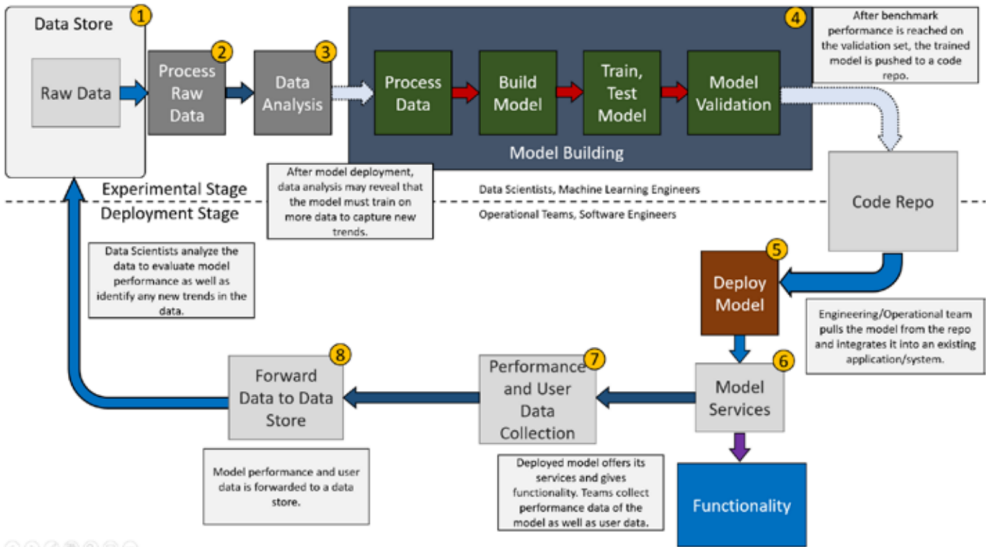

Figure 2.2: Graph depicting a possible deployment setup of a machine learning model without MLOps principles (First Setup).[2]

- **Continuous model delivery**. It is a good middle ground between a manual setup and a fully automated one. The main feature of this type of setup is that it relies on pipelines for the automation of the machine learning side of the process and more specifically to continuously train the model on new data, even after deployment. Automation of the experimental stage, or the model development stage, also emerges along with modularization of code to allow for further automation in the subsequent steps In this setup, continuous delivery refers to expedited development and deployment of new machine learning models. Given the automation of the deployment process, models can now be created or updated at a much faster pace.

- **Continuous integration / continuous delivery of pipelines**. It refers to a setup where pipelines in the experimental stage are thoroughly tested in an automated process to make sure all components work as intended. From there, pipelines are packaged and deployed, where deployment teams deploy the pipeline to a test environment, handle additional testing to ensure both compatibility and functionality, and then deploy it to the production environment. In this setup, pipelines

[2]Source: [32]

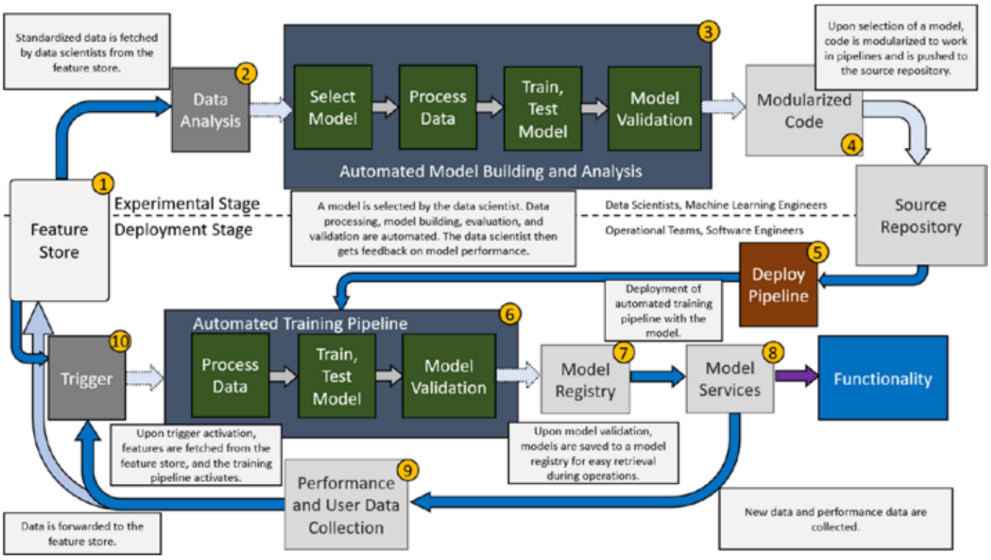

Figure 2.3: Graph depicting a possible deployment setup of a machine learning model with automation via pipelines (Second Setup - Continuous model delivery)[2]

can be created and deployed at a quick pace, allowing for teams to continuously create new pipelines built around the latest in machine learning architectures without any of the resource barriers associated with manual testing and integration.

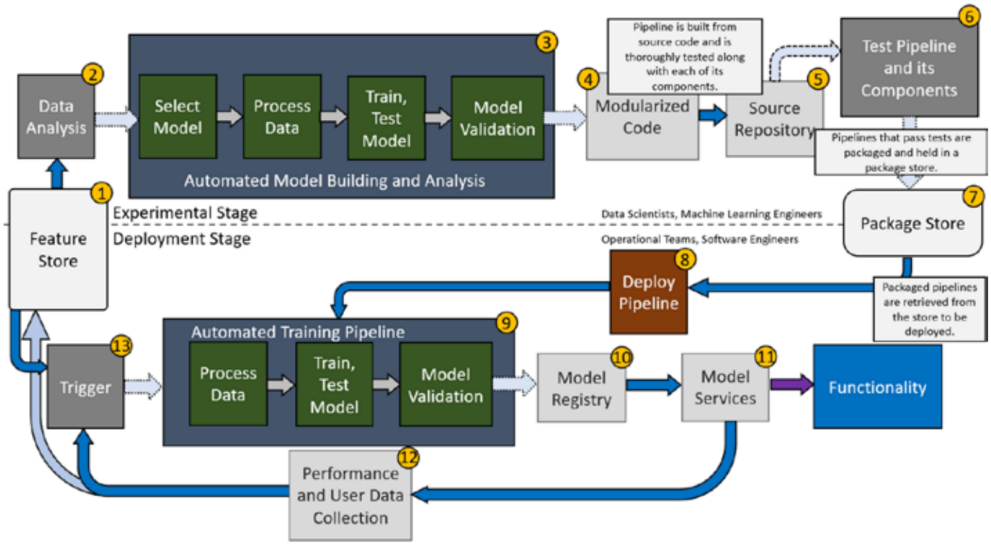

Figure 2.4: Graph depicting the addition of testing systems and a package store to the automation setup (Third setup - Continuous integration / Continuous delivery of pipelines)[2]

2.2.3 MLops Tools Overview

Currently, there is a huge selection of MLOps tools aiming at matching the different categories of operations. Briefly:

- **Model and Data Versioning** are covered by tools such as Pachyderm [65], DVC [66] and Dolt [67]

- **Distributed Processing and Analytics** are covered by software such as Apache Spark [68], Dask [69] and Ray [70]
- **Distributed Model Training** requirements are met by as Horovod [71], RaySGD [72] and Tensorflow Distributed [73] / Pytorch Distributed [74]
- **ML Lifecycle Management and Pipeline Orchestration** is covered by tools such as Apache Airflow [75], Kubeflow [76] and MLflow [8]. These tools are further analysed in the following section as they cover the central part of ML operations and were examined and compared for the management of RiskML workflows.
- **Model Serving** is covered by tools such as Seldon-core [77], Cortex [78], TorchServe [79] and TFX [80]. These tools are further analysed in the following section as they cover the central part of ML operations and were examined and compared for the management of RiskML workflows.

2.2.4 MLflow as the MLOps Core for RiskML

At this point, the main characteristics of the most popular pipeline orchestration tools are briefly in an attempt to identify and justify the best matching option for RiskML that is the application developed in the context of the current dissertation.

- **Apache Airflow** It is a workflow automation and scheduling system that can be used to author and manage data pipelines. Airflow uses workflows made of directed acyclic graphs (DAGs) of tasks. It is a mature open source platform with many capabilities designed for definition, programming and monitoring workflows. Airflow provides capabilities of scaling, dynamic pipeline generation and even execution of tasks in cloud services such as Google Cloud Platform [81], AWS [82] and other third party services. Airflow is a powerful tool but has a really steep learning curve, rendering it inappropriate for simple ML workflows.
- **Kubeflow** It was developed for the deployment of integrated ML workflows in Kubernetes [64] environments. It is compatible with many machine learning technologies such as TensorFlow [83], Seldon-core [77], Jupyter [84], Pachyderm [65], and many more. As it is a very recent tool its capabilities and the technologies it supports are constantly increasing. Kubeflow is a complete solution as it addresses end-to-end ML operations namely recording of experiments, pipeline management, service deployment). However, the wide range of functions provided, lead to high computational and storage requirements.
- **MLflow** MLFlow [8] is an API that allows the integration of MLOps principles into projects with minimal changes made to existing code [33]. This open source platform focuses on the management of the whole ML lifecycle. It is used to create experiments, logging the produced models and metrics (MLflow tracking [85]) and allowing their fast and easy reproduction by developer teams (MLflow Projects [86]). It also enables the deployment of predictive models in various locally as well as on popular cloud services such as Amazon SageMaker [87], Microsoft Azure [88], Google Cloud, and Databricks along with the labeling and management of developed models in a central registry (MLflow Model Registry [89]). MLflow is integrated with major Python [90] ML frameworks such as Scikit-Learn [91], PyTorch [92], Tensorflow [83] and PySpark [93].

The option for this dissertation was MLflow as it enables the end-to-end handling of the whole ML lifecycle and combines efficiency with simplicity and a reasonable learning curve. At the same time it is mostly Python based, leaving space for potential contributions relating to the Python integrations of the abovementioned ML frameworks. Finally, MLflow provides an integration with SHAP [94] which was used to provide interpretations of the developed ML models as transparency and explainability are major principles that should be present in AI systems especially when deployed in critical infrastructures such as a healthcare environments that are considered as high risk according to the recent EU regulation on AI [95]. The official MLflow guide can be found in [33] and as this framework was the choice for this dissertation it was extensively used during RiskML's implementation.

Chapter 3

Experimental Stage - Data Analysis

As the problem has been already defined in Chapter 1 and the data collection process has been solved by an open-source dataset (VCDB) [1] this section dives straight into the Exploratory Data Analysis of VCDB, tackling step 4 of the ML lifecycle as it was previously described in Section 2.2.1. Steps 2 and 3 are taken care of in Chapter 4

3.1 Qualitative Characteristics of VCDB

VCDB aims to collect and disseminate data breach information for all publicly disclosed data breaches. The data are coded into the VERIS [7] format and are available for public use. The initial release had just over 1,200 incidents, primarily from years 2012 and 2013. Since then, the dataset has grown to nearly 9,000 individual incidents. Data sources include the Department of Health and Human Services (HHS) incidents, the sites of the various Attorneys General that provide breach notification source documents, media reports and press releases. The database contains information such as incident demographics, victim demographics (industry, country of operation, size etc.) and incident description (actors, actions, assets affected and breached attributes aka confidentiality, availability, integrity). At this point, it is important to mention that the default VCDB schema [96] is hierarchical and consists of descriptive nested JSON files for each incident. It is worth noting that VCDB does not only include cybersecurity related events but even physical ones, as long as they relate to information security. Nevertheless, the majority of recorded events originate from the digital world and larger emphasis is put on such.
An insight of the database's hierarchical structure can be gained by observing that, for example, an incident that has not involved hacking can neither involve DoS attack and therefore the incident json file will not contain any reference to DoS attacks, as there will not exist a path that could lead there via a nested approach. However, a "flattened" tabular version [97] of the dataset can be used, one that is appropriate for training ML models. This version was used throughout the ML tasks of this analysis. To illustrate the extensions of the nested approach in the tabular version with an example, the absence of "the action of hacking" during an incident / observation results to a zero value (not True) of the action.Hacking column (1st level) which consequently leads all children or "action.Hacking" or 2nd level features (e.g., action.hacking.variety.DoS, action.hacking.variety.Use of backdoor or C2) to also get zero values as the "parent" feature is "not activated". To better demonstrate this hierarchical structure of the dataset the FP-Growth [98] algorithm was utilized in a rather uncommon but effective way. Leveraging the one-hot-encoded tabular structure of VCDB each incident can be perceived as a "shopping basket" that even contains or not a specific product (VCDB feature). The association rules provided by the FP-Growth algorithm resulted to the structure of Figure 3.1.
This process does not lead to any impressive knowledge extraction, however it confirms the hierarchical structure of the dataset as it can be observed that the "parent" features (e.g. asset.variety.User Dev $\rightarrow$ asset.assets.variety.U - Desktop or laptop) that appear in the right column are usually combined with their "children" features amongst others

frozenset({'asset.assets.variety.U - Desktop or laptop'})	frozenset({'asset.variety.User Dev'})
frozenset({'asset.cloud.Unknown', 'asset.assets.variety.U - Desktop or laptop'})	frozenset({'asset.variety.User Dev'})
frozenset({'security_incident.Confirmed', 'asset.assets.variety.U - Desktop or laptop'})	frozenset({'asset.variety.User Dev'})
frozenset({'security_incident.Confirmed', 'asset.cloud.Unknown', 'asset.assets.variety.U - Desktop or laptop'})	frozenset({'asset.variety.User Dev'})
frozenset({'actor.external.country.Unknown', 'attribute.availability.variety.Loss'})	frozenset({'attribute.Availability', 'actor.External'})
frozenset({'asset.cloud.Unknown', 'actor.external.country.Unknown', 'attribute.availability.variety.Loss'})	frozenset({'attribute.Availability', 'actor.External'})
frozenset({'asset.assets.variety.M - Documents'})	frozenset({'asset.variety.Media'})
frozenset({'attribute.Confidentiality', 'asset.assets.variety.M - Documents'})	frozenset({'asset.variety.Media'})
frozenset({'asset.cloud.Unknown', 'asset.assets.variety.M - Documents'})	frozenset({'asset.variety.Media'})
frozenset({'asset.cloud.Unknown', 'attribute.Confidentiality', 'asset.assets.variety.M - Documents'})	frozenset({'asset.variety.Media'})
frozenset({'security_incident.Confirmed', 'asset.assets.variety.M - Documents'})	frozenset({'asset.variety.Media'})
frozenset({'security_incident.Confirmed', 'attribute.Confidentiality', 'asset.assets.variety.M - Documents'})	frozenset({'asset.variety.Media'})

Figure 3.1: Association rules produced by the FP-Growth algorithm on the one-hot-encoded features of the tabular VCDB

and that is expected given that ”children” features are active only if ”parents” are active too.

3.2 Exploratory Data Analysis and Visualisations

This section proceeds with some interesting exploratory analytics on VCDB, revealing interesting patterns of the dataset. Additionally they illustrate the the current status quo in the cybersecurity domain, often emphasizing on the public and healthcare sectors as they were of higher interest during this research. To better handle data transformations amongst the one hot encoded and ordinary tabular forms of VCDB the verisr2 [99] R [100] package was utilised allowing for easier analytics (see Figure 3.2). Additionally, ggplot2 [101] graphics' creation package was used for the visualizations part.

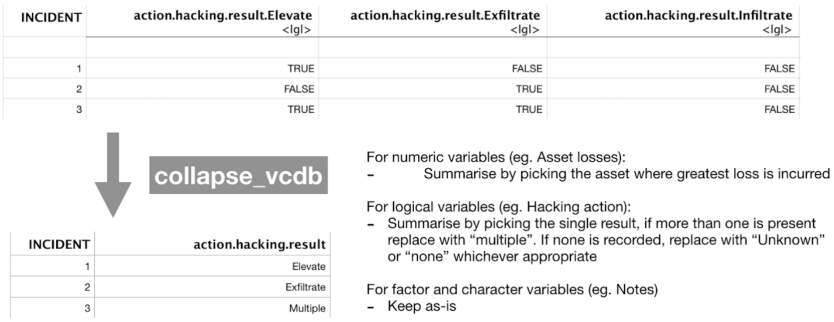

Figure 3.2: The contribution of verisr2 package to the exploratory analysis task

3.2.1 Timeseries Statistics

Initially, the distribution of incidents over time is presented in Figure 3.3 for the period from 2000 to 2021. Observing the graph of Figure 3.3, it appears that the attempted breach events began to increase in 2009, peaking in 2012, 2013 and 2014 and then followed a smooth decline to the levels of 2009 and 2010. However, it is important to investigate the bigger picture here by exploring more statistics, relevant to the ratio of data breaches over incidents and the records lost.

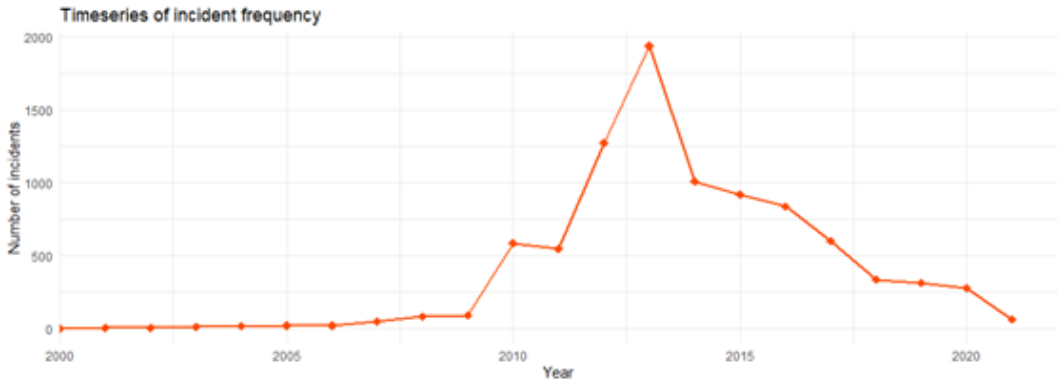

Figure 3.3: Distribution of incidents over the years 2000-2021

Incidents and Breaches

Following, events are separated into data breaches (data leaked - the confidentiality of information was affected) and those that no information leakage was recorded and that are being referred to from now on as incidents (during an incident only information availability and/or integrity are affected). Figure 3.4 illustrates the distributions of the two types of events per year.

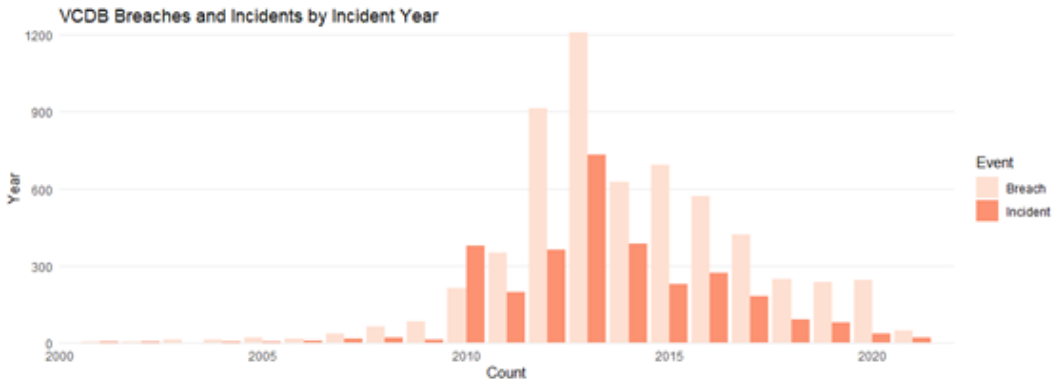

Figure 3.4: Distribution of incidents / breaches over the years 2000-2021

It can be observed that the proportion between them is somehow preserved amongst years while data breach events are consistently higher in number (usually between 2 and 3 times) which seems to happen for 3 main reasons:

1. A company is more likely to initiate the process of realizing, analyzing and listing on VCDB an event if this has resulted in data disclosure as such an event has great impact and damage both to itself and possibly to other stakeholders of its value chain (customers, partners etc.)

2. It is more likely that the attacker's goal is to disclose information either for direct profit (e.g. selling patient records or blackmailing) or for damaging the organization's reputation.

3. An incident is generally more likely to be perceived if it involves personal data and information disclosure as it indirectly involves numerous victims (leaked data holders).

Leaked Records

Observing the numbers of leaked records in VCDB, it is worth noting that the volume of leaked data is not proportional to the number of cybersecurity events. For example, a data leak from Yahoo in 2013 resulted in loss of about 1 billion of records and one from Advanced Info Service [102] in 2020 resulted to 8.3 billion disclosed records. These values are considered enormous compared to the average of a data breach. For this reason the distribution of the number of records the average leaked records per year is depicted in 3.5 leading to the conclusion that the peaks of leaked records are not aligned with those of the number of data breaches. Furthermore, it is worth noting that while the trend of events is decreasing, that of average lost records per incident is increasing with a sharp peak in 2020 which might reveal the following:

- The sophistication of cyberattacks increases rapidly as years go by, given that cyberincidents result to higher losses of data
- Businesses on average tend to store more and more personal data, which is expected given the penetration of technology in contemporary life
- The Covid-19 pandemic seems to have seriously affected the severity of data breaches. This seems to be strongly related to the previous bullet as Covid-19 was an opportunity for digital transformation for many states, organizations and businesses [103], leading to many more data records stored online.

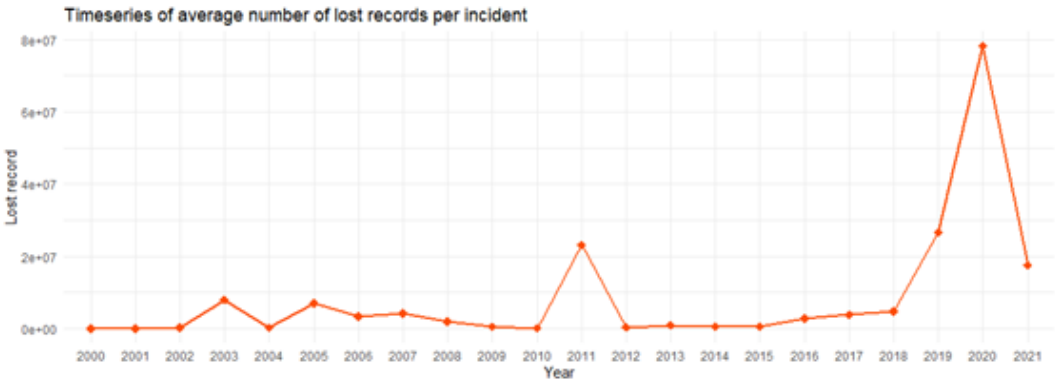

Figure 3.5: Distribution of leaked records per year for the period 2000-2021

3.2.2 Generic Analytics on VCDB

This section provides statistics regarding the actors, actions, assets and patterns observed within the cybersecurity incidents and breaches that took place between 2000 and 2021.

Actor-Centric Analytics

Figure 3.6 depicts the actor type distributions depending on whether it is external or internal to the victim organization. It is worth noting that in most cases the actor variety is unknown (>60%). For external actors a quarter is dedicated to unaffiliated actors, organized crime and activists. As far as internal actors are concerned, we finally see the damage to be done mainly by end-users and system-admins (elevated permissions). This is somehow expected as end-users are many in number, impersonal and indifferent

to the organization's wellness while system-admins have too much power and elevated permissions (privileges) inside the organization's systems.

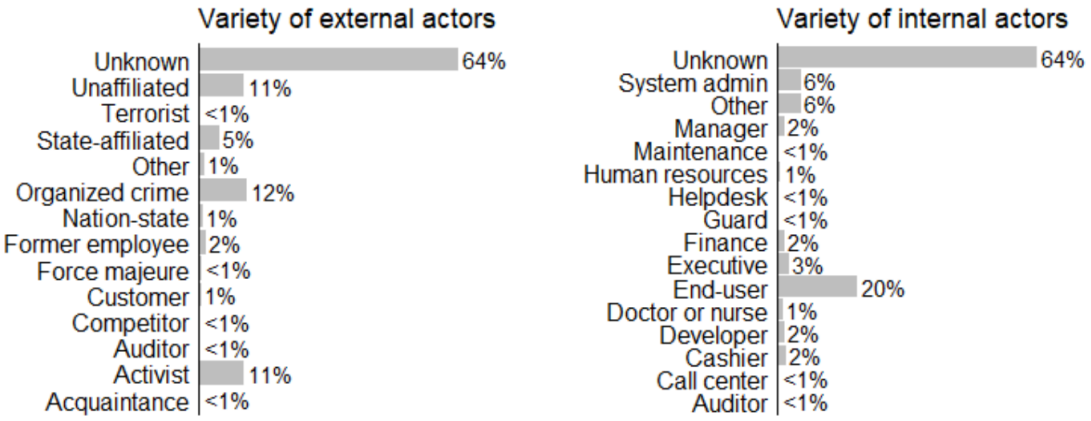

Figure 3.6: Distributions of actors conditioned by their origin (internal, external)

Following, the distribution of of actors over events conditioned by attack patterns are illustrated in Figure 3.7. These pattern categories have been defined within the Data Breach Investigation Report [104] (DBIR) conducted yearly by Verizon [105].

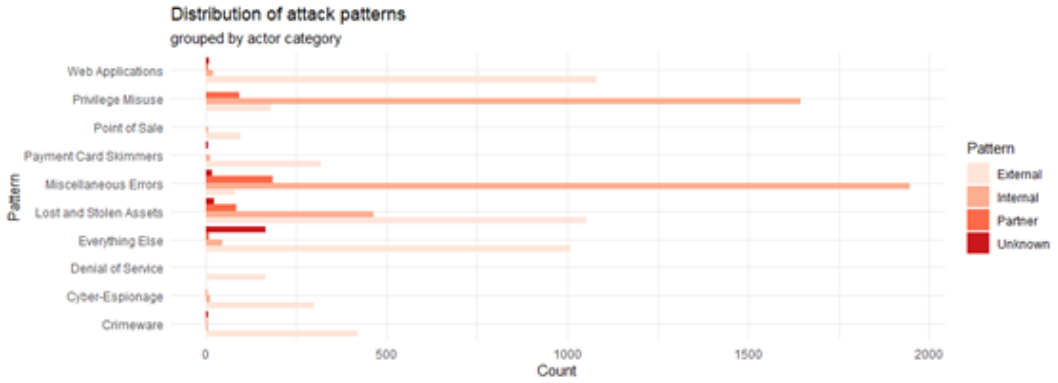

Figure 3.7: Distribution of actors over events conditioned on DBIR pattern

Again it is worth to mention that most cybersecurity events are caused by actors that are internal to the victim organization (usually due to lost assets or privilege misuse)

Action and Asset-Centric Analytics

The distribution of cybersecurity events conditioned on the variety of targeted asset and the type of action that led to the event is with is illustrated in 3.8. It becomes obvious that most incidents are related to server hacking followed by human errors related to storage media as also mentioned previously.
Following, the distribution of the size of the victim organization conditioned by the actions is illustrated in Figure 3.9. It is observed that large organizations can better report their incidents leading to lower percentage of "Unknown" actions.

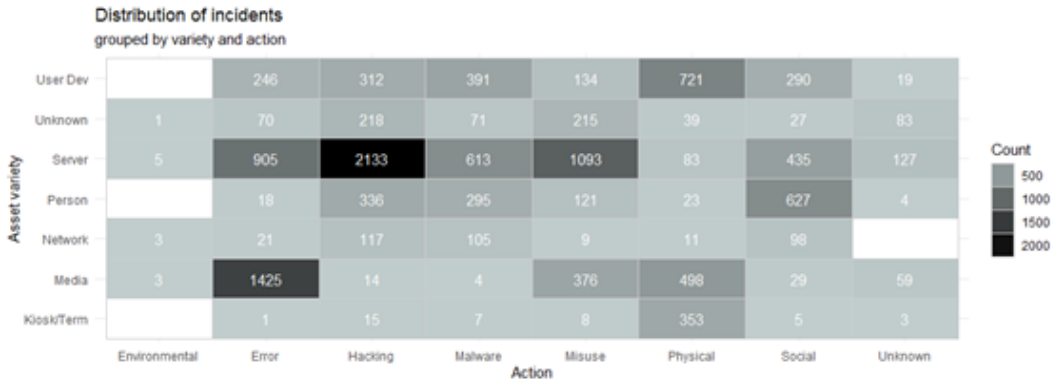

Figure 3.8: Two dimensional histogram representing the joint distribution of action and asset-variety within cybersecurity events

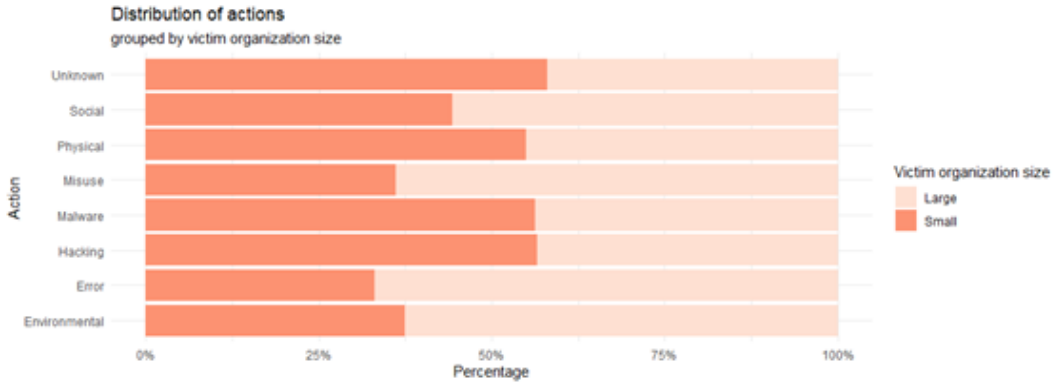

Figure 3.9: Barplot depicting the joint distribution of organization size over the action of the incident / breach

An unsupervised approach for DBIR attack patterns validation

As previously mentioned, Verizon has defined within its yearly Data Breach Investigation Reports the concept of attack patterns which is a second level classification of the VCDB events. To validate the success of such categorization the T-distributed stochastic neighbor embedding method [106] has been used which allows for meaningful visualizations through dimensionality reduction and clustering.The results of TSNE on VCDB are illustrated in Figure 3.10 and serve as a validator of the proposed pattern ontology by Verizon as data clusters seem to be well separated in an organized manner.

Spatial Analysis

It has also been considered of interest to perform a spatial analysis of the events recorded in VCDB. Therefore, Figure 3.11 depicts a heatmap of the number of events per country. The victim country is decided based on the location of the headquarters of the victim organization. It can be observed that large countries are colored with deep reds meaning that many events have ocurred there, however the size of a country per se normally leads to larger population, larger markets and industry and therefore higher chances of reporting cybersecurity events in general and therefore such a map cannot offer many insights.
To further extract spatial knowledge on the dataset, the concept of frequent subsets is

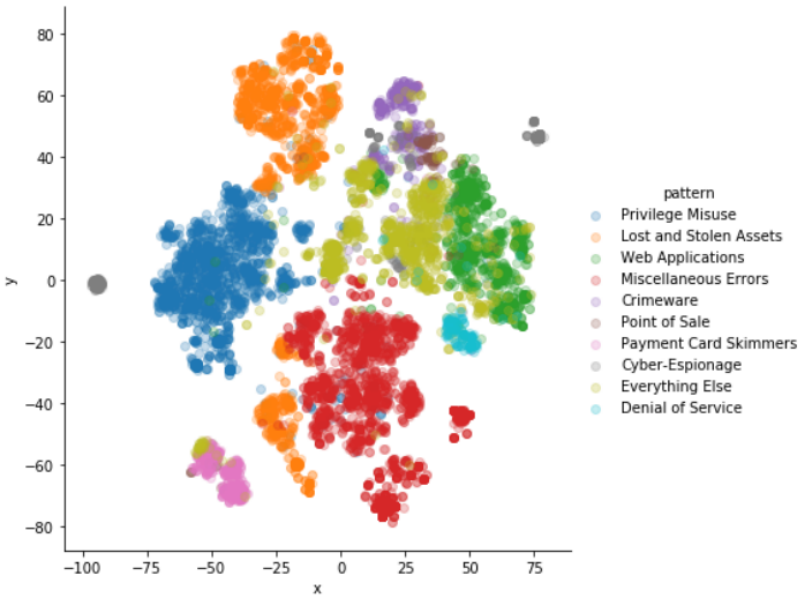

Figure 3.10: Clustering of DBIR patterns with the TSNE method

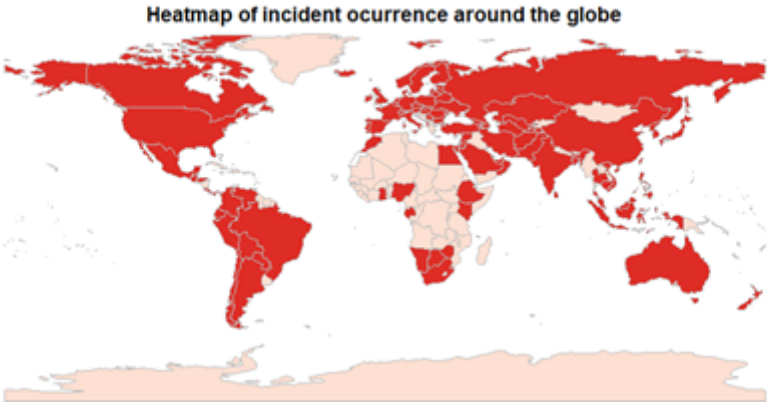

Figure 3.11: Heatmap of VCDB observations (cybersecurity events) per victim country

utilised again in order to provide some insights linking perpetrator and victim countries of a cyberincident. The FP-Growth algorithm was reused on the "victim.country actor.external.country" subcolumns of the dataset. The results are demonstrated in Table 3.1. Additionally performing some counts to reveal "bad relationships" amongst countries some insights are the following:

- Large countries such as UK and US tend to suffer from cyberattacks originating from the inside.
- Pakistan tends to attack India with 22 recorded incidents until now.
- North Korea tends to attack South Korea with 18 recorded events until now.
- Amongst others frequent victims of actors of unknown origin are Germany, China, Canada and Japan.

Table 3.1: Countries with "rich" history of incidents and data breaches

antecedents	**consequents**
actor.external.country.US	victim.country.US
actor.external.country.US, actor.external.country.Unknown	victim.country.US
actor.external.country.KP	victim.country.KR
actor.external.country.GB	victim.country.GB
victim.country.Unknown	actor.external.country.Unknown
victim.country.IL	actor.external.country.Unknown
victim.country.DE	actor.external.country.Unknown
victim.country.CN	actor.external.country.Unknown
actor.external.country.PK	victim.country.IN
victim.country.JP	actor.external.country.Unknown
victim.country.KR	actor.external.country.KP
victim.country.IN	actor.external.country.PK
victim.country.GB	actor.external.country.GB
victim.country.US	actor.external.country.US
actor.external.country.Unknown,	victim.country.Unknown
victim.country.US	actor.external.country.US, actor.external.country.Unknown
actor.external.country.Unknown	victim.country.CN
actor.external.country.Unknown	victim.country.DE
actor.external.country.Unknown	victim.country.JP
actor.external.country.Unknown	victim.country.IL

3.2.3 Sectoral Analytics

As this dissertation aims at the development of a production-oriented architecture, aimed at assisting cybersecurity toolkits, a more detailed approach has been adopted in terms of sectoral analytics, to allow the mining relationships and patterns and revelation of important cybersecurity aspects in the industrial landscape. The public and healthcare domains are emphasized on during the analysis as RiskML has initially been intended for relevant critical infrastructures.

Initially, Figure 3.12 depicts the distribution of events across victim industries. The top 6 are included for a better quality of the visualization. It can be observed that the public and healthcare sectors are the top victims of cybersecurity incidents and data breaches. This is expected as the public domain is most probably characterised by weak security protocols, employee training and standards while healthcare is always in the spotlight of cyberattacks as it is connected with valuable data such as healthcare records.

To extract time-related information Figure 3.13 illustrates the sectoral frequency of incidents (and data breaches) demonstrating that unlike the other sectors the healthcare domain maintained high rates during the period 2014-2017 where the rest of domains exhibited considerable decrease of occurring incidents and breaches.

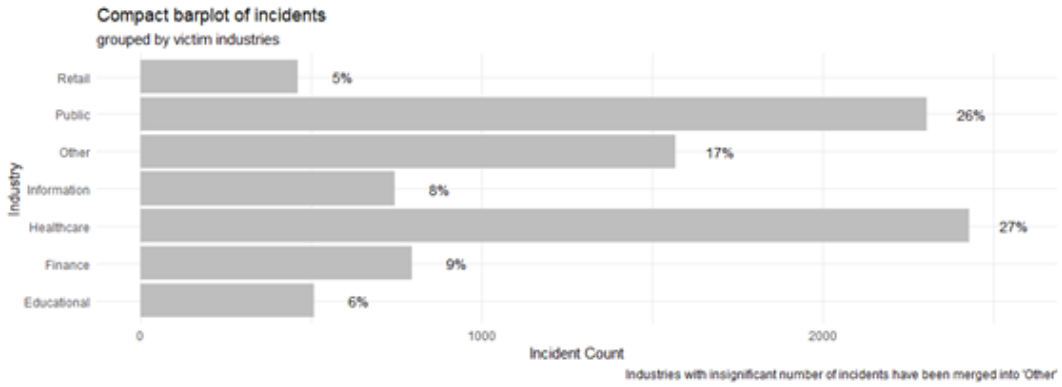

Figure 3.12: Barplot of incidents across industry domains

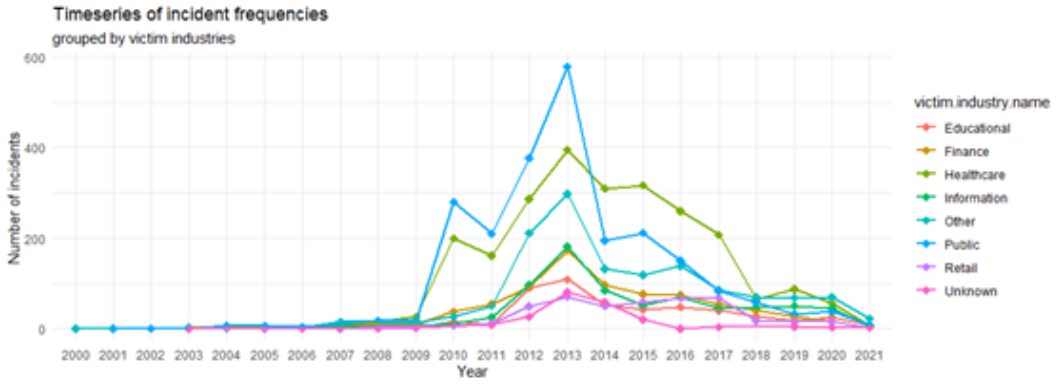

Figure 3.13: Timeseries of incidents (and breaches) per sector for the period 2000-2020

Asset-Wise Perspective of Sectoral Analytics

In terms of victim assets (Figure 3.14), the majority of events are aimed at servers which are the main target of hacking attacks and often media which are usually related to errors (as previously shown in Figure 3.8).

Figure 3.14: Joint distribution of industry and asset variety in VCDB

Action-Wise Perspective of Sectoral Analytics

Following, the joint distribution of events across industry and performed actions is demonstrated in Figure 3.15. The most obvious facts to be noted are the connections of errors and misuse with the public and healthcare sectors. As already mentioned the lack of security standards usually met in the public domain leads to frequent erroneous behaviours. The healthcare sector, like the public sector, seems to suffer very strongly from mistakes of medical staff, and abuse of permissions (e.g. a nurse can leak patient data). On the contrary, IT companies suffer mainly from hacking attacks instead of human error which is expected as they are more likely to adopt security standards and apply security protocols, while employees are expected to be better trained at handling their digital assets as their daily work revolves around them. For example it seems rational that a developer or security engineer is more likely recognize a phishing attempt against their mail account compared to a doctor, nurse or post office employee that are using the organization's electronic devices in an auxiliary manner and that might have never been through cybersecurity related training. Considerable number of events is also related to physical attempts which are mainly observed against the domains of finance (e.g. banks) and again healthcare.

Figure 3.15: Joint event distribution across victim industry and performed action

Actor-Wise Perspective of Sectoral Analytics

Figure 3.16 examines the distribution of actor origins, that is internal and external ones, in terms of their affiliation to the affected organization. It is impressive that only the public sector and the health sector exhibit a larger amount of external than internal actors which means that they are constantly targeted by attackers.
To extend the analysis to actor motives, the lower level graphs of Figure 3.17 are plotted. It is worth to mention that financial motives are mainly behind cybersecurity events in the healthcare domain. This is more than expected as all 3 factors of the information security triad (CIA) are extremely sensitive in what relates with the healthcare domain. To further elaborate, healthcare records can be used both for blackmailing or for sale (e.g. to insurance companies) leading to huge profits (confidentiality breach). Additionally, downtimes of healthcare infrastructures (e.g. via DoS attacks) are unacceptable and can even lead to loss of human life (availability breach). Alteration of patient records can lead to misinformation, reputation damage and even to loss of human life once more (integrity breach). Therefore, attackers are likely to target healthcare infrastructure and data seeking to make profit either directly through selling data or indirectly through

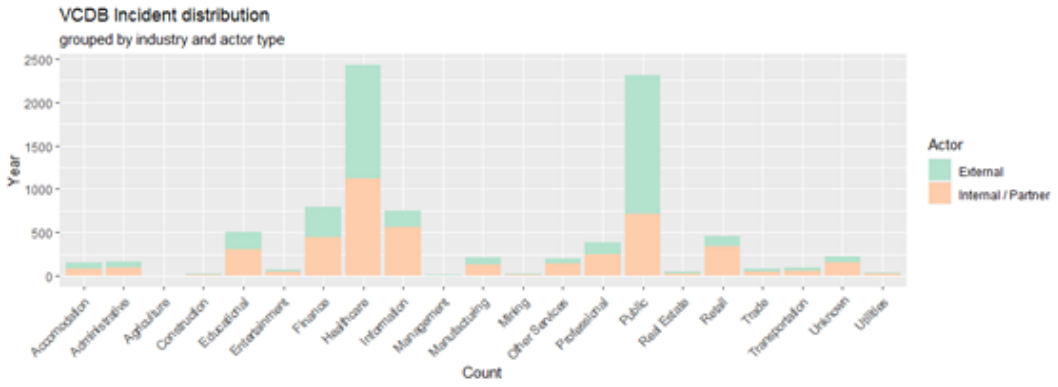

Figure 3.16: Joint event distribution across victim industry and actor origin (internal, external)

damaging the reputation of healthcare organizations or specific individuals.

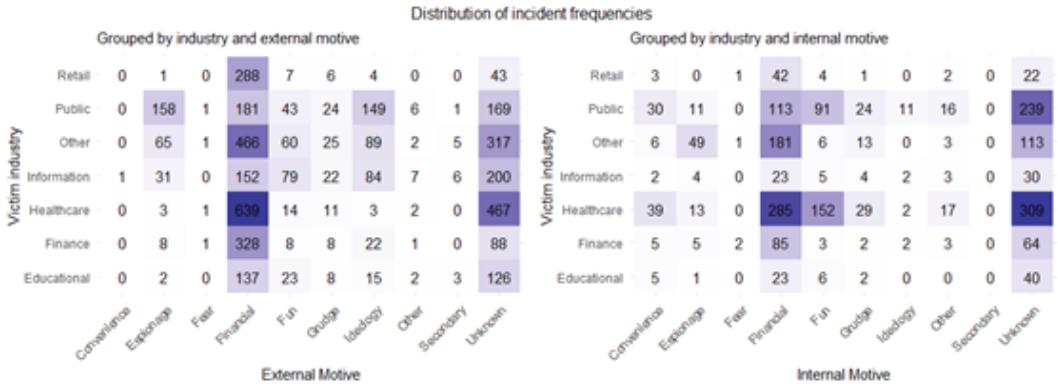

Figure 3.17: Joint conditional distributions of events across the types of motive of internal and external actors

It is also worth noting that numerous events in the public and healthcare domains have been committed "for fun" mainly by internal actors (e.g. end-users of provided web applications), revealing huge vulnerabilities and cybersecurity gaps. Finally, it is observed that espionage related events are also very frequent and, as expected, they mainly target public organizations as they are a result of the continuous inter-state conflicts that have also been examined previously in Section 3.2.2 using the methodologies of frequent subsets.

Pattern-Wise Perspective of Sectoral Analytics

To gain a holistic view of the sectoral status quo of cybersecurity events as depicted by VCDB the joint distribution of victim industry and the already mentioned DBIR patterns is plotted in Figure 3.18.
Once more, the lack of appropriate security policies is apparent in the healthcare domain both in physical (protection from stolen assets) and digital level (protection from cybersecurity incidents and breaches). Healthcare organizations tend to disregard good practices such as physical security (leading to stolen assets), employee training (leading to lost assets and miscellaneous errors), advanced access control and limited provision of user permissions (leading to privilege misuse). In this context, there are numerous

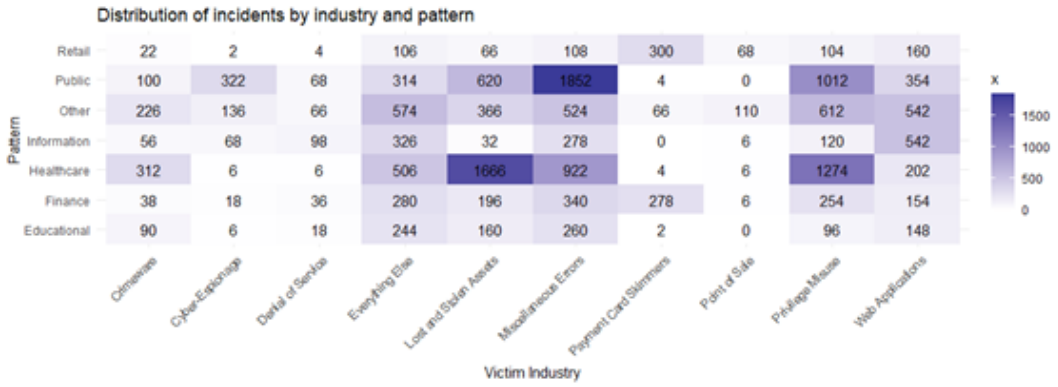

Figure 3.18: Joint event distribution across victim industry and DBIR pattern

cases where employees of large hospital units use common accounts, or share passwords to access the applications of the hospital infrastructure that grant access to health services and potentially to patient medical records as well as critical infrastructure such as servers and medical equipment.

Chapter 4

Experimental Stage - Machine Learning Modeling

This chapter covers the experimental stage of the project. It describes the MLOps pipeline that was implemented along with some indicative, interesting ML problems that can be potentially useful for the business world. Specifically, it addresses the implementation stages 3 (Data Cleaning and Transformations), 5 (Machine Learning Modeling and 6 (Model Evaluation and Hyperparameter Tuning) of the ML lifecycle (refer to Chapter 2) for producing various cyber-risk predictions on the VCDB dataset, leveraging the automated experimentation ML pipeline managed by MLflow. It should be noted that most of the concepts and work explained in this chapter were finally integrated in the the final RiskML application MLops workflows whose purely technical aspects and specifications are described in more detail in chapter 5.

4.1 Defining Specific ML Problems

The purpose of selecting VCDB for to experiment with is that it offers a wide range of well documented security incidents, in a common format, dealing with various industries, assets, actions while also including the effects on the security triad (CIA), and thus offering an ideal opportunity for knowledge and correlation extraction amongst its features as also shown in Chapter 3. Aiming at contributing to the overall situational awareness [107] of a holistic real-time cybersecurity system, a variety of risk risk factors can be defined depending on the current state of the system / infrastructure in defence (e.g. normal conditions or under attack) and the current knowledge (e.g. available information / available asset list of the infrastructure) of the agent that leverages these risk predictions with the final purpose of producing more specific and a fine-tuned risk assessment.

4.1.1 Risk Estimation Use Cases

Assuming some realistic cybersecurity context, which in this case was derived from the setting of SPHINX project, the central risk assessment agent is supposed to have access to the asset list (asset types) of the infrastructure, to the basic characteristics of the protected organization (size, industry, number of employees etc.) along with the threat type (referred to as ”action” with the VERIS ontology [7]) that is potentially threatening the organization. In this context, by creating different input / output combinations derived from VDCB, several asset-related risk estimation problems can be formed for serving the potential needs of the risk assessment agent:

1. Likelihood of occurrence of cyberthreats (actions) such as different types of hacking and malware, social engineering, and human errors:

$$\mathbb{P}(\text{threat type} \mid \text{organization demographics, asset type}) \tag{4.1}$$

   The occurrence of each threat type is being modelled by a unique soft binary classifier that produces the corresponding predictions during the normal conditions

within the infrastructure. This estimation contributes to the overall awareness by predicting the level of exposure/sensitivity of each network asset (e.g. database server, laptop/desktop end-user device) to the different cyber threats (e.g. rootkit, ransomware, DoS).

2. Likelihood of an asset to be victim of a particular threat that the system has already been exposed to and has been reported in the system:

$$\mathbb{P}(\text{asset type} \mid \text{organization demographics, threat type}) \tag{4.2}$$

Separate soft binary classifiers are being trained for each one of the asset types separately. These estimations contribute to the overall awareness by estimating the probability of a specific asset to be targeted by a threat that is already apparent in the system (under attack state).

3. Likelihood of an incident to affect Confidentiality, Integrity, Availability (Information security triad):

$$\mathbb{P}(\text{attribute} \mid \text{organization demographics, threat type, asset type}) \tag{4.3}$$

This estimation contributes to the overall awareness by estimating the probability of an attack that exhibits specific characteristics to target each attribute of the CIA triad. One model is trained for each one of the three attributes returning some risk predictions that depend on the nature of the incident based on past incidents from the real world. For example, the model will return a high risk value for Availability when the threat type is a DDoS attack, accompanied by low values for Confidentiality and Integrity.

4.1.2 ML Problems Definition

According to the VERIS vocabulary [7] the generic term ”threat type” refers to threat action related features (action.* for the 1^{st} level and action.*.variety.Y for the 2^{nd} level - for details refer to [108]) while the term ”asset type” refers to asset variety related features (asset.variety for the 1^{st} level and asset.assets.variety.X - Y for the 2^{nd} level - for details refer to [109]) demonstrating the compromised assets during an incident. The term ”organization demographics” refers to organization size (victim.orgsize=Large, victim.orgsize=Small) and organization industry (e.g. victim.industry=Healthcare) related features (for details refer to [110]). These features are always included as inputs to the risk estimation as they are a priori known and constant demographics for a specific infrastructure. It should be noted that, more features could have been included in the problems, such as victim countries or number of employees and many more. Nevertheless, in order to achieve a satisfactory degree of generalization to many businesses and mostly ensure a sufficient number of VCDB observations without missing values for training ML models these combinations have been decided as the most appropriate. Therefore, taking the hierarchical structure of the dataset into account as previously described in Chapter 3, the generic risk estimation problem 1 (refer to Section 4.1.1) is matched to the two categories of probabilistic ML subtask groups of Formulas 4.4 and 4.5.

$$\begin{aligned}\mathbb{P}(\text{action.X} = 1 \mid \text{victim.orgsize.*, victim.industry.*,} \\ \text{asset.variety.*,asset.assets.variety.* - *})\end{aligned} \tag{4.4}$$

$$\mathbb{P}(\text{action.X} = 1,\ \text{action.X.variety.Y} = 1 \mid \text{victim.orgsize.*},\ \text{victim.industry.*},\ \text{asset.variety.*},\ \text{asset.assets.variety.* - *}) \tag{4.5}$$

The probability of formula 4.4 can be directly estimated by setting up the respective classification problem on VCDB. Regarding the probability of Formula 4.5, leveraging the chain rule of probability it results to Formula 4.6 which can be directly estimated by the estimated by soft classification problems on VCDB.

$$\mathbb{P}(\text{action.X.variety.Y} = 1) \mid \text{victim.orgsize.*},\ \text{victim.industry.*},\ \text{asset.variety.*},\ \text{asset.assets.variety.* - *},\ \text{action.X} = 1) \cdot \mathbb{P}(\text{action.X=1} \mid \text{victim.orgsize.*},\ \text{victim.industry.*},\ \text{asset.variety.*},\ \text{asset.assets.variety.* - *}) \tag{4.6}$$

Similarly, the generic risk estimation problem 2 relates to the two categories of probabilistic ML subtask groups of Formulas 4.7 and 4.8.

$$\mathbb{P}(\text{asset.variety.A} = 1 \mid \text{victim.orgsize.*},\ \text{victim.industry.*},\ \text{action.*},\ \text{action.*.variety.*}) \tag{4.7}$$

$$\mathbb{P}(\text{asset.variety.A} = 1,\ \text{asset.assets.variety.A-B} = 1 \mid, \text{victim.orgsize.*},\ \text{victim.industry.*},\text{action.*},\ \text{action.*.variety.*}) \tag{4.8}$$

The probability of formula 4.7 can be again directly estimated by setting up the respective classification problem on VCDB. Regarding the probability of Formula 4.8, leveraging the chain rule of probability it results to Formula 4.9 which can be directly estimated by the estimated by soft classification problems on VCDB.

$$\mathbb{P}(\text{asset.assets.variety.A - B} = 1 \mid \text{victim.orgsize.*},\ \text{victim.industry.*},\ \text{action.*},\ \text{action.*.variety},\ \text{asset.variety.A} = 1) \cdot \mathbb{P}(\text{asset.variety.A=1},\ \text{asset.assets.variety.A - B} = 1 \mid \text{victim.orgsize.*},\ \text{victim.industry.*},\ \text{action.*},\ \text{action.*.variety.*}) \tag{4.9}$$

Following, the risk estimation problem 3 relates to a unique ML subproblem group (4.10) as it does not dive to a deeper level hierarchy.

$$\mathbb{P}(\text{attribute.Z} = 1 \mid \text{victim.orgsize.*},\ \text{victim.industry.*},\ \text{action},\ \text{action.*.variety.*},\ \text{asset.variety.*},\ \text{asset.assets.variety.* - *}) \tag{4.10}$$

As the tabular (CSV) format of VCDB has been used, all of the described features are treated as one hot encoded variables meaning that for example action.variety corresponds to all ”children” columns action.Hacking, action.Malware, action.Social etc. with boolean (0,1) values defining the desired label. Such an approach is the main workaround to solve ML problems against hierarchically structured datasets. Additionally, at the same time avoids the one-hot-encoding step that would be required at later preprocessing stages.
The Formulas 4.6, 4.9, 4.10 can be decomposed to 5 probabilistic terms in total that reflect 5 groups of soft binary classification ML tasks. These ML tasks allow for realising all the desired probabilistic risk estimations. Depending on the varying values of (A, B), (X, Y), and Z each group can result to multiple classification problems. These values

mainly depend on the asset list of the infrastructure, the threat detection capabilities of its cybersecurity system (X, Y) and the data attributes examined (Z) respectively. For example in case A = "Server" it could be that B = "Database or Web Application", however B ≠ "User desktop or laptop" as this would require A = "User Dev" according to VERIS [7]). Similarly, in case X = "Malware" it could be that B = "Adware", however B ≠ "Phishing" as this would require A = "Social" according to VERIS). Finally Z can only take the values "Confidentiality", "Availability", "Integrity". At this point it is important to mention that such features are not mutually exclusive, that is an incident can contain both social and malware based actions leading to action.Malware = 1 and action.Social = 1. Similarly, both confidentiality or availability can be affected during the same incident leading to attribute.Confidentiality = 1 and attribute.Intergrity = 1 at the same time. For this reason it is rational to proceed with individual binary classification problems instead of multi-label ones at least for non deep learning approaches that are employed in the context of this dissertation. Regarding the star values, they also unfold to a group of boolean features based again on the VERIS vocabulary (e.g victim.orgsize.* = {"victim.orgsize.Small", "victim.orgsize.Large"}).

4.2 Experimentation Pipeline

As the business needs for risk assessment can vary among different infrastructure, the focus of this research has been the facilitation of conducting various experiments relating to the different ML problem groups. Such an approach allows for mass training, evaluation and comparison of multiple ML algorithms in an automated and easily configurable manner. In this context, the experimentation related ML stages, namely ETL, training and evaluation stage have been developed as modularized code components that are then combined as an MLOps pipeline to provide the desired results. More specifically the pipeline is able to accept any constructed model and perform the corresponding steps given some data without hardcoding anything, meaning that the code flow generalizes for any model and data (feature combinations). Thus, more experiments can be performed, leading to higher productivity and quicker identification of optimal solutions. In terms of validation / hyperparameter tuning a separate pipeline was created (available at the project's code repository [19]) but as it was proved to be inefficient compared to its added value to the project, it was removed and will not be included in this chapter. The components of the developed ML pipeline are described in the sections to follow.

4.2.1 ETL Stage

As it is necessary to allow the forming of various ML problems, characterized by different input / output combinations it is important that the ETL step allows for certain flexibility regarding the structure of the dataset to be promoted to the next steps of the pipeline. As already mentioned the tabular format of VCDB [97] has been decided as the most appropriate for ML tasks. Requiring as input the VCDB dataset, the task (see Section 4.1.2) and the specific target column of the ML problem, the main operations performed on this version of the dataset throughout the ETL step are the following:

- Timely sorting of the dataset.
- Extraction of predictors and target variable for the ML task.

- Removal of observations with missing values as they are inappropriate for ML modeling.
- Removal of entries that are irrelevant to the specific classification problem. For example, all entries where asset.variety.S=0 are irrelevant for the classification task seeking to estimate the probability of Formula 4.11 as this estimation is conditional to asset.variety.Server=1..

$$\begin{gathered}\mathbb{P}(\text{asset.variety.S - Server} = 1 \\ \text{action.*,action.*.variety.*, victim.orgsize.*, victim.industry.*})\end{gathered} \tag{4.11}$$

  Thus it becomes obvious that each model is built on datasets of varying size.
- Removal of entries that contain environmental actions as they are out of the scope of this project and there was a minor number of them in VCDB.
- Merging redundant columns that overlap between ”action.hacking.variety.*” and ”action.malware.variety.*” columns.
- Construction of a dataset ready to enter the ML stage of the pipeline consisting of two variables, namely the predictors and the binary target variable.

Therefore, each execution of the ETL step with the appropriate parameters results to an ML ready pandas ?? dataframe to enter the rest of the pipeline. The results of this process could form a feature store for the project by logging them as artifacts to the MLflow server, however this step has been avoided mainly for storage saving purposes. For more details the Github repository [19] and specifically the corresponding python file *etl.py* can be referred to.

4.2.2 Model Training Stage

This stage incorporates preprocessing and training in the stage step through a scikit-learn [91] pipeline [111] that combines one-hot-encoding (mainly for the ”victim.industry” feature which has been extracted from VCDB as string), optionally PCA [112] and finally the desired classifier / estimator. Currently, scaling is unnecessary as all values are categorical (0, 1 valued). As the training process initiates an exhaustive logging procedure takes place including model hyperparameters, metrics and training / data characteristics (e.g. positive / negative sample percentage, train / test size etc.). More details regarding the training stage can be found at the well documented script file *train.py* of the code repository [19]. During the training stage, various classical ML classification algorithms have been considered, namely LightGBM [10], Random Forest [11] K-Nearest Neighbors [9], Support Vector Machine [12], Logistic Regression [113], Gaussian Naive Bayes [14] with the purpose of examining and comparing their performance. The detailed results are provided in Section 4.3.

4.2.3 Evaluation Stage

The evaluation stage comprises the testing of trained models on the predefined test set. Given the temporal sorting of the dataset, the test set is being selected to contain the more recent observations, thus allowing to examine the capability of generalization of models to more recent observations. During evaluation all standard classification metrics [114] have been used such as Accuracy, Precision, Recall (TPR), F1-score (micro and macro),

AUC score (micro and macro). Nevertheless, due to the high cardinality of different ML problems which involved different dataset structures and sizes and most importantly different class balances the F1-macro metric was considered as the most strict and appropriate to adopt for a holistic and common to all problems approach, especially during the deployment stage (refer to Chapter 5). F1-macro is the harmonic mean of precision and recall, equally averaged on both classes of the problem. This metric harshly penalizes classifiers that tend to better predict the majority class, thus leading to more unbiased classifiers. On the contrary, other evaluation methods, such as accuracy and micro averages that penalize performance on the minority class proportionally to its cardinality compared to the whole dataset, can potentially lead to impressive but misleading results. At this point, it is should be noted once more that all ML models have been trained to predict probabilities as risk estimations require probabilistic outputs instead of binary ones. Consequently, it was necessary to seek for a metric that would judge the quality and calibration of results inside the [0, 1] scale as F1-score fails to do so. In this context, the Hosmer-Lemeshow statistic was introduced in the soft classification ML tasks which is known for effectively judging the absolute probabilistic outputs of a model in comparison with the ground truth. For instance, given a positive output label as ground truth, a probabilistic prediction of 0.3 will receive a higher penalty than on of 0.4 while F1-score will penalise them equally as erroneous predictions. The Hosmer-Lemeshow evaluation takes place through a statistical significance test resulting to a p-value that should preferably lower than a specific threshold. Usually this threshold ranges within 0.01 and 0.1 with a common value being 0.05. The purely technical details on the implementation of the evaluation stage can be found at the code repository [19] of the project and specifically the corresponding python file *evaluation.py*.

4.2.4 Pipeline Execution

For the execution of the pipeline that includes the stages of the previous sections the MLflow CLI is utilized and specifically the *mlflow run* command. The parameters / arguments that are accepted by the pipeline can be found below the ”train” entrypoint of the MLproject file (refer to A.3) and are the following:

- **task**: String of the ML task to solve (type: string, default: ’asset.variety’). Currently allowed values:
  - ”attribute”
  - ”asset.variety”
  - ”asset.assets.variety.S”
  - ”asset.assets.variety.M”
  - ”asset.assets.variety.U”
  - ”asset.assets.variety.P”
  - ”asset.assets.variety.T”
  - ”action”
  - ”action.error.variety”
  - ”action.hacking.variety
  - ”action.misuse.variety”
  - ”action.physical.variety”

  - ”action.malware.variety”
  - ”action.social.variety”

- **target**: String of the target variable of the ML problem. type: string, default: 'Server'. Currently allowed values depend on the task selection and are too many too be be included here.
- **algo**: String that defines the algorithm to train on the ML task. Currently allowed values: ”LGBM”, ”KNN”, ”SVM”, ”RF”, ”LR”, ”GNB”. (type: string, default: ”LGBM”)
- **hyperparams**: The hyperparameters, in a JSON formatted string, for the ML algorithm. (type: string, default: ''). The allowed values depend on the corresponding scikit-learn python API [91] for each one of KNN, SVM, RF, LR, GNB algorithms and on the LightGBM API [115] for LGBM. Example of allowed value for the SVM algorithm: '''C': 100, 'gamma': 'auto'”
- **imputer**: A string for the imputation method to follow for missing values, during the ETL step. Currently only ”dropnan” is supported which just removes observations with missing values. (type: string, default: ”dropnan”)
- **train_size**: Float argument that defines the training set size as a float factor of the whole dataset. Allowed values range in (0,1]. (type: float, default: 0.8)
- **split_random_state**: Integer argument that defines the random state for the split to training and testing sets. Currently this parameter is not active as shuffling is discouraged in order to keep the newest events as test set. Allowed values are all positive integers (type: int, default: 0)
- **pca**: This integer parameter defines the number of principal components to derive before training using the PCA method. 0 means no PCA. (type: int, default: 0)
- **merge**: String parameter that decides whether to merge the overlapping features between action.hacking.variety.<x> and action.hacking.variety.<y> during ETL. Allowed values: (true, t, on, yes, y | false, f, off, no, n). (type: str, default: ”yes”)
- **explain**: String parameter that decides whether to use SHAP to produce model explanations. Allowed values: (true, t, on, yes, y | false, f, off, no, n). (type: str, default: ”no”)
- **shap_data_percentage**: Float argument that defines the dataset fraction to be used for explanations. This should be small as the process is very resource consuming. Allowed values range in [0,1]. (type: float, default: 0.1)
- **shap_test_over_train_percentage**: Float argument that defines the training set fraction to be used as test set within SHAP explanation process. Allowed values range in [0,1]. (type: float, default: 0.3)

It is apparent that this setup provides certain flexibility to the data science processes as the experimentation can be fully configured producing easily various different experiments. The following command represents a valid example of producing an MLflow experiment leveraging the MLflow CLI:

```
mlflow run --experiment-name 'asset.variety' --entry-point train .
-P task="asset.variety" -P target="Server" -P algo="LGBM" -P train_size=0.8
-P split_random_state=44 -P n_folds=5 -P pca=2 -P explain="yes" -P merge="yes"
```

The execution of such commands led to the indicative results that are provided in the following section. Additionally, the automated execution of similar commands is the core of the deployment stage and will be further analysed in Chapter 5.

4.3 Experimentation Results

In an attempt to demonstrate the automation capabilities of the MLOps pipeline, the performance metrics of several interesting ML tasks / risk estimation problems are presented that have been studied, amongst others, throughout this dissertation. The analysis limits itself to two use cases, for demonstration purposes, as the main purpose of the dissertation was to build the MLOps framework along with a real-world application (see Chapter 5), allowing end users to automatically perform risk estimation tasks on VCDB on their own. Some more performance results are provided in Chapter 5 for the group of ML tasks that were included in the deployment phase of the RiskML application.

4.3.1 Support Vector Machines for Server Security Risk Prediction

The purpose of this experimentation was to compare the performance of different SVM variants, by altering the hyperparameters, in predicting the probability of a server pertaining to an organization to be targeted by a specific type of threat action. More specifically the ML problem comprises the estimation of the probability 4.12 through a soft binary classification problem.

$$\begin{aligned}\mathbb{P}(\text{asset.variety.S - Server} = 1 \mid \text{action.*, action.*.variety.*,} \\ \text{victim.orgsize.*, victim.industry.*})\end{aligned} \tag{4.12}$$

The MLflow CLI commands that were run to produce the desired results are structured as follows:

```
mlflow run --experiment-name 'asset.variety.Server' --entry-point train .
-P task='asset.variety' -P target='Server' -P algo='SVM'
-P hyperparams='{"C":<X>, "kernel":<Y>, "gamma":<Z>}'
```

where $X \in \{1, 10, 100\}$, $Y \in \{\text{"linear"}, \text{"rbf"}\}$ and $Z \in \{\text{"auto"}, \text{"scale"}\}$.

Regarding the visualization of results in the MLflow UI it is depicted in Figure 4.1. Further information regarding each one of the models can be explored by clicking on the specific run of interest.
A summary of the most significant content of 4.1, including hyperparameters and performance metrics of this setup are depicted in Table 4.1. It can be observed that the SVM algorithm with an rbf kernel, a penalty parameter of the error term (C) equal to 100 and the curvature parameter gamma set to 'auto' outperforms all the other SVM models and fortunately that result is unanimously confirmed by most performance metrics not leaving space for any doubts. The Hosmer Lemeshow statistic (HL) is 0.056 for

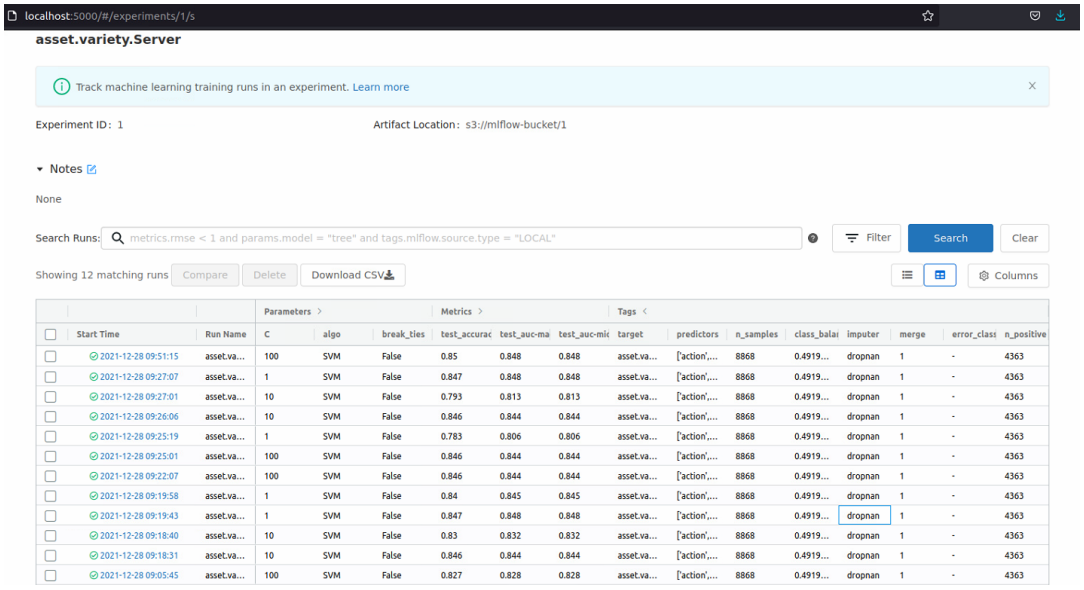

Figure 4.1: The MLflow UI having logged the SVM runs on the asset.variety.Server experiment.

the best SVM demonstrating that the predicted probabilities are acceptably calibrated. Regarding PCA, it was observed that it dramatically dropped performance down by near to 30% in all cases, so this option not included in the results.

Table 4.1: Evaluation of SVM models on test set for the "asset.variety.Server" experiment

Hyperparameters			Test Set Performance Metrics				
C	Gamma	Kernel	Accuracy	Auc	F1-macro	F1-micro	HL
1	auto	linear	0.847	0.848	0.839	0.847	0.048
1	scale	linear	0.847	0.848	0.839	0.847	0.093
1	auto	rbf	0.783	0.806	0.779	0.783	0.310
1	scale	rbf	0.840	0.845	0.833	0.840	0.130
10	auto	linear	0.846	0.844	0.837	0.846	0.043
10	scale	linear	0.846	0.844	0.837	0.846	0.123
10	auto	rbf	0.793	0.813	0.788	0.793	0.135
10	scale	rbf	0.830	0.832	0.822	0.830	0.054
100	auto	linear	0.846	0.844	0.837	0.846	0.122
100	scale	linear	0.846	0.844	0.837	0.846	0.038
100	auto	rbf	0.850	0.848	0.841	0.850	0.056
100	scale	rbf	0.827	0.828	0.818	0.827	0.120

4.3.2 Multiple Algorithms for Predicting the Risk of Data Availability Compromise

The second ML problem refers to the estimation of the risk of data availability loss -that is the probability of an incident resulting to downtime of a data serving digital asset or loss of a physical data source- during an incident given the performed action, the targeted asset and the victim demographics. Specifically the probability of the Formula

4.13 is being through a soft binary classification problem. Common reasons for loss of availability are lost / stolen media, voltage drops that result to server downtimes and even worse hacking attacks such as DoS or Ddos that most often flood servers with numerous malicious requests aiming to exhaust their resources and consequently lead to denial of service for legitimate users.

$$\begin{aligned}\mathbb{P}(\text{attribute.Availability} = 1 \mid \text{action.*, action.*.variety.*,} \\ \text{asset.variety.*, asset.variety.* - *, victim.orgsize.*, victim.industry.*})\end{aligned} \tag{4.13}$$

The MLflow CLI commands that need to be run to produce the desired results are structured as follows:

```
mlflow run --experiment-name 'attribute.Availability' --entry-point train .
-P task='attribute' -P target='Availability' -P algo=<X>
```

where $X \in \{"LGBM", "KNN", "SVM", "RF", "LR", "GNB"\}$.

Regarding the visualization of results in the MLflow UI it is illustrated in Figure 4.2. Further information regarding each one of the models can be explored by clicking on the specific run of interest.

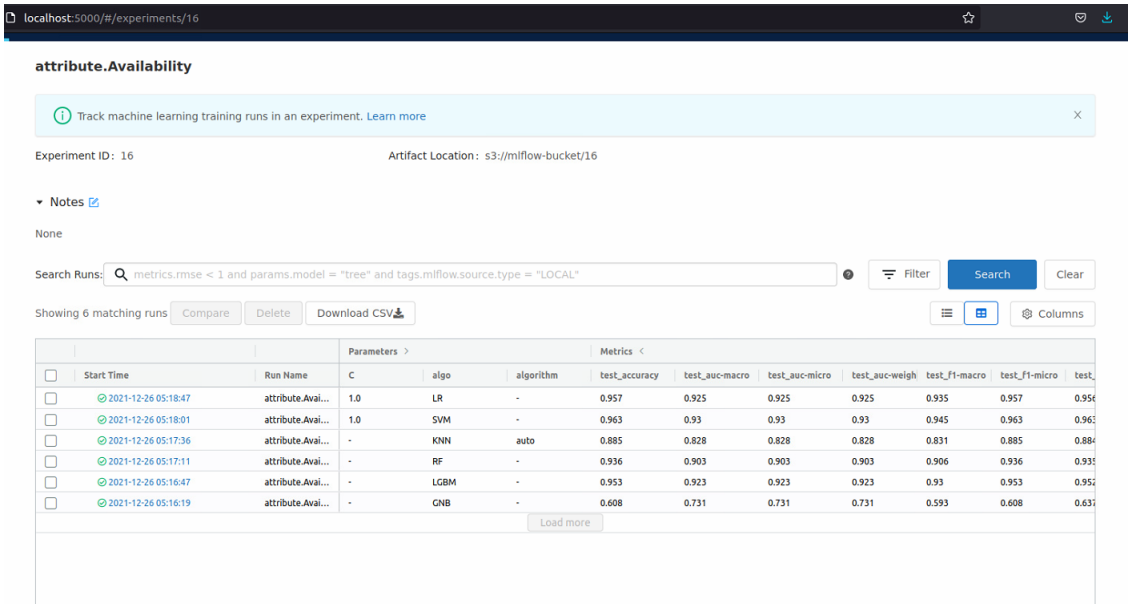

Figure 4.2: The MLflow UI having logged the multi-algorithm runs on the 'attribute.Availability' experiment.

A summary of the most significant content of 4.2, including hyperparameters and performance metrics of this setup are depicted in Table 4.2. It is worth mentioning that the Naive Bayes algorithm expectedly fails to perform well due to high correlations amongst features (hierarchical nature) given that the algorithm calculates a separate likelihood for each one of the features and then multiplies relying on the totally false assumption of feature independence. The performance of Logistic Regression is quite impressive for a baseline technique, though the respective HL value is pretty high. SVM and LGBM offer the best results with SVM outperforming the rest of the algorithms regarding the

F1-macro metric. Neverthelss, LGBM seems to be more balanced as it exhibits well calibration (in terms of probabilitistic predictions) combined with satisfying classification performance.

Table 4.2: Evaluation of various ML models trained on the attribute.Availability experiment. (Default hyperparameter values have been used)

Algorithm	Hyperparameters	Accuracy	AUC	F1-macro	F1-micro	HL
LR	-	0.957	0.925	0.935	0.957	0.191
SVM	kernel: 'rbf'	0.963	0.930	0.945	0.963	0.158
KNN	n_neighbors: 5	0.885	0.828	0.831	0.885	0.308
RF	n_estimators: 100	0.936	0.903	0.906	0.936	0.091
LGBM	n_estimators: 100	0.953	0.923	0.930	0.953	0.113
GNB	-	0.608	0.731	0.593	0.608	0.690

4.3.3 Model Interpretability Concepts

As the explainability of ML models is more than valuable for most sectors (e.g. healthcare) several frameworks have been developed to produce interpretations of ML algorithms. One of them is SHAP [116]. MLflow has been integrated with the SHAP implementation [94] and this intergration has also been included in the automation framework to store interpretation results as model artifacts. Running the following command a sample interpretation for the LightGBM training can be obtained for the "attribute.Availability" task:

```
mlflow run --experiment-name 'attribute.Confidentiality' --entry-point train .
-P task='attribute' -P target='Availability' -P algo='LGBM' -P explain='yes'
```

The results of the explanation process are illustrated in Figure 4.3 where the importance of all features regarding each class is depicted in descending order. It can be observed that the top 3 features heavily affecting the model's decisions (equally for each class) are similar to the ones that have already been mentioned previously, namely lost or stolen assets and hacking attacks (e.g. DoS).

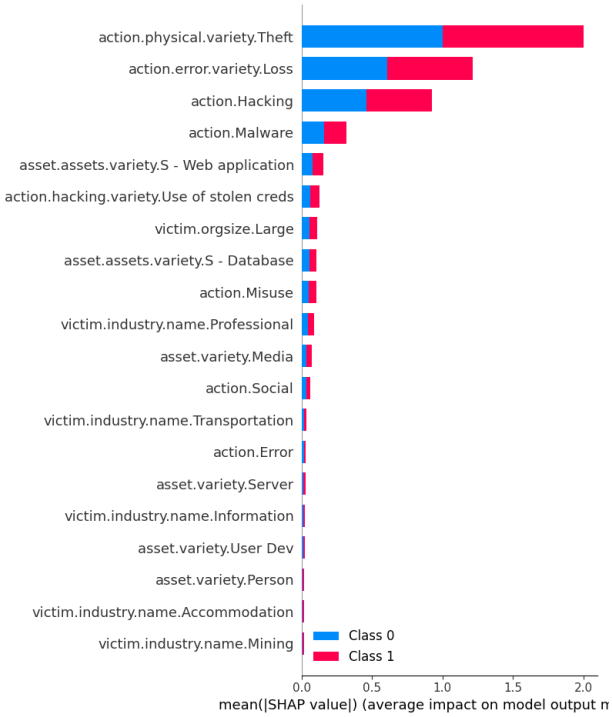

Figure 4.3: Feature importance bar plot produced by SHAP based on the LightGBM algorithm for the experiment "attribute.Availability".
"

Chapter 5

Deployment Stage - The RiskML application

This section covers the "Deployment and Monitoring" stage of the ML lifecycle through the deployment of RiskML: a real-world application that automatically implements the pipelines described in Chapter 4 following the basic principles of MLOps at a stage 2 automation level, namely continuous model delivery (refer to Chapter2). RiskML is an ML powered state-of-the-art software application that aims at providing threat related data-driven estimations through ML models that are being trained on the VERIS Community Database (VCDB), thus capturing the status quo of cybersecurity incidents / data breaches in the enterprise world. The training scenarios of the deployment can be defined by the end-user through configuration files. RiskML is not a stand alone service but can be used to serve the desired risk predictions to other programmatical agents (e.g. cyber-risk assessment agent as mentioned in Chapter 4). That has been the case as well in the context of SPHINX Project where RiskML has been intergrated within a real time risk assessment module to provide on-demand asset-related predictions. RiskML consists of a microservice-based architecture that is built around MLflow [33] and Docker [15] and follows. The implementation code of RiskML is publically available on Github [19].

5.1 Technological Stack

The RiskML service is developed as a machine learning application combining multiple services whose main core is MLflow. Other technologies that are being used in RiskML are the following:

- MinIO [17] that is used as storage for the saved models by MLflow. A local container is used, however MinIO is an S3 [117] compatible storage unit that can be extended to the cloud, also offering distributed storage options.
- Postgresql [118] for storing the MLflow experiment tracking results (run details, model parameters and tags and performance metrics).
- Various python libraries for machine learning including numpy [119], pandas [120], scikit-learn [91], and lightgbm [115].
- Python libraries such as connexion [121] and swagger-ui-bundle [122] that were used to create the API / UI features that came additionally to MLflow.
- Verispy [123] for effectively handling and extracting analytics from VCDB during the pre-processing stages of the ML lifecycle.
- Docker that allows for a modular approach and thus the deployment different functions of the application in different containers as separate interacting services. Docker-compose [124] was also utilized to coordinate and automate their deployment.

5.2 Architectural Design and Core Functionalities

A technology and deployment-aware architectural diagram of RiskML is illustrated in Figure 5.1, where the coordination amongst different technologies becomes clear. Five different Docker containers are being deployed as demonstrated in the schema to effectively implement the functionalities of RiskML. All containers are essential for all functional stages of RiskML, except for the minIO client container that it is just used once at the beginning of deployment to initialize the bucket storage of the minIO server that is used to store the trained models and other artifacts at the backend of MLflow.

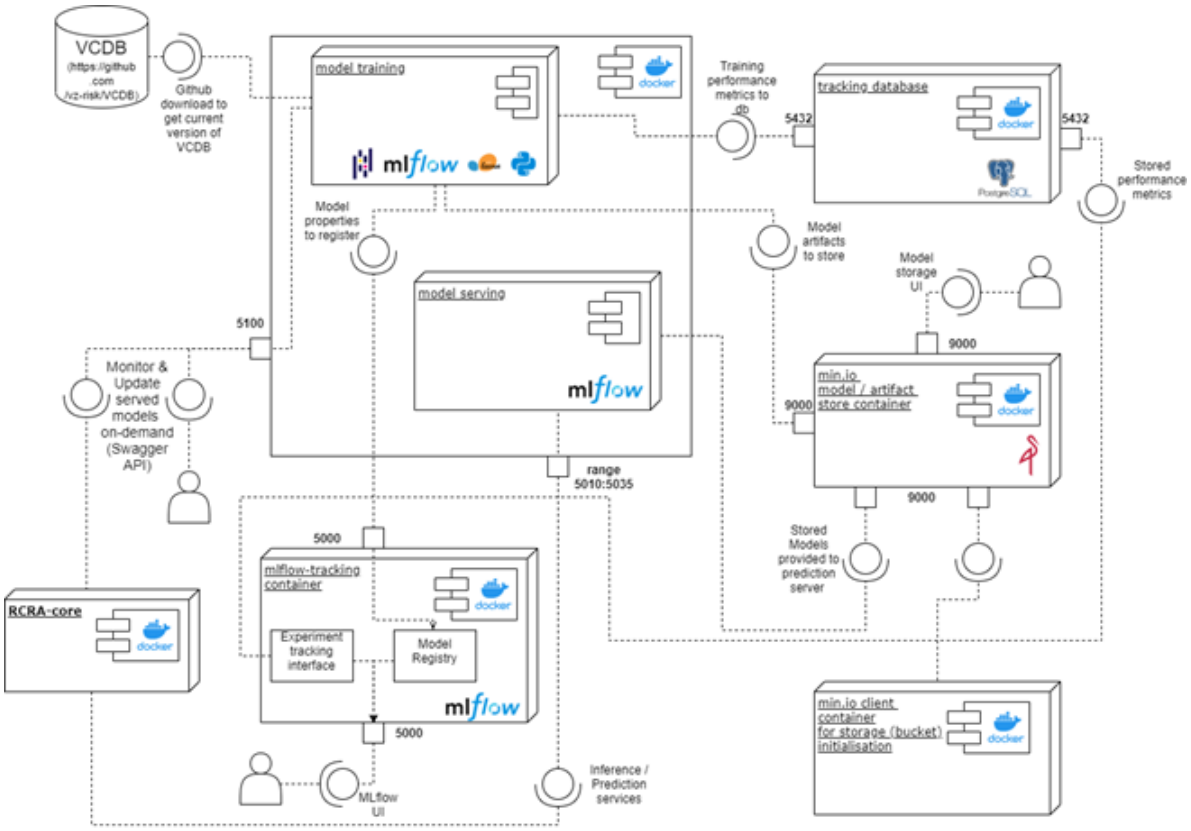

Figure 5.1: The technology oriented architecture of RiskML

The docker-compose.yml file required for a similar deployment is provided in in Appendix A.2.
The core functionalities of RiskML are the following:

1. An automated framework of training ML models for risk estimations. Before deployment, the end-user can opt for the models of their preference by simply modifying the appropriate JSON configuration file that is formed as in Appendix A.1. Specifically, the end-user can set the serving port (e.g 5010) and target variable of each model following the vocabulary of the tabular version of VCDB [97] proposed by VERIS [7] (e.g. asset.variety.Server), while also selecting the desired ML algorithms to be trained.

2. Some technical details concerning the training setup and implementation are the following:

- Port selection can vary amongst any available ports except 5000 which is already in use by the experiment tracking server.
- Model identifiers can vary amongst all VCDB features that start with asset.variety, asset.assets.variety or are of the form action.X action.X.variety or attribute.Confidentiality, attribute.Integrity, attribute.Availability.
- The training pipeline starts right after the deployment of the RiskML application. The candidate algorithms are automatically trained on each target variable and compared based on their F1-score performance. The Hosmer-Lemeshow statistic has been also used as an on/off criterion of accepting and deploying a model to production (a p-value smaller than 0.05 is required, otherwise the previous model is kept). After the end of the training stage, the best model is deployed in production to its corresponding port, which means that it can function as risk prediction server ready to serve requests from the central cyber-risk assessment agent that is also deployed within the healthcare environment.
- Taking into account the hierarchical structure of the dataset, a consumer of the models that requires a 2^{nd} level prediction (for instance asset.assets.variety.S - Database) first needs to get the corresponding first level prediction (asset.variety.Server) and then multiply both results to get a final probabilistic estimation (refer to Equation 4.9). This is the result of the fact that 2^{nd} level models are explicitly trained on VCDB instances, whereas the first level is activated (e.g., the model referring to asset.assets.variety.S - Database is only being trained on VCDB instances where asset.variety.Server=1). This technique of hierarchically structured local classifiers and specifically of training one local classifier per parent node is extremely useful and "clean" in hierarchical classification tasks.

3. An easy-to-use administrative UI for real-time monitoring of ML training experiments, namely visualization of model parameters, hyperparameters, performance metrics and training times of all different models and algorithms trained (Figure 5.2 and Figure 5.3). Registered models (that can be in production or not) can also be monitored, as demonstrated in Figure 5.4. End-user defined queries are also permitted for search amongst experiments and registered models. This UI will be mainly used for administrative purposes and not every-day end-user interaction.

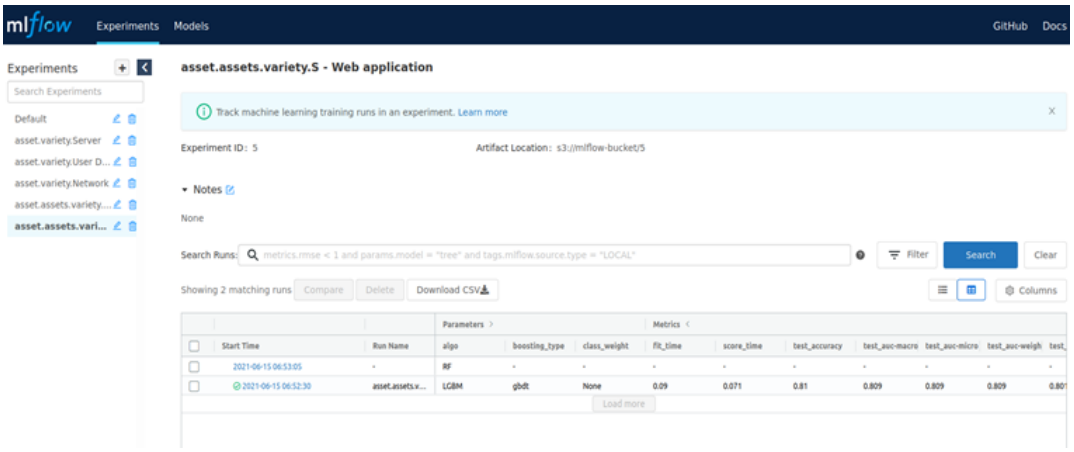

Figure 5.2: The MLflow tracking UI

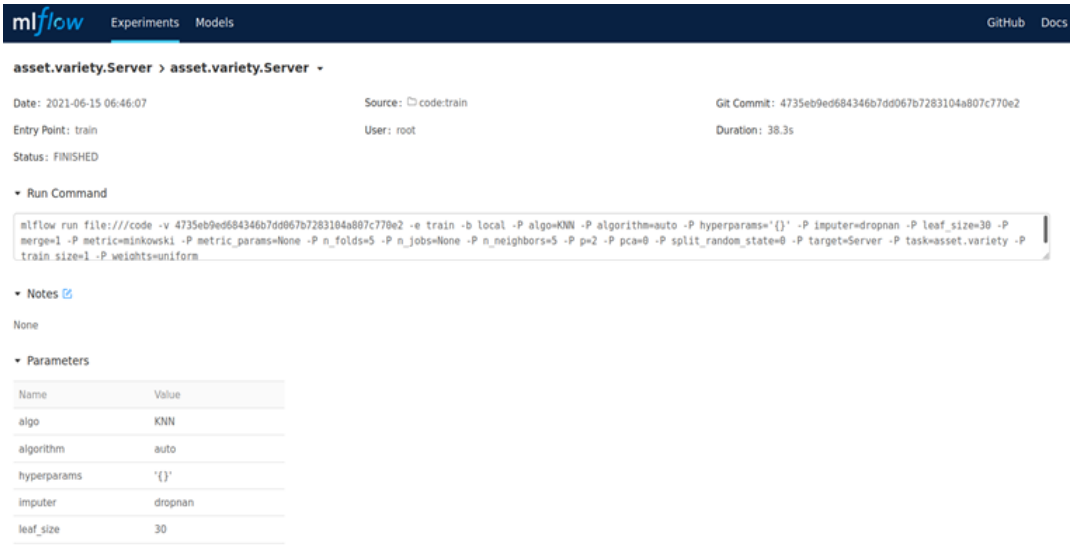

Figure 5.3: Tracking a specific model inside the MLflow tracking UI

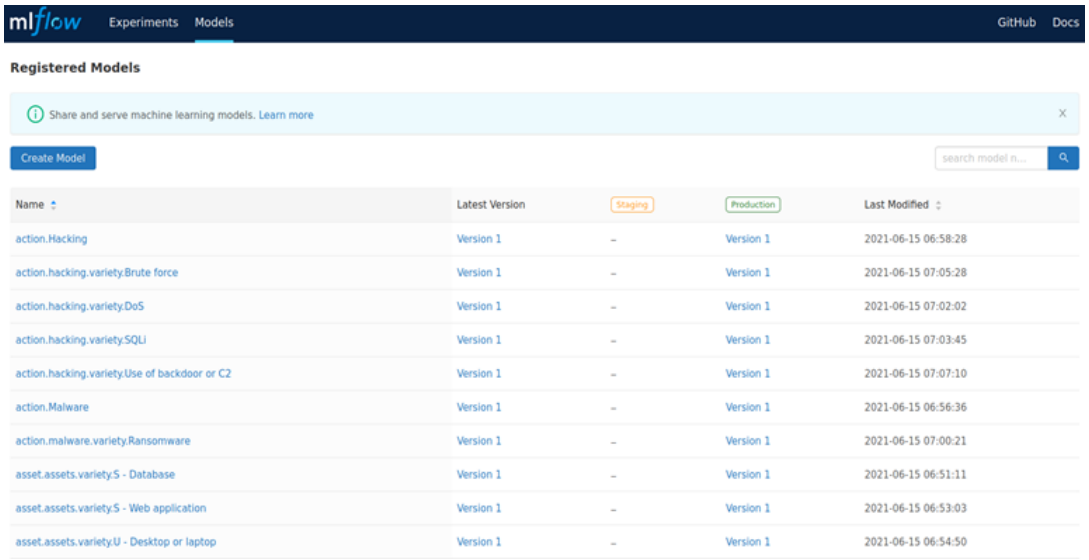

Figure 5.4: Tracking a specific model inside the MLflow tracking UI

4. A REST API was built for monitoring model serving ports and most importantly database updating as well as model retraining on-demand. Figure 5.5 illustrates a simple test UI (Swagger UI [122]) for this API and its detailed specifications are presented in the next section. This API is typically consumed by programmatical clients however human agents (administrative end-users) can also trigger its functions and potentially use it for monitoring and retraining purposes (manual trigger of training pipeline according to [32]).

5. In case the end-user desires to dive deeper into the details of the ML deployment and further parameterize it, such as for example change training and test proportions, modify the random states of the splits and even produce interpretability plots with SHAP [116] or employ the PCA [112] dimensionality reduction technique before training they should alter accordingly the MLproject file (Appendix A.3) of the project repository [19] before composing up the Docker deployment.

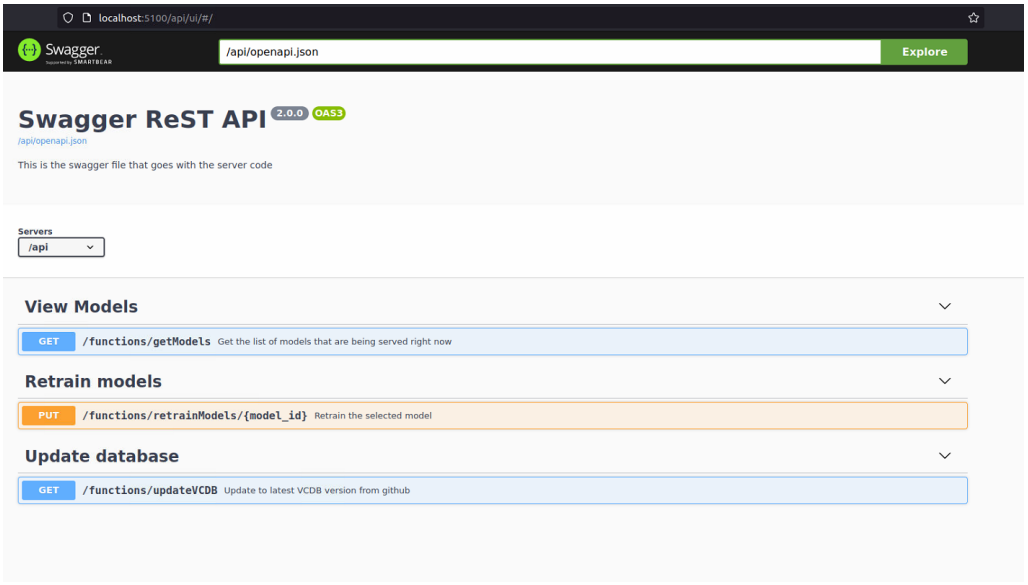

Figure 5.5: The Swagger UI that is responsible for model serving monitoring, database updating and model retraining

Alternatively, individual and totally configurable experiments (similar to those of Chapter 4) can be triggered and added on top of the initial deployment results from inside the container of the central service of the deployment (mlflow_code container) using the MLflow CLI. [125].

6. In order to consume the served models, any programmatical or human client can perform HTTP requests against the port serving the desired model. An example of terminal command is the following:

```
curl http://0.0.0.0:5011/invocations -H 'Content-Type: application/json'
-d @/input_examples/asset.variety.Server/model/input_example.json
```

The correct structure of the input can be found at the respective folder of the project's code repository.

The tables of Appendix B exhaustively list and document all interface endpoints (APIs / UIs) provided by the RiskML application.

5.3 Deployment and Automation Backend Concepts

The purpose of RiskML is to automate the training procedures of Chapter 3 for multiple ML tasks and ML algorithms, automatically identifying and deploying the best one for each task, leading to a production setup serving ML models to be consumed by risk assessment agents. To achieve that, based on the information derived from the previously mentioned configuration files, a set of shell scripts are automatically generated (refer to *shell_scripts/shell_script_composer.py* at the code repository) during runtime. These shell scripts interact with the MLflow CLI to automatically train and serve models according to the end-user required ML experiments (before deployment, via configuration files) after comparing them. The training and evaluation processes are based on the pipeline

steps that were defined throughout the experimental stage (Chapter 4) of the project (*mlflow run* CLI). In terms of model deployment, every selected algorithm is trained on every ML task. Then, as long as it satisfies the specific model deployment conditions - namely F1-score is higher than the previous models and Hosmer-Lemeshow p-value <= 0.05 - the model is automatically labeled as "Production", while the previous one is labeled as "Archived". Finally, the model replaces the previous one in production at the same port, leveraging the *mlflow serve* CLI locally. More details can be found at the *utils.py* file of the code repository. Similarly, the services offered by the Swagger API, during deployment, are also implemented by similar shell scripts (refer to *shell_scripts/* and *api/* folders of the code repository) that allow for monitoring model serving, stopping it and updating the dataset as well as retraining ML models on-demand.

5.3.1 Deploying RiskML

The installation process of RiskML is pretty simple, as docker and docker-compose assume the responsibility of initializing the services, training the models and finally serving them locally. To start the deployment, the end-user is required to follow the steps below:

1. Clone the Github repository:
   git clone https://github.com/pelekhs/RiskML.git
2. Define the desired deployment configuration within *deployment_config.yml*.
3. Run the following inside the root directory of the project:
   docker-compose up –build

5.4 Performance Indicators in a Real Deployment Scenario

This section lists the performance indicators for the sample RiskML deployment scenario that occurs for the specific configuration files of Appendices A.1 (deployment_config.yml) and A.2 (docker-compose.yml) that can also be found in the root directory of the project's repository [19]. The specific configuration has been utilized in a real deployment scenario that serves the cybersecurity needs and matches the asset list of a replicated hospital infrastructure, in the context of the SPHINX EU project. In this environment, RiskML predictions cannot entirely serve the mission of a cyber-risk assessment agent per se as they are generic, long-term and business world oriented, ignoring the current state of the monitored infrastructure. However, they can be used in an ancillary / consulting way. That was exactly the case in SPHINX scenario where RiskML serves predictions related to the healthcare domain to a programmatical risk assessment agent. The latter, fuses the results of this prediction with further risk assessment models and other internal parameters of the system in defence (e.g. alerts from IDS systems) that the RiskML models are obviously agnostic of, as they only deal with the generic state of the business world as represented every time by VCDB. The result is a real-time and targeted cyber-risk factor is produced for each asset of the hospital infrastructure.
Using an AMD Ryzen 9 4900H processor (single core training), the services need about 32 minutes to build including the training of 3 different algorithms (KNN, LightGBM and Random Forest were selected for this use case) for each target and successfully serving the 16 resulting models namely:

- asset.variety.Server
- asset.variety.User Dev
- asset.variety.Network
- asset.assets.variety.S - Database
- asset.assets.variety.S - Web application
- asset.assets.variety.U - Desktop or laptop
- action.Malware
- action.Hacking
- action.malware.variety.Ransomware
- action.hacking.variety.DoS
- action.hacking.variety.SQLi
- action.hacking.variety.Brute force
- action.hacking.variety.Use of backdoor or C2
- attribute.Confidentiality
- attribute.Integrity
- attribute.Availability

On average, the whole training process of one algorithm on a specific target is ranging from 30 to 40 seconds to complete, depending on the number and type features required to train the respective models. Table 5.1 lists the achieved F1 scores (range [0,1]) of the algorithm that managed to reach the deployment stage satisfying the deployment critera of section 5.3. The p-value (range [0,1]) of the Hosmer-Lemeshow statistic is also reported to monitor the calibration of predictions inside the probabilistic range [0,1].
The LightGBM algorithm generally outperformed the other two at all ML tasks and managed to reach the deployment stage for achieving a better balance between F1-score and the HL p-value. Results are far from satisfying for several tasks such as asset.assets.variety.U - Desktop or laptop or action.hacking.variety.DoS and usually this due to the low support of the positive class, that prevents the learning algorihtm from fitting appropriately.

Table 5.1: Evaluation of ML models trained and served on the sample deployment resulting from configuration the files included in Appendices A.1 and A.2.

ML task	Best algorithm	F1-macro	Hosmer-Lemeshow p-value
asset.variety.Server	LightGBM	0.858	0.201
asset.variety.User Dev	LightGBM	0.778	0.228
asset.variety.Network	LightGBM	0.856	0.023
asset.assets.variety.S - Database	LightGBM	0.737	0.007
asset.assets.variety.S - Web application	LightGBM	0.801	0.161
asset.assets.variety.U - Desktop or laptop	LightGBM	0.707	0.801
action.Malware	LightGBM	0.761	0.172
action.Hacking	LightGBM	0.797	0.076
action.malware.variety.Ransomware	LightGBM	0.793	0.412
action.hacking.variety.DoS	LightGBM	0.514	0.052
action.hacking.variety.SQLi	LightGBM	0.507	0.168
action.hacking.variety.Brute force	LightGBM	0.859	1.084x10-7
action.hacking.variety.Use of backdoor or C2	LightGBM	0.912	0.055
attribute.Confidentiality	LightGBM	0.771	0.007
attribute.Integrity	LightGBM	0.870	0.044
attribute.Availability	LightGBM	0.911	0.113

Chapter 6

Conclusions and Future Work

At the fist part of this this thesis work, an extensive background of machine learning based risk prediction research approaches has been presented along with current state-of-the-art of Machine Learning Operations (MLOps) [33] software methodologies and solutions.
Given that the Veris Community Database (VCDB) [1] is at the core of this work, an extensive exploratory data analysis has been conducted on the dataset focusing on thoroughly describing its structure. Additionally, it enabled the extraction of useful correlations and analytics amongst its features and also the the exploration of the current status quo in cybersecurity. A bit more emphasis has been put on the healthcare domain as the main use case of the product application was deployed in a relevant environment.Some of the main conclusions of this analysis are the following:

- The number of events recorded in VCDB is declining, without that necessarily implying a real global decline, but a decline in registrations in the specific dataset. However, the Covid-19 pandemic has led to a significant increase in 2020.
- The number of cybersecurity incidents does not necessarily correspond to the number of breaches recorded per year. On the contrary, this indicator seems to exhibit a constantly increasing trend with a global peak in 2020. This means that the impact per incident / cyberattack is increasing significantly.
- The public sector and the health sector seem to be the most affected, mainly due to human errors. It is speculated that this is because the first one is usually characterized by limited security protocols and staff training. Respectively, the second seems to be the most attractive for adversaries as the electronic infrastructures there are full of highly sensitive and valuable data.

The main experimental work of this thesis project is the implementation of an automated machine learning pipeline, adopting the current state-of-the-art in MLOps. The purpose of this pipeline is the automation of multiple risk prediction tasks, derived from VCDB, using MLflow [8] and training multiple machine learning algorithms (LightGBM [115] , K-nearest neigbours [9], Support Vector Machines [12], Logistic Regression [13], Random Forest [11]). The pipeline is modularized in ETL (Extract-Transform-Load), preprocessing & training, and evaluation steps. The risk prediction problems are decomposed to 5 groups of probabilistic binary classification tasks relating to the compromised assets, threat actions and the demographics of the victim. The tasks are formed as follows:

$$\mathbb{P}(\text{action.X} = 1 \mid \text{victim.orgsize.*, victim.industry.*, asset.variety.*, asset.assets.variety.* - *}) \tag{1}$$

$$\mathbb{P}(\text{action.X} = 1, \text{action.X.variety.Y=1} \mid \text{victim.orgsize.*, victim.industry.*, asset.variety.*, asset.assets.variety.* - *}) \tag{2}$$

$$\mathbb{P}(\text{asset.variety.A} = 1 \mid \text{,victim.orgsize.*, victim.industry.*, action.*, action.*.variety.*}) \tag{3}$$

$$\mathbb{P}(\text{asset.variety.A} = 1 \text{ , asset.assets.variety.A - B} = 1 \mid \text{victim.orgsize.*, victim.industry.*, action.*, action.*.variety.*}) \tag{4}$$

$$\mathbb{P}(\text{attribute.Z} = 1 \mid \text{ , victim.orgsize.*, victim.industry.*, action, action.*.variety.*, asset.variety.*, asset.assets.variety.* - *}) \tag{5}$$

The ontology is derived from the VERIS framework. Extensive results have been provided for two specific experiments as a proof of concept, namely:

- **Support vector machines for server security risk prediction**, where the SVM algorithm with an rbf kernel, a penalty parameter of the error term (C) equal to 100 and the curvature parameter gamma set to 'auto' outperforms all the other SVM models
- **Multiple algorithms for prediction of the risk of data availability loss**, where LightGBM seems to be the most balanced amongst others as it exhibits well calibration (in terms of probabilitistic predictions) combined with satisfying classification performance.

Leveraging the implemented machine learning pipeline, the final product of the project is a real-world Python [90] application intended for data scientists, security analysts and organization IT staff. The RiskML application enables the automated and mass deployment of machine learning experiments using various ML algorithms trained on VCDB to perform the abovementioned risk prediction tasks. Following, it allows for serving them in production for consumption by mainly programmatical clients such as risk assessment agents. Additionally, monitoring of the deployment and the deployed machine learning models is enabled via dedicated user interfaces. The RiskML architecture is structured on the coordination several state-of-the art modularized solutions such as MLflow, Docker [15] and MinIO [17]. A validation of the deployment of RiskML has taken place in the context of SPHINX H2020 project that has been deployed for the needs of holistic cybersecurity toolkit aimed at the healthcare domain. The results of this deployment scenario showed that the LightGBM algorithm perfoms on average best on all related machine learning classification tasks (both in terms of classification scoring and probability calibration).

In terms of potential future work the following are proposed as interesting research directions:

- The machine learning models could be extended so that one model could predict the probabilities for multiple targets (such as for example of the form action.* which decomposes to action.Hacking, action.Malware, action.Social etc.). Given that such targets are not mutually exclusive along with the inability of classical machine learning techniques to handle them as such, deep learning techniques could be incorporated, implementing multitask approaches. Such approaches usually employ logit binary cross entropy loss functions, instead of softmax, for predicting multiple classification labels at once.

- Sophisticated imputation techniques can be leveraged to prevent the removal of dataset entries with missing values, thus leading to better exploitation of the VCDB dataset.
- It would be valuable to consider methods that resolve the class imbalance problem that often occurs for several machine learning tasks and significantly drop the macro averaged performance metrics. Oversampling techniques such as the SMOTE algorithm are a good starting point for such future work [126].
- The scheduled or even real-time data ingestion instead of manual updating of VCDB would be time-saving for the end-user and would also contribute to the models being up-to-date. Therefore, such an upgrade is strongly recommended for RiskML.
- Automatic model retraining, based on the performance of the deployed models in production, can be considered as it would better serve the quality of the served models.
- The ingestion of more data sources (e.g. Hackmaggedon incident database [2]) in the pipeline except for VCDB would further widen the approach leading to more generalized results.
- Tools such as Kubernetes [64] and Nginx [127] would lead to the enhancement of the scaling and the security of the deployment along with implementation of authentication methods for accessing the machine learning services.
- RiskML could be also validated in more production environments except for SPHINX and the healthcare domain.

References

[1] *The veris vommunity database (vcdb)*. [Online]. Available: http://veriscommunity.net/vcdb.html (visited on 12/20/2021).

[2] *HACKMAGEDDON – Information Security Timelines and Statistics*. [Online]. Available: https://www.hackmageddon.com/ (visited on 12/20/2021).

[3] AshmoreRob, CalinescuRadu, and PatersonColin, "Assuring the Machine Learning Lifecycle," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, May 2021. DOI: 10.1145/3453444. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3453444.

[4] Y. Liu, A. Sarabi, J. Zhang, P. Naghizadeh, M. Karir, M. Bailey, and M. Liu, "Cloudy with a chance of breach: Forecasting cyber security incidents," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 1009–1024.

[5] A. Sarabi, P. Naghizadeh, Y. Liu, and M. Liu, "Prioritizing security spending: A quantitative analysis of risk distributions for different business profiles.," in *WEIS*, 2015.

[6] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraş, "The dropper effect: Insights into malware distribution with downloader graph analytics," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1118–1129.

[7] *The VERIS Framework*. [Online]. Available: http://veriscommunity.net/ (visited on 12/27/2021).

[8] *MLflow*. [Online]. Available: https://mlflow.org/ (visited on 12/20/2021).

[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, ISSN: 1557-9654. DOI: 10.1109/TIT.1967.1053964.

[10] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," Tech. Rep. [Online]. Available: https://github.com/Microsoft/LightGBM..

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 08856125. DOI: 10.1023/A:1010933404324. [Online]. Available: https://link.springer.com/article/10.1023/A:1010933404324.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, ISSN: 0885-6125. DOI: 10.1007/bf00994018. [Online]. Available: https://link.springer.com/article/10.1007/BF00994018.

[13] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression*. Springer, 2002.

[14] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, 2001, pp. 41–46.

[15] *Empowering App Development for Developers | Docker*. [Online]. Available: https://www.docker.com/ (visited on 12/24/2021).

[16] *OpenAPI Specification - Version 3.0.3 | Swagger*. [Online]. Available: https://swagger.io/specification/ (visited on 12/30/2021).

[17] *MinIO | High Performance, Kubernetes Native Object Storage*. [Online]. Available: https://min.io/ (visited on 12/25/2021).

[18] *Sphinx-project*. [Online]. Available: https://sphinx-project.eu/ (visited on 12/20/2021).

[19] *pelekhs/RiskML*. [Online]. Available: https://github.com/pelekhs/RiskML (visited on 12/25/2021).

[20] J. Tully, J. Selzer, J. P. Phillips, P. O'Connor, and C. Dameff, "Healthcare Challenges in the Era of Cybersecurity," *https://home.liebertpub.com/hs*, vol. 18, no. 3, pp. 228–231, Jun. 2020. DOI: 10.1089/HS.2019.0123. [Online]. Available: https://www.liebertpub.com/doi/abs/10.1089/hs.2019.0123.

[21] R. Altawy and A. M. Youssef, "Security Tradeoffs in Cyber Physical Systems: A Case Study Survey on Implantable Medical Devices," *IEEE Access*, vol. 4, pp. 959–979, 2016. DOI: 10.1109/ACCESS.2016.2521727.

[22] L. Zobal, D. Kolař, and R. Fujdiak, "Current State of Honeypots and Deception Strategies in Cybersecurity," *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops*, vol. 2019-Octob, Oct. 2019. DOI: 10.1109/ICUMT48472.2019.8970921.

[23] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity 2019 2:1*, vol. 2, pp. 1–22, 1 Jul. 2019, ISSN: 2523-3246. DOI: 10.1186/S42400-019-0038-7. [Online]. Available: https://link.springer.com/articles/10.1186/s42400-019-0038-7%20https://link.springer.com/article/10.1186/s42400-019-0038-7.

[24] G. González-Granadillo, S. González-Zarzosa, and R. Diaz, "Security information and event management (siem): Analysis, trends, and usage in critical infrastructures," *Sensors 2021, Vol. 21, Page 4759*, vol. 21, p. 4759, 14 Jul. 2021. DOI: 10.3390/S21144759. [Online]. Available: https://www.mdpi.com/1424-8220/21/14/4759/htm%20https://www.mdpi.com/1424-8220/21/14/4759.

[25] A. Applebaum, D. Miller, B. Strom, H. Foster, and C. Thomas, "ANALYSIS OF AUTOMATED ADVERSARY EMULATION TECHNIQUES," DOI: 10.5555/3140065. [Online]. Available: http://www.dlvsystem.com/k-planning-system/.

[26] M. Vasilescu, L. Gheorghe, and N. Tapus, "Practical malware analysis based on sandboxing," *Proceedings - RoEduNet IEEE International Conference*, Nov. 2014. DOI: 10.1109/ROEDUNET-RENAM.2014.6955304.

[27] S. Karagiannis, E. Magkos, C. Ntantogian, and L. L. Ribeiro, "Sandboxing the cyberspace for cybersecurity education and learning," pp. 181–196, Sep. 2020. DOI: 10.1007/978-3-030-66504-3_11. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-66504-3_11.

[28] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access*, vol. 6, pp. 35 365–35 381, May 2018. DOI: 10.1109/ACCESS.2018.2836950.

[29] H. Liu and B. Lang, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey," *Applied Sciences 2019, Vol. 9, Page 4396*, vol. 9, no. 20, p. 4396, Oct. 2019. DOI: 10.3390/APP9204396. [Online]. Available: https://www.mdpi.com/2076-3417/9/20/4396/htm%20https://www.mdpi.com/2076-3417/9/20/4396.

[30] I. M. M. Matin and B. Rahardjo, "Malware Detection Using Honeypot and Machine Learning," *2019 7th International Conference on Cyber and IT Service Management, CITSM 2019*, Nov. 2019. DOI: 10.1109/CITSM47753.2019.8965419.

[31] V. Mehta, P. Bahadur, M. Kapoor, P. Singh, and S. Rajpoot, "Threat prediction using honeypot and machine learning," *2015 1st International Conference on Futuristic Trends in Computational Analysis and Knowledge Management, ABLAZE 2015*, pp. 278–282, Aug. 2015. DOI: 10.1109/ABLAZE.2015.7155011.

[32] S. Alla and S. K. Adari, "What Is MLOps?" *Beginning MLOps with MLFlow*, pp. 79–124, 2021. DOI: 10.1007/978-1-4842-6549-9_3. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4842-6549-9_3.

[33] Sridhar, A. Suman, K. Adari, S. Alla, and S. K. Adari, "What Is MLOps?" *Beginning MLOps with MLFlow*, pp. 79–124, 2021. DOI: 10.1007/978-1-4842-6549-93. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4842-6549-9%5C_3.

[34] N. Sun, J. Zhang, P. Rimba, S. Gao, L. Y. Zhang, and Y. Xiang, "Data-Driven Cybersecurity Incident Prediction: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1744–1772, Apr. 2019, ISSN: 1553877X. DOI: 10.1109/COMST.2018.2885561.

[35] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *computers and security*, vol. 77, pp. 578–594, 2018.

[36] X. Pan, X. Wang, Y. Duan, X. Wang, and H. Yin, "Dark hazard: Learning-based, large-scale discovery of hidden sensitive operations in android apps.," in *NDSS*, 2017.

[37] G. Lin, J. Zhang, W. Luo, L. Pan, Y. Xiang, O. De Vel, and P. Montague, "Cross-project transfer representation learning for vulnerable function discovery," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3289–3297, 2018.

[38] L. Bilge, Y. Han, and M. Dell'Amico, "Riskteller: Predicting the risk of cyber incidents," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1299–1311.

[39] J. Zhang, Z. Durumeric, M. Bailey, M. Liu, and M. Karir, "On the mismanagement and maliciousness of networks.," in *NDSS*, vol. 14, 2014, pp. 23–26.

[40] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling dynamic behavior in large evolving graphs," in *Proceedings of the sixth ACM international conference on Web search and data mining*, 2013, pp. 667–676.

[41] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM transactions on networking*, vol. 23, no. 4, pp. 1257–1270, 2014.

[42] S. Banescu, C. Collberg, and A. Pretschner, "Predicting the resilience of obfuscated code against symbolic execution attacks via machine learning," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 661–678.

[43] K. Soska and N. Christin, "Automatically detecting vulnerable websites before they turn malicious," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 625–640.

[44] K. Borgolte, C. Kruegel, and G. Vigna, "Delta: Automatic identification of unknown web-based infection campaigns," in *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, 2013, pp. 109–120.

[45] H. Yang, X. Ma, K. Du, Z. Li, H. Duan, X. Su, G. Liu, Z. Geng, and J. Wu, "How to learn klingon without a dictionary: Detection and measurement of black keywords used by the underground economy," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 751–769.

[46] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, "Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 755–766.

[47] D. Kong, L. Cen, and H. Jin, "Autoreb: Automatically understanding the review-to-behavior fidelity in android applications," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 530–541.

[48] R. P. Khandpur, T. Ji, S. Jan, G. Wang, C.-T. Lu, and N. Ramakrishnan, "Crowdsourcing cybersecurity: Cyber attack detection using social media," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1049–1057.

[49] C. Sabottke, O. Suciu, and T. Dumitraș, "Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 1041–1056.

[50] A. Ritter, E. Wright, W. Casey, and T. Mitchell, "Weakly supervised extraction of computer security events from twitter," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 896–905.

[51] *The Web Application Security Consortium | Web-Hacking-Incident-Database*. [Online]. Available: http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database (visited on 12/20/2021).

[52] *Composite Blocking List*. [Online]. Available: https://www.abuseat.org/ (visited on 12/20/2021).

[53] *Spamhaus*. [Online]. Available: https://www.spamhaus.org/sbl/ (visited on 12/20/2021).

[54] *Spamcop*. [Online]. Available: https://www.spamcop.net/ (visited on 12/20/2021).

[55] *WPBL*. [Online]. Available: http://www.wpbl.info/ (visited on 12/20/2021).

[56] *Phishtank*. [Online]. Available: https://phishtank.org/ (visited on 12/20/2021).

[57] *Titanfile article*. [Online]. Available: https://www.titanfile.com/blog/the-three-largest-data-breaches-of-2014/ (visited on 12/20/2021).

[58] *AWIS*. [Online]. Available: https://docs.aws.amazon.com/awis/ (visited on 12/20/2021).

[59] *PandaLabs*. [Online]. Available: https://www.pandasecurity.com/en/mediacenter/pandalabs/pandalabs-q3/ (visited on 12/20/2021).

[60] *Symantec*. [Online]. Available: https://securitycloud.symantec.com/cc/landing (visited on 12/20/2021).

[61] *Virustotal*. [Online]. Available: https://www.virustotal.com/gui/home/upload (visited on 12/20/2021).

[62] *NIST National Software Reference Library*. [Online]. Available: https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsrl (visited on 12/20/2021).

[63] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," *Encyclopedia of database systems*, vol. 5, pp. 532–538, 2009.

[64] *Kubernetes*. [Online]. Available: https://kubernetes.io/ (visited on 12/20/2021).

[65] *Pachyderm*. [Online]. Available: https://www.pachyderm.com/ (visited on 12/20/2021).

[66] *DVC*. [Online]. Available: https://dvc.org/ (visited on 12/20/2021).

[67] *Dolthub*. [Online]. Available: https://www.dolthub.com/ (visited on 12/20/2021).

[68] *Apache Spark*. [Online]. Available: https://spark.apache.org/ (visited on 12/20/2021).

[69] *Dask*. [Online]. Available: https://dask.org/ (visited on 12/20/2021).

[70] *Ray*. [Online]. Available: https://www.ray.io/ (visited on 12/20/2021).

[71] *Horovod*. [Online]. Available: https://github.com/horovod/horovod (visited on 12/20/2021).

[72] *Raysgd*. [Online]. Available: https://docs.ray.io/en/latest/raysgd/raysgd.html (visited on 12/20/2021).

[73] *Tensorflow distributed*. [Online]. Available: https://www.tensorflow.org/guide/distributed_training (visited on 12/20/2021).

[74] *Pytorch Distributed*. [Online]. Available: https://pytorch.org/docs/stable/distributed.html (visited on 12/20/2021).

[75] *Apahe Airflow*. [Online]. Available: https://airflow.apache.org/ (visited on 12/20/2021).

[76] *Kubeflow*. [Online]. Available: https://www.kubeflow.org/ (visited on 12/20/2021).

[77] *Seldon Core*. [Online]. Available: https://www.seldon.io/tech/products/core/ (visited on 12/20/2021).

[78] *Cortex*. [Online]. Available: https://www.cortex.dev/ (visited on 12/20/2021).

[79] *Pytorch Serve*. [Online]. Available: https://pytorch.org/serve/ (visited on 12/20/2021).

[80] *Tensorflow Extended*. [Online]. Available: https://www.tensorflow.org/tfx (visited on 12/20/2021).

[81] *Google Cloud*. [Online]. Available: https://cloud.google.com/ (visited on 12/20/2021).

[82] *AWS*. [Online]. Available: https://aws.amazon.com/ (visited on 12/20/2021).

[83] *Tensorflow*. [Online]. Available: https://www.tensorflow.org/ (visited on 12/20/2021).

[84] *Jupyter*. [Online]. Available: https://jupyter.org/ (visited on 12/20/2021).

[85] *Mlflow Tracking*. [Online]. Available: https://mlflow.org/docs/latest/tracking.html (visited on 12/20/2021).

[86] *Mlflow Projects*. [Online]. Available: https://mlflow.org/docs/latest/projects.html (visited on 12/20/2021).

[87] *Sagemaker*. [Online]. Available: https://aws.amazon.com/sagemaker/ (visited on 12/20/2021).

[88] *Microsoft Azure*. [Online]. Available: https://azure.microsoft.com/en-us/ (visited on 12/20/2021).

[89] *MLflow Model Registry*. [Online]. Available: https://mlflow.org/docs/latest/model-registry.html (visited on 12/20/2021).

[90] *Welcome to Python.org*. [Online]. Available: https://www.python.org/ (visited on 12/20/2021).

[91] *scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation*. [Online]. Available: https://scikit-learn.org/stable/ (visited on 12/20/2021).

[92] *PyTorch*. [Online]. Available: https://pytorch.org/ (visited on 12/20/2021).

[93] *Pyspark*. [Online]. Available: https://spark.apache.org/docs/latest/api/python/ (visited on 12/20/2021).

[94] *slundberg/shap: A game theoretic approach to explain the output of any machine learning model.* [Online]. Available: https://github.com/slundberg/shap (visited on 12/20/2021).

[95] *EUR-Lex - 52021PC0206 - EN - EUR-Lex*. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1623335154975uri=CELEX (visited on 12/23/2021).

[96] *scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation*. [Online]. Available: https://scikit-learn.org/stable/ (visited on 12/21/2021).

[97] *VCDB/data/csv at master vz-risk/VCDB*. [Online]. Available: https://github.com/vz-risk/VCDB/tree/master/data/csv (visited on 12/28/2021).

[98] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM sigmod record*, vol. 29, no. 2, pp. 1–12, 2000.

[99] *Verisr2*. [Online]. Available: https://github.com/onlyphantom/verisr2 (visited on 12/20/2021).

[100] *scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation*. [Online]. Available: https://scikit-learn.org/stable/ (visited on 12/15/2021).

[101] *Ggplot2*. [Online]. Available: https://ggplot2.tidyverse.org/ (visited on 12/20/2021).

[102] *Advanced Info Service*. [Online]. Available: https://www.ais.th/ (visited on 12/20/2021).

[103] L. Nagel, "The influence of the covid-19 pandemic on the digital transformation of work," *International Journal of Sociology and Social Policy*, 2020.

[104] *DBIR*. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/ (visited on 12/20/2021).

[105] *Verizon*. [Online]. Available: https://www.verizon.com/business/ (visited on 12/20/2021).

[106] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[107] A. D'Amico, K. Whitley, D. Tesone, B. O'Brien, and E. Roth, "Achieving Cyber Defense Situational Awareness: A Cognitive Task Analysis of Information Assurance Analysts:" *http://dx.doi.org/10.1177/154193120504900304*, pp. 229–233, Nov. 2016, ISSN: 10711813. DOI: 10.1177/154193120504900304. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/154193120504900304.

[108] *Actions*. [Online]. Available: http://veriscommunity.net/actions.html (visited on 12/27/2021).

[109] *Assets*. [Online]. Available: http://veriscommunity.net/assets.html#section-variety (visited on 12/27/2021).

[110] *Demographics*. [Online]. Available: http://veriscommunity.net/victim-demo.html (visited on 12/27/2021).

[111] *6.1. Pipelines and composite estimators — scikit-learn 1.0.2 documentation*. [Online]. Available: https://scikit-learn.org/stable/modules/compose.html (visited on 12/28/2021).

[112] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, Aug. 1987, ISSN: 0169-7439. DOI: 10.1016/0169-7439(87)80084-9.

[113] S. Menard, *Applied logistic regression analysis*. Sage, 2002, vol. 106.

[114] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[115] *Welcome to LightGBM's documentation! — LightGBM 3.3.1.99 documentation*. [Online]. Available: https://lightgbm.readthedocs.io/en/latest/ (visited on 12/25/2021).

[116] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," *34th International Conference on Machine Learning, ICML 2017*, vol. 7, pp. 4844–4866, 2017. arXiv: 1704.02685. [Online]. Available: https://github.com/slundberg/shap.

[117] *Cloud Object Storage – Amazon S3 – Amazon Web Services*. [Online]. Available: https://aws.amazon.com/s3/ (visited on 12/25/2021).

[118] *PostgreSQL: The world's most advanced open source database*. [Online]. Available: https://www.postgresql.org/ (visited on 12/25/2021).

[119] *NumPy*. [Online]. Available: https://numpy.org/ (visited on 12/25/2021).

[120] *pandas - Python Data Analysis Library*. [Online]. Available: https://pandas.pydata.org/ (visited on 12/25/2021).

[121] *Welcome to Connexion's documentation! — Connexion 2020.0.dev1 documentation*. [Online]. Available: https://connexion.readthedocs.io/en/latest/ (visited on 12/25/2021).

[122] *REST API Documentation Tool | Swagger UI*. [Online]. Available: https://swagger.io/tools/swagger-ui/ (visited on 12/25/2021).

[123] *verispy PyPI*. [Online]. Available: https://pypi.org/project/verispy/ (visited on 12/25/2021).

[124] *Overview of Docker Compose | Docker Documentation*. [Online]. Available: https://docs.docker.com/compose/ (visited on 12/25/2021).

[125] *Command-Line Interface — MLflow 1.22.0 documentation*. [Online]. Available: https://www.mlflow.org/docs/latest/cli.html (visited on 12/26/2021).

[126] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002, ISSN: 1076-9757. DOI: 10.1613/JAIR.953. arXiv: 1106.1813. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10302.

[127] *Advanced Load Balancer, Web Server, Reverse Proxy - NGINX*. [Online]. Available: https://www.nginx.com/ (visited on 12/20/2021).

Appendices

A Critical Configuration Files

A.1 Deployment Configuration Json File of RiskML

```
{
"algorithms": "LGBM KNN RF",
"ports_models": {
   "5010" : "asset.variety.Server",
   "5011" : "asset.variety.User Dev",
   "5012" : "asset.variety.Network",
   "5013" : "asset.assets.variety.S.Database",
   "5014" : "asset.assets.variety.S.Web application",
   "5015" : "asset.assets.variety.U.Desktop or laptop",
   "5020" : "action.Malware",
   "5021" : "action.Hacking",
   "5022" : "action.malware.variety.Ransomware",
   "5023" : "action.hacking.variety.DoS",
   "5024" : "action.hacking.variety.SQLi",
   "5025" : "action.hacking.variety.Brute force",
   "5026" : "action.hacking.variety.Use of backdoor or C2",
   "5030" : "attribute.Confidentiality",
   "5031" : "attribute.Integrity",
   "5032" : "attribute.Availability"
   }
}
```

A.2 Docker-Compose File for the Deployment of RiskML

```
version: '3.7'

services:

    minio:
        image: minio/minio:latest
        container_name: myminio
        volumes:
            - artifact-store:/mlflow-bucket
        ports:
            - "9000:9000"
        environment:
            MINIO_ROOT_USER: minio-id
            MINIO_ROOT_PASSWORD: minio-key
            AWS_ACCESS_KEY_ID: 'minio-id'
            AWS_SECRET_ACCESS_KEY: 'minio-key'
        networks:
            - backend
```

```
        command: server /mlflow-bucket
        healthcheck:
            test: ["CMD", "curl", "-f",
                "http://localhost:9000/minio/health/live"]
            interval: 10s
            timeout: 10s
            start_period: 3s
            retries: 3

    create_bucket:
        image: minio/mc:latest
        depends_on:
            - minio
        networks:
            - backend
        environment:
            MINIO_ROOT_USER: minio-id
            MINIO_ROOT_PASSWORD: minio-key
            PORT: 9000
        entrypoint: >
            /bin/sh -c "
            /usr/bin/mc config host rm local;
            /usr/bin/mc config host add --quiet --api s3v4 \
                local http://myminio:9000 minio-id minio-key;
            /usr/bin/mc rb --force local/mlflow-bucket/;
            /usr/bin/mc mb --quiet local/mlflow-bucket/;
            /usr/bin/mc policy set public local/mlflow-bucket;
            "
    pgdb:
        container_name: pgdb
        restart: always
        build: ./docker-db
        image: pgdb
        volumes:
            - database:/var/lib/postgresql/data
        ports:
            - 5432:5432
        expose:
            - 5432
        environment:
            POSTGRES_USER: ${POSTGRES_USER}
            POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
            POSTGRES_DATABASE: ${POSTGRES_DATABASE}
            PGDATA: /var/lib/postgresql/data/pgdata
        networks:
            - backend

    mlflow_server:
        restart: always
```

```
        build: ./docker-mlflow-server
        image: mlflow_server
        container_name: mlflow_server
        ports:
            - 5000:5000
        expose:
            - 5000
        environment:
            POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
            POSTGRES_USER: ${POSTGRES_USER}
            POSTGRES_DATABASE: ${POSTGRES_DATABASE}
            MLFLOW_S3_ENDPOINT_URL: http://minio:9000
            AWS_ACCESS_KEY_ID: minio-id
            AWS_SECRET_ACCESS_KEY: minio-key
        depends_on:
            - minio
            - pgdb
        networks:
            - frontend
            - backend
        command: bash -c
            "mlflow server --backend-store-uri \
             postgresql+psycopg2://${POSTGRES_USER}: \
               ${POSTGRES_PASSWORD}@pgdb:54/${POSTGRES_DATABASE} \
             --host 0.0.0.0 \
             --default-artifact-root s3://mlflow-bucket/
            "

    mlflow_core:
        restart: always
        build: .
        image: mlflow_code
        container_name: mlflow_core
        ports:
            - 5010:5010
            - 5011:5011
            - 5012:5012
            - 5013:5013
            - 5014:5014
            - 5015:5015
            - 5020:5020
            - 5021:5021
            - 5022:5022
            - 5023:5023
            - 5024:5024
            - 5025:5025
            - 5026:5026
            - 5030:5030
            - 5031:5031
```

```
            - 5032:5032
            - 5100:5100
        environment:
            MLFLOW_TRACKING_URI: http://mlflow_server:5000
            MLFLOW_S3_ENDPOINT_URL: http://minio:9000
            AWS_ACCESS_KEY_ID: 'minio-id'
            AWS_SECRET_ACCESS_KEY: 'minio-key'
        depends_on:
            - minio
            - create_bucket
            - pgdb
            - mlflow_server
        working_dir: '/code/'
        networks:
            - backend
            - frontend
        command: bash -c
            "sleep 5 && (python3 api/server.py & \
             python3 shell_scripts/shell_script_composer.py && \
             ./shell_scripts/init_models.sh )
            "

networks:
    frontend:
        driver: bridge
    backend:
        driver: bridge
volumes:
    database:
    artifact-store:
```

A.3 MLproject for the Deployment of RiskML

```
name: thesis

conda_env: conda.yml

entry_points:

  gridsearch:
    parameters:
      task: {type: string, default: 'asset.variety'}
      metric: {type: string, default: 'f1'}
      averaging: {type: string, default: 'macro'}
      imputer: {type: string, default: 'dropnan'}
      random_state: {type: int, default: 0}
      merge: {type: str, default: 'yes'}
      pca: {type: int, default: 0}
      n_jobs_cv: {type: int, default: 1}
      n_folds: {type: int, default: 5}
```

```
    command: |
      python3 hyperparameter_tuning/gridsearch.py --pca {pca} \
        --merge {merge} --task {task} --imputer {imputer} \
        --n-folds {n_folds} --n-jobs-cv {n_jobs_cv} \
        --metric {metric} --averaging {averaging} \
        --random-state {random_state}

  train:
    parameters:
      task: {type: string, default: 'asset.variety'}
      target: {type: string, default: 'Server'}
      algo: {type: string, default: 'LGBM'}
      hyperparams: {type: string, default: '{}'}
      imputer: {type: string, default: 'dropnan'}
      train_size: {type: float, default: 0.8}
      split_random_state: {type: int, default: 0}
      merge: {type: str, default: "yes"}
      pca: {type: int, default: 0}
      explain: {type: str, default: "no"}
      shap_data_percentage: {type: float, default: 0.1}
      shap_test_over_train_percentage: {type: float, default: 0.3}
    command: |
      python3 train.py --pca {pca} --merge {merge}
        --explain {explain} --task {task} \
        --target {target} --algo {algo} \
        --hyperparams {hyperparams} --imputer {imputer} \
        --train-size {train_size} \
        --split-random-state {split_random_state} \
        --n-folds {n_folds} \
        --shap-data-percentage {shap_data_percentage} \
        --shap-test-over-train-percentage \
          {shap_test_over_train_percentage}
```

B RiskML Interfaces Specification (APIs and UIs)

B.1 View Models

Endpoint Name	View Models
Description	Get the list of models that are being served right now (functions.getModels). This API function employs shell scripts (created at the moment of initial deployment) that scan all ports configured in the deployment configuration file (config.json).
HTTP Method	GET
Endpoint URL	http://<mlflow_core>:5100/api/functions/getModels
Parameters	-
Request Body Example	-

B.2 Retrain Models

Endpoint Name	Retrain Models
Description	Retrain the selected model (functions.retrainModels). This API function accepts the model_id (or "all") and performs retraining of the model according to the predefined ML algorithms (at the moment of initial deployment) in the configuration file (config.json), compares with the existing one that is in production and redeploys the best one. At the back-end, this function employs a shell script that handles the MLflow CLI.
HTTP Method	PUT
Endpoint URL	http://mlflow_core>:5100/api/functions/retrainModels
Parameters	model_id
Request Body Example	-

B.3 Update Database

Endpoint Name	Update Database
Description	Update to latest VCDB version from github (see functions.updateVCDB() of *api/functions.py* file of the RiskML repository [19]). This API function updates the stored version of the VCDB dataset in order to capture any updates relating to the cyber-security incidents status quo. In the back-end, this API function triggers a shell script that clones the VCDB repository (https://github.com/vz-risk/VCDB.git) using git.
HTTP Method	GET
Endpoint URL	http://<mlflow_core>:5100/api/functions/updateVCDB
Parameters	-
Request Body Example	-

B.4 Experiment Tracking UI

Endpoint Name	Experiment Tracking UI
Description	Track ML experiments. This is the main MLflow Tracking UI. The administrative end-user can monitor performance of all trained models and experiments in real time.
HTTP Method	GET
Endpoint URL	http://<mlflow_tracking_server>:5000
Parameters	-
Request Body Example	-

B.5 Model Registry

Endpoint Name	Model Registry
Description	Track registered models. This is the main MLflow Model Registry. The administrative end-user can monitor registered models along with their general status, versions and stages.
HTTP Method	GET
Endpoint URL	http://<mlflow_tracking_server>:5000/#/models
Parameters	-
Request Body Example	-

B.6 MinIO Server Administration UI

Endpoint Name	MinIO Server Administration UI
Description	View saved models. This the endpoint of save S3 compatible model storage. Credentials are necessary to enter.(*.env* file)
HTTP Method	GET
Endpoint URL	http://<minio>:9000/
Parameters	-
Reques Body Example	-

B.7 Inference Endpoints - asset.variety.X

Endpoint Name	Inference Server - asset.variety.X
Description	Get predictions for models governed by the type asset.variety.X, where X can be Server, User Dev, Network etc.. Normally, ports for such models will vary in the range 5010:5019 depending on the initial configuration.
HTTP Method	POST
Endpoint URL	http://<mlflow_core:<model_port>/invocations
Parameters	{"model_port" : "int"}
Request Body Example	See input_examples/ folder of RiskML repository [19]

B.8 Inference Endpoints - asset.assets.variety.X - Y

Endpoint Name	Inference Server - asset.variety.X - Y
Description	Get predictions for models governed by the type asset.variety.X - Y, where X can be S, U etc. and Y is a corresponding asset of a lower level. (for example if X=S then Y could be Database, Web application etc.). Normally, ports for such models will vary in the range 5010:5019 depending on the initial configuration.
HTTP Method	POST
Endpoint URL	http://<mlflow_core:<model_port>/invocations
Parameters	{"model_port" : "int"}
Request Body Example	See input_examples/ folder of RiskML repository [19]

B.9 Inference Endpoints - action.X

Endpoint Name	Inference Server - action.X
Description	Get predictions for models governed by the type action.X, where X can be Hacking, Malware, Social etc.). Normally, ports for such models will vary in the range 5020:5029 depending on the initial configuration.
HTTP Method	POST
Endpoint URL	http://<mlflow_core:<model_port>/invocations
Parameters	{"model_port" : "int"}
Request Body Example	See input_examples/ folder of RiskML repository [19]

B.10 Inference Endpoints - action.X.variety.Y

Endpoint Name	Inference Server - action.X.variety.Y
Description	Get predictions for models governed by the type action.X.variety.Y , where X can be Hacking, Malware, Social etc. and Y the lower level action (for example if X=hacking then Y could be SQLi, Denial of service etc.). Normally, ports for such models will vary in the range 5020:5029 depending on the initial configuration.
HTTP Method	POST
Endpoint URL	http://<mlflow_core:<model_port>/invocations
Parameters	{"model_port" : "int"}
Request Body Example	See input_examples/ folder of RiskML repository [19]

B.11 Inference Endpoints - attribute.X

Endpoint Name	Inference Server - attribute.X
Description	Get predictions for models governed by the type attribute.X, where X can be Confidentiality, Intergrity or Avalability). Normally, ports for such models will vary in the range 5030:5039 depending on the initial configuration.
HTTP Method	POST
Endpoint URL	http://<mlflow_core:<model_port>/invocations
Parameters	{"model_port" : "int"}
Request Body Example	See input_examples/ folder of RiskML repository [19]