Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

DDoS események vizsgálata és validálása mélytanulási modellekkel

TDK dolgozat

Készítette:
Kiss Péter

Konzulens:
Dr. Orosz Péter

2023

Tartalomjegyzék

Kivonat

A mai kibervédelmi incidensek jelentős részét DDoS támadások teszik ki. Ezeknek a támadásoknak legfőbb jellemzője, hogy a támadók célja az adott szerver vagy szolgáltatás elérhetetlenné tétele, megbénítása. A támadások többségéért általában nem csupán egy-egy IP-cím, hanem sok, úgynevezett botnet hálózat a felelős, a támadók ezzel a magatartással soksor súlyos anyagi kárt okoznak a felhasználóknak és az üzemeltetőknek egyaránt. Fontos, hogy nem csupán a nagy volumenű hálózati forgalommal lehetséges túlterheléses támadást véghezvinni, gyakran megfigyelhetőek olyan támadások is, melyek valamely ismert hálózati protokoll hibáit, gyengeségeit próbálják meg kihasználni. Néhány, jól ismert támadás ellen hagyományos módszerekkel is lehetséges hatékony védelmet képezni, azonban idővel a támadók eszközei is fejlődnek. Egyre ügyesebben imitálják a normál felhasználók hálózati forgalmi karakterisztikáit, illetve gyakoriak a rövid felfutású, tranziens-szerű események is. Ezeket a megszokott, hagyományos hálózatbiztonsági megoldásokkal szinte lehetetlen detektálni. Emellett a probléma kiemelt nehézségűnek tekinthető azért is, mert hatalmas volumenű hálózati adatforgalomban kell felismernünk azokat, akik veszélyt jelenthetnek, valamint azért, mert a modelltől elvárt pontosság igen nagy, kritikus fontosságú, hogy a fals pozitív és fals negatív detekciós rátát a lehető legkisebbre csökkentsük.

A munkám során azt vizsgáltam meg, hogy felügyelt és felügyelet nélküli mesterséges intelligencia modellek segítségével milyen hatékonysággal tudjuk ezeket az eseményeket feldolgozni, és helyesen detektálni. A vizsgálatok során egy ipari partner által rögzített, valós, aktuális és címkézett hálózati forgalmi adatokból dolgoztam. Természetesen nagyon fontos az, hogy minden mélytanulási feladat során megismerkedjünk a kapott adatbázissal, így a dolgozatomban egy feltáró adatelemzési szakaszban bemutatom, hogy hogyan is néz ki egy ilyen adatsor, hogyan oszlanak el a legfontosabb mért metrikák és a különböző hálózati forgalom típusok. Ezek után térek rá arra, hogy hogyan lehetne ezeket valamilyen alkalmazott mélytanulási eszközzel feldolgozni. A legtöbbet a Random Forest becslővel és a K-means klaszterezéssel dolgoztam, de más algoritmusokat is segítségül hívtam. Ezen algoritmusok rövid elméleti bemutatása után kifejtem és összehasonlítom, hogy milyen eredményeket sikerült ezek alkalmazásával és optimalizálásával elérnem. Arra is megpróbálok magyarázatot adni, hogy miért tapasztalhatunk ilyen teljesítménybeli különbséget a különböző modellek között. Következtetésként pedig bemutatom azt, hogy milyen módszerekkel és megoldásokkal lehetne egy ilyen rendszer pontosságán tovább javítani, valamint, hogy milyen egyéb kihívásokat rejt még ez a terület.

Abstract

In today's cyber-security world, many incidents are made by DDoS attackers. The most important feature of these attacks is that the goal of the attackers is to make a service or an entire server unavailable for all users. Most of the time, the source of these attacks cannot be entitled to only one IP address, but many different IPs, so called a botnet network. With these methods, attackers are able to cause serious financial damage for operators and normal users too. It is important to note, that such DDoS attacks can not only be made using huge amount of network data traffic. They usually try to take advantage of implementation errors in widely used network protocols of the TCP-IP protocol stack. Some of the well-known DDoS attacks can be mitigated using traditional network security devices, but as time goes by, attackers develop more and more advanced technics. They try to mimic normal user traffic characteristics, and we can often observe fast, short timespan transient-like events too. Such events cannot be detected using usual network security methods. Moreover, the problem is difficult, since we have to detect small malicious events in huge amount of normal user network traffic, and the expected reliability of our solution is very strict. It is critical, to have the smallest possible false negative and false positive detection rates. In my thesis, I examined how accurately can we detect DDoS attacks using unsupervised and supervised artificial intelligence algorithms.

During my work, I used a dataset captured by industrial partners. This network data was recorded on real life working servers and services, and it contains output labels, weather it was normal traffic or a DDoS event. Naturally, as in every deep learning project, it is important to analyze the database beforehand, that is why I dedicated a paragraph in my thesis to an exploratory data analysis section. I will show the reader, how a network traffic dataset like this is built up, which are the most important metrics, and how are they distributed in the captured data. After all this, I explain how we can use all this to train ai models using supervised and unsupervised methods. In most of my work, I used Random Forest and K-means clustering models, but I also mention neural models and other clustering methods. After explaining the theoretical background of these algorithms, I will show how effective are they for detecting DDoS attacks. Some of them performed way better than others. I try to find explanation of why there is such a difference between the different solutions. As a conclusion, I show the reader how could these methods be improved for being even more reliable, and what are the different further challenges in the field of network security and DDoS attacks.

1. Bevezetés, a terület bemutatása

1.1. A DDoS támadások világa

A hálózati biztonság területén egyre gyakrabban találkozhatunk az úgynevezett DDoS (Disributed Denial of Service) támadásokkal, melyekre a magyar szaknyelvben túlterheléses támadásokként utalunk. Érzékelhető tendencia, hogy ezek az események évről évre egyre gyakrabban történnek meg, és egyre súlyosabb károkat képesek okozni. Ezek a kibervédelmi incidensek sok különböző megjelenési formában ismertek, azonban a támadók célja mindig ugyanaz: egy hálózat vagy szerver szolgáltatásait a felhasználók számára elérhetetlenné tenni úgy, hogy olyan hatalmas mennyiségű adattal árasztjuk azt el, amelyet már képtelen lassulás vagy teljes leállás nélkül kezelni. Ezzel a magatartással a támadók igen súlyos kárt tudnak okozni az üzemeltetőknek és a felhasználóknak egyaránt. Számos formában ismertek ezek a támadások, gyakran a TCP/IP protocol stack különböző rétegeiben jelennek meg. Legtöbbször a forgalom nagy volumenén felül valami egyéb hálózattudományi ismeretet is felhasználnak a támadók, például sok olyan támadás ismert, ahol valamilyen elterjedt hálózati protokoll megvalósításának gyengeségeit is igyekeznek kihasználni. Ennek a munkának nem célja az, hogy ezeket a támadásokat részletesen bemutassa, azonban néhány példával könnyen megmutatható, hogy mit értek ezalatt. Igen elterjedt példa lehet az úgynevezett SNY flood támadás, ahol TCP SYN üzenetekkel bombázzák a szervert, míg annak szabad portjai egy idő után elfogynak. A támadás hatékonyságának oka a three-way handshake mechanizmus implementációjának hibájában keresendő. Vagy egy másik, szintén elterjedt támadás során úgy járnak el, hogy HTTP GET csomagokkal bombáznak egy szervert, melynek során azt használják ki, hogy a HTTP protokoll természeténél fogva állapotmentes alkalmazásrétegbeli megoldás.

A klasszikus, egy-egy IP-ről érkező DoS támadások manapság már nem túlzottan jellemzőek. Ennek oka, hogy nehéz egyetlen IP-címről igazán nagy mennyiségű adatforgalmat generálni, valamint, hogy az utóbbi években számos hagyományos kibervédelmi megoldás kidolgozásra került ezeknek a támadásoknak a megállítására. A támadásoknak egy manapság már sokkal elterjedtebb válfaja az elosztott szolgáltatás megtagadáson alapuló támadás, vagyis DDoS (Distributed Denial of Service) támadás. Ennek során a támadók úgy járnak el, hogy egy automatizált alkalmazás segítségével felkutatják az internetre kapcsolódó sebezhető, könnyen feltörhető számítógépeket és okoseszközöket, amikre aztán gyors feltörésük után feltelepítenek egy támadó programot, mindezt annak felhasználójának tudta nélkül. Ezzel a módszerrel egy úgynevezett botnet hálózatot hoznak létre, melyben a "zombi" eszközök később távoli paranccsal utasíthatók arra, hogy megtámadjanak egy, a támadók által megjelölt szolgáltatást vagy szervert úgy, hogy valamilyen formában folyamatos adatforgalmat bonyolítanak vele. Egy-egy ilyen megfertőzött eszköz forgalma önmagában természetesen nem feltétlenül jelentene túlzottan nagy adatmennyiséget, azonban nem ritka, hogy egy botnet akár több tízezer megfertőzött gépből áll, amelyek erejüket egyesítve már hatalmas károkat is képesek okozni. Ezek ellen a támadások ellen rendkívül nehéz védekezni, ugyanis a botnetek forgalmát általában nehéz a jóindulatú felhasználók

forgalmától megkülönböztetni, azért is, mert a támadók előszeretettel igyekeznek olyan módszereket használni, mely a normál forgalmat jól imitálja. Emellett az is elmondható, hogy a támadás kezdeményezőjének kilétét (az úgynevezett mastert) sokszor szinte lehetetlen felfedni. Ezek a típusú támadások azért is kezdenek egyre jobban elterjedni, mert évről évre egyre több a hálózatra kapcsolt okoseszköz (IP kamera, okos világítás illetve egyéb IoT- és mobileszközök), melyek tervezésekor a gyártók célja, hogy minél egyszerűbbek és olcsóbbak legyenek, a lehető legkisebb energiafogyasztással. Ennek hátulütőjeként viszont általában ezeknek az eszközöknek a beépített biztonsági megoldásaik is nagyon egyszerűek, vagyis a hackereknek szinte gyerekjáték feltörni és botnetbe szervezni őket. Ezt kihasználva, az interneten elterjedőben vannak olyan szolgáltatások, melyek igénybevételével gyakorlatilag DDoS támadást rendelhetünk meg. Így szinte bárki indíthat támadást egy általa kiszemelt szolgáltatás ellen, akár komolyabb szakértelem nélkül is. Mindezen tényezők következtében az utóbbi évek kibertámadási eseményeinek jelentős hányadát DDoS támadások tették ki, amik egyre gyakrabban egyre komolyabb károkat képesek okozni, és amik ellen valamilyen formában védekeznünk kell, azonban a hagyományos védelmi módszerek ebben a feladatban általában nem teljesítenek túlzottan jól. Az is érezhető tendencia, hogy a védelmi mechanizmusok fejlődésével a támadók módszerei is egyre fejlettebbé, összetettebbé válnak. Évről évre újabb és újjabb támadástípusok jelennek meg, valamint gyakran találkozhatunk a korábbról ismert támadások kombinációival is, melyekre a szakterület a támadás vektor kifejezéssel utal.

1.2. A feladat célja

Az adatközponti hálózatokban azonosított biztonsági események számossága manapság olyan hatalmas mértékű, melyet emberi erővel lehetetlen volna felügyelni. Számos ismert támadás ellen hatékony védelmi mechanizmusokat implementálhatunk a klasszikus hálózatbiztonsági eszközeinkbe, azonban ezek felhasználhatósága korlátozott, hiszen csak ismert támadástípusok felismerésére képesek. Éppen ezért valamilyen mesterséges intelligencia alapú megoldással kívánjuk támogatni az elterjedt hálózatbiztonsági megoldások tárházát. Kritikus elvárás azonban az, hogy egy ilyen rendszer a jóindulatú felhasználók kommunikációját semmiképpen se korlátozza, a támadókét azonban lehetőleg minél nagyobb magabiztossággal legyen képes felismerni.

A feladat célja tehát kettősnek mondható. Egyrészről meg kell vizsgálnunk és össze kell hasonlítanunk, hogy néhány elterjedt mélytanuló algoritmus segítségével milyen teljesítménnyel vagyunk képesek a különböző események osztályozására, a támadások felismerésére. Ezen felül pedig meg kell néznünk, hogy milyen mértékben vagyunk képesek megközelíteni az ipari rendszerek pontossági követelményeit a felhasznált modellek optimalizálásával, további bemenetek képzésével. A munka során felügyelt és felügyelet nélküli tanító algoritmusokat is alkalmaztam, ezek közül legtöbbet a Random Forest becslővel, valamint a K-means osztályozással dolgoztam, ezeket szeretném ebben a beszámolóban bemutatni, valamint röviden kitérek a Multi Layer Perceptron és a DBSCAN alkalmazási lehetőségeire is.

1.3. A felhasznált eszközök

A munkám során használt adatbázist az Aitia International Zrt. bocsátotta rendelkezésemre. Ennek az adatsornak fontos tulajdonsága, hogy valós forgalmi mérési eredményekből jött létre, a cég ügyfelei által üzemeltetett, valós ipari környezetben megfigyelt hálózati események rekordjai találhatók meg benne, anonimizált, vissza nem követhető formátumban. Az adatbázis tartalmazza a hálózati forgalom legfontosabb adatait, cél IP-címeket és portokat, az események kezdetének és végének időpontjait, az adatsebességet és még sok más mellett azt, hogy az adott esemény DDoS támadás, gyanús forgalom, vagy csupán normál forgalom volt e. Azonban fontos megjegyezni, hogy ez az adatbázis a detektorok által mért adatoknak már egy redukált formája, ennek következtében a modell által elvárható pontosság is más kontextusba kerül, hiszen a döntéshozatal során nem vagyunk minden információ birtokában. Az adatbázis részletesebb bemutatását a negyedik fejezetben teszem meg. A feladat kidolgozása során a Python 3.11.3-as verzióját használtam, ezen belül is a legtöbbet a Pandas és a Scikit-Learn függvénykönyvtárakat, valamint a Tensor Flow alapú Keras API-t.

1.4. A felhasznált módszertan

A dolgozatban bemutatott munka módszertana modellezésen, modell optimalizáción, illetve klasszikus összehasonlító teljesítményelemzésen alapul. Több, különböző megközelítés és algoritmus detekciós hatékonyságának eredményeit mutatom be, melyeket egy valós, ipari környezetben rögzített, Big Data jellegű adatbázison sikerült elérni. A tanítást és a tesztelést egységesen, 80-20%-os arányban felosztott tanító- és tesztadatbázisokon végeztem. Természetesen a különböző modellek különböző paramétereinek vizsgálata is részét képezte a munkának, emellet a modellek optimalizálása is fontos lépés volt. Azonban ebben a dolgozatban sokkal inkább az eredmények bemutatására, összehasonlítására, és megértésére összpontosítok.

1.5. A dolgozat további felépítése

A dolgozat felépítése a következő struktúrát követi: a második fejezetben bemutatom, hogy milyen korábbi tapasztalatokkal rendelkezik ez a terület, kik, hogyan és milyen eredményekkel vizsgálták azt, hogy milyen hatékonysággal lennénk képesek DDoS támadások felismerésére mesterséges intelligencia modellekkel. Ezek az eredmények az én munkám közvetlen előzményeinek is tekinthetők. Ezek után, a harmadik fejezet részeként bemutatom azt, hogy milyen elméleti alapokat használtam fel a dolgozat készítése során. Mindezt természetesen egy masszív irodalomkutatási munkafázis előzte meg. Persze minden egyes algoritmusról és metódusról akár több különálló dolgozatot is lehetne készíteni, amit nyilvánvaló okokból nem teszek meg, hiszen ennek a munkának nem ez a legfőbb célja. Itt sokkal inkább a leglényegesebb pontokat kiemelve igyekszem minden szükséges és a munka során felhasznált ismeretanyagot átadni, megmagyarázni. Természetesen minden mélytanulási feladat elvégzése előtt elengedhetetlen, hogy megismerkedjünk a számunkra rendelkezésre álló adatsorral, így a negyedik fejezetben bemutatom,

hogy hogyan épül fel egy ilyen forgalmi mintázatokat tartalmazó adatbázis, és hogy klasszikus feltáró adatelemzési módszerekkel milyen összefüggéseket sikerült felismerni. Ezek után, a dolgozat ötödik szekciójában pedig részletesen kifejtem, hogy milyen detekciós eredményeket sikerült elérni az egyes felügyelt és felügyelet nélküli mesterséges intelligencia modellekkel, hogyan hangoltam a különböző paramétereket és számoltam a további bemeneti metrikákat a pontosság további növelésének érdekében. Az utolsó, hatodik fejezetben pedig egy összegző értékelés formájában minősítem az elért eredményeket, valamint kitekintésként néhány további vizsgálati lehetőséget is bemutatok, amelyekkel érdemes lehet ezt a feladatkört további munkákkal kiegészíteni.

2. Irodalomkutatás

A dolgozat témáját körüljáró elmélet, valmint a saját eredmények bemutatása előtt rendkívül fontos, hogy körüljárjuk azt, hogy milyen korábbi eredményekkel rendelkezik ez a terület az akadémiai színtéren, mik azok a munkák, melyek ennek a dolgozatnak az alapjait, előzményeit biztosítják. Ez amiatt is fontos, hiszen ezekből a publikációkból megismerhetjük a technológia jelenlegi állását, gyakran hasznos tanulságokat szűrhetünk le, valamint sok új nézőpontot is megismerhetünk, amelyek a saját munkánk során is rendkívül hasznosnak bizonyulhatnak. Kritikus azonban, hogy figyeljünk ezeknek a kutatásoknak az aktualitására. A technológia rohamos fejlődésével természetesen a hálózattudomány is évről évre fejlődik és változik, emiatt gyakran már akár egy néhány éves kutatási anyag is elavultnak bizonyulhat ezen a területen. Éppen ezért ügyeltem arra, hogy ebben az összefoglalóban csak olyan anyagokat említsek meg, melyek öt évnél nem régebbiek. Amennyiben rákeresünk valamely, a dolgozatban gyakran említett címszóra, akkor számos tudományos munkát találhatunk, mind a mesterséges intelligencia, mind a DDoS támadások mitigációjának területén.

Számos olyan vizsgálattal találkozhatunk, ahol jól ismert, tipikus támadás típusokat próbálnak meg detektálni. Ilyen jól ismert támadás például a HTTP GET flood is. Kimondható, hogy számos ismert támadástípus igen hatékonyan kivédhető hagyományos algoritmusok alkalmazásával is. Ajay Shastry és társainak cikkében például egy rate-limiting megoldást alkalmaztak, mely egyébként az iparban is elterjedt megoldást jelent [1]. Természetesen azonban a hagyományos módszerekkel hamar korlátokba ütközhetünk, hiszen ismeretlen támadások felismerése lehetetlen ilyen módon. Egy 2019-es publikációban már vizsgálták a mesterséges intelligencia felhasználási lehetőségeit, akkor még inkább egyfajta elméleti szintű lehetőségként [2]. Érdekes belegondolni abba, hogy a DDoS támadások világa már abban az időben is igen erősen jelen volt, azonban ez a jelenség évről évre egyre komolyabb fejtörést okoz a szakembereknek, hiszen az az általános tendencia, hogy egyre gyakrabban és egyre súlyosabb DDoS incidenseket tapasztalhatunk. A következő időszakban, különösen a mesterséges intelligencia, mint újfajta számítási paradigma intenzív térnyerésének következtében, egyre több olyan vizsgálati anyag jelent meg, ahol ezt már nem csupán elméleti szinten, hanem a gyakorlatban is megpróbálták alkalmazni, különösen az SDN (Software Defined Networking) környezetekben. Fontos azonban látni azt, hogy a DDoS támadások gyakorlatilag minden elterjedt hálózati megoldásra komoly fenyegetést jelentenek. A. Aljuani cikkében [3] ennek egy igen részletes összefoglalóját találhatjuk meg. Különös veszélyben vannak az IoT (Internet of Things) eszközök hálózatai, hiszen ezeknél az eszközöknél általában a legfontosabb szempont az ár csökkentése és a lehető legegyszerűbb megvalósítás. De emellett a hagyományos hálózati környezetek, valamint a felhő alapú és a szolgáltatók NFV (Network Function Virtualization) infrastruktúrái is fenyegetve vannak.

A mitigációs technikákban számos különböző megközelítésre láthatunk példákat, azonban gyakran itt is csupán ismert, tipikus támadásformákkal találkozhatunk. Érdekes kérdéskör az, hogy milyen eredményeket láthatunk, ha a hagyományos módon működő algoritmusokat összevetjük a mesterséges modellekel. Mudar Sarem

és társainak cikkében a KNN (K-Nearest Neighbors) nevű megoldást alkalmazták, mellyel bőven 90% feletti pontosságot értek el, ami egyébként egy gépi tanulási feladatban jó eredménynek számít [4].

A DDoS támadások a hálózati protocol stack szinte minden rétegében megjelenhetnek, azonban az egyik legelterjedtebb vállfajuk az alkalmazásrétegbeli támadások csoportja. Ezek ellen számos hagyományos algoritmust használó metódus létezik (ilyenek például a Captcha-k), azonban jelenleg egységesen elterjed forma nincsen. H. Beitollah és társainak kutatásában [5] ezeknek a detekciós lehetőségeit vizsgálták RBF (Radial Base Function) hálók segítségével. Ebben az anyagban egy részletes összehasonlítást is találhatunk arra vonatkozóan, hogy más elterjedt felügyelt módszerekhez képest (MLP, RNN, SVM) milyen előnyökkel rendelkezik ez a megoldás, valamint megismerkedhetünk egy érdekes újszerű tanítási metódussal is.

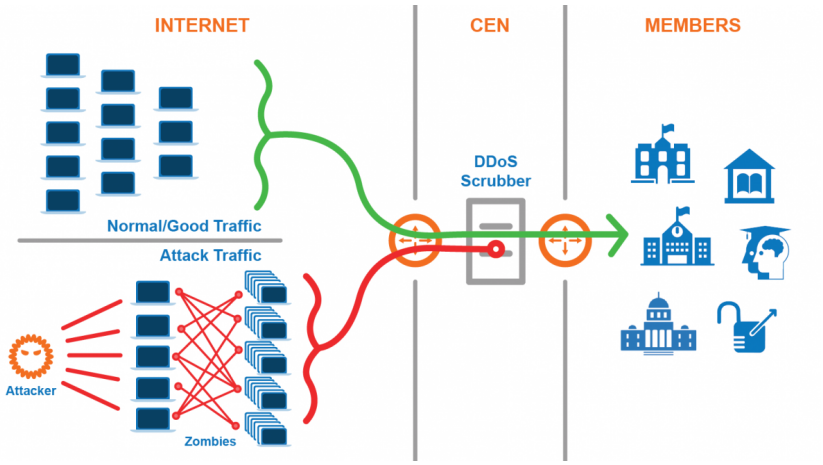
A legtöbb esetben a védekezési mechanizmusok elhelyezése a védendő objektum vagy szolgáltatás mellett történik, a bejövő csomagforgalom viszgálatával. Érdekes megközelítés az a munka, ahol K. Kyungbaek és társai azt vizsgálták, hogy hogyan volna lehetséges ezeket a támadásokat már a forrás oldalon is felismerni, és megállítani [6]. Ennek számos előnye volna, hiszen a támadások többletforgalma nem terhelné feleslegesen a szolgáltatók rendszereit. Ebben a kutatásban egy LSTM cella alapú hálót használtak egy adaptív rate-limiting megoldással kombinálva, melyet az elképzelések szerint a hálózatok fő routerében helyeznének el, és mely értelemszerűen a kimenő forgalmakat szűrné. Egy ilyen megoldással szemben azonban kritikus elvárás a fals-pozitív ráta lehető legkisebb méretűre történő leszorítása, melyet ebben az esetben 4-5% környékén sikerült minimalizálni.

Látható tehát, hogy ennek a területnek számos előzménye van, a kutatási világban rengetegen foglalkoznak azzal a kérdéskörrel, hogy hogyan lehetne a mesterséges intelligenciát kibervédelmi feladatok ellátására alkalmazni. A legfontosabb dilemma azonban mindig az, hogy egy-egy ilyen laborkörülmények között vizsgált, általában mesterségesen generált forgalmi adatok alapján tanított megoldás a valós ipari környezetben, valós adatközponti forgalmak analizálásakor ténylegesen mennyire használható. Kritikus elvárás egy ilyen megoldástól az, hogy a falsrátát a lehető legkisebb mértékűre lecsökkentsük, mely ipari környezetekben általában a néhány százalékos hibaarányt sem engedi meg. Természetesen ez különböző benchmark adatbázisokkal vagy mesterségesen generált adatokkal elérhető laboratóriumi körülmények között, viszont nehezen tudnánk garanciát vállalni arra, hogy egy ilyen megoldás egy ipari környezetben is ennyire tökéletesen működik. Minden a tanító adatok minőségétől függ. Az én munkámban bemutatom, hogy az egyes mélytanuló algoritmusok felhasználásával milyen mértékben vagyunk képesek ezeket a pontossági elvárásokat megközelíteni valós, ipari partner által biztosított adatok felhasználásával.

3. Elméleti alapok

3.1. DDoS támadások kivédése hagyományos kibervédelmi módszerekkel

Klasszikus algortimusokat alkalmazó DDoS védelmi megoldásokat sok elterjedt hagyományos hálózatbiztonsági megoldásban is láthatunk. Minden (nagy)vállalati hálózatban elengedhetetlen, hogy tűzfallal és valamilyen IDS/IPS rendszerrel (Intrusion Detection / Prevention System) rendelkezzen. Bár ezek fő feladata nem ez, de DDoS detekciós szempontból is rendkívül fontos védvonalat képeznek. Ezeken felül az utóbbi években elkezdtek terjedni az úgynevezett scrubber megoldások is, melynek sematikus működését az 1. ábrán figyelhetjük meg. Ezek az eszközök jellemzője, hogy szintén nem csupán DDoS védelmi feladatot láthatnak el, sok egyéb felhasználási területen is megjelennek, nagyon hasznosak lehetnek például a hálózati terheléselosztásban is. A technikának sokféle implementációja is létezik, a lényeg azonban mindenhol az, hogy valamilyen sorosan beépített monitorozó eszköz folyamatosan figyeli a szerver és a kliensek között történő adatforgalmat, és amennyiben ez a forgalom valamilyen előre felállított szabálynak nem tesz eleget, a scrubber átirányítja ezt a forgalmat vagy DDoS védelem esetében gyakorlatilag eldobja (ez nyilvánvalóan implementáció függő) [7][8]. A legfőbb cél, hogy ezek a csomagok semmiképpen se érjék el a védendő objektumot, szolgáltatást.



1. ábra. A scrubber működésének sematikus ábrája[1]

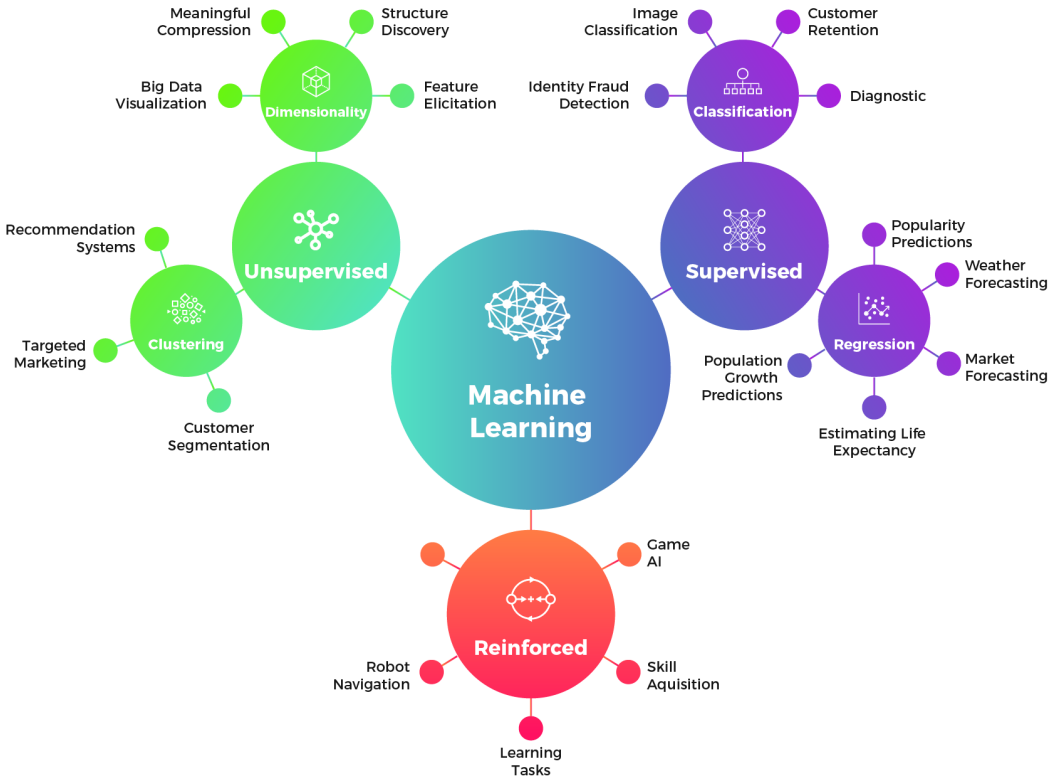
Egy jól implementált tűzfal, IDS/IPS rendszer és egy scrubber kombinációjával támadások rendkívül széles körét kivédhetjük. Azonban fontos látni, hogy teljeskörű védelem megvalósítása még a legkiválóbb megoldások használatával sem valószínű, hogy lehetséges volna. Egész egyszerűen amiatt, mert mindhárom védelmi rendszer előre ismert, korábban már látott támadástípusokat tud csak kezelni, valamint az ezekben alkalmazott programok is csupán hagyományosnak tekinthető algoritmusokat használnak. Például könnyen implementálható egy

[1]Forrás: `https://ctedunet.net/ddos-detection-mitigation-2/`

scrubber rendszerbe olyan mechanizmus, mely azonnal kiszűri a feketelistákon lévő IP-címek forgalmát. Azonban ez a bizonyos lista csupán lassan, és általában csak a támadások felfedezése után tud újabb IP-címekkel bővülni. Egy másik példa lehet, hogy egy scrubbert igen hatékonyan fel lehet készíteni arra, hogy a SYN flood-típusú támadásokat kiszűrje, azonban lehetetlen felkészíteni olyan támadásokra, melyek korábban nem voltak ismertek. A védelmi mechanizmusok fejlődésével természetesen a támadási metódusok is feljődnek, egyre komplexebbek lesznek. Nem elég az, ha csupán az ismert támadásokra készítjük fel a rendszerünket, hiszen bármikor érhet meglepetés valamilyen korábban még nem látott támadás formájában. Ezáltal felmerül az igény arra, hogy valamilyen intelligensebb, fejlődni képes védekezési módot is találjunk, mely megbecsüli a forgalmi mintázatokból annak veszélyességi jellegét, ezáltal képes akár ismeretlen támadásokat is felismerni.

3.2. Bevezetés a gépi tanulásba

A gépi tanulás a modern informatika egyik legdinamikusabban fejlődő ágazata, ami manapság a médiában is igen komoly hangot kap. Maga a fogalom rengeteg különböző metódust, algoritmust, felhasználási szcenáriót foglal össze egyetlen fogalommá, ezért nyilvánvalóan a területet többféleképpen is csoportosíthatjuk.



2. ábra. A gépi tanulás szakterületének egy lehetséges felosztása[2]

Ha fel szeretnénk rajzolni a terület térképét, egy olyasmi ábrát kapunk, mint ami a 2. ábrán látható. Talán egyik legelterjedtebb felosztás az, ami a tanítás módszerei

[2]Forrás: `https://www.rocketsource.com/blog/machine-learning-models/`

szerint tesz különbséget a különböző területek között [9]. Ezen felosztás szerint megkülönböztetjük a felügyelt (supervised), a felügyelet nélküli (unsupervised) és a megerősítéses (reinforced) tanulást. Egyébként gyakran találkozhatunk olyan felosztással is, amely egy negyedik, úgynevezett félig-felügyelt (semi-supervised) tanítást is megnevez. Ez azonban valójában a felügyelt és a felügyelet nélküli módszerek keverékét jelenti, melynek bár hatalmas létjogosultsága van, sokan nem tekintik különálló területnek. Ezen dolgozatban bemutatott feladat megoldása során felügyelt és felügyelet nélküli tanítással foglalkoztam, ezek elméleti alapjait a harmadik fejezet későbbi részeiben részletesebben is bemutatom.

A legfontosabb különbség a két megközelítés között az, hogy milyen adathalmazt használunk fel, és azt hogyan alkalmazzuk egy intelligens rendszer létrehozására. Míg felügyelt tanításnál egy felcímkézett adatbázisunk van, addig a felügyelet nélküli esetben ezekkel a címkékkel nem rendelkezünk. Természetesen ez a tény erősen determinálja azt is, hogy melyik módszer milyen gyakorlati problémákban használatos. A 2. ábrán látható még az is, hogy az egyes kisebb területeken belül, feladattípus szerint hogyan indulhatunk tovább a felosztásban. A felügyelt módszereken belül léteznek osztályozási és regressziós feladatok. Ezekben az a lényeges különbség az, hogy a megtanított modell kimenete folytonos vagy diszkrét értékkészletű. Osztályozási problémákban a modellnek adott darabszámú, előre definiált csoportokba kell osztania a bemenetére adott adatok alapján definiált pontokat, míg regressziós feladatok esetén egyfajta folytonos értékkészletű predikciót várunk el válaszként. A felügyelet nélküli esetben teljesen más megközelítést alkalmazunk. A szükséges címkék hiányában nem tudunk a felügyelt algotirmusokhoz hasonló módszerekkel tanítani, azonban az adatok felhasználásával, a változók közti összefüggések megismerésével így is látványos eredményeket érhetünk el. A legfontosabb feladatokat itt a klaszterezést és az adatok dimenzióinak vizsgálatát jelentik. Előbbi kérdéskör az osztályozási feladatokhoz hasonlóan az adatok által reprezentált pontok közt csoportokat próbál definiálni, azonosítani, míg utóbbit az adatok elemzésében, a dimenzióredukcióban, az adatok közötti további összefüggések keresésében használjuk.

3.2.1. A felügyelt tanítás

A felügyelt (supervised) tanító algoritmusok mindegyikében közös, hogy a tanítás során felhasznált adatbázis minden eleme előre rendelkezik az adott bemenetekhez tartozó elvárt kimeneti címkékkel, és ezt a tanítás során fel is használjuk a tanuló rendszer belső változóinak, súlyainak hangolására [10]. Matematikai szempontból az a cél, hogy a helyes osztályozást megvalósító leképezés paramétereit megtaláljuk. Ezt úgy is szokták mondani - különösen osztályozási feladatok esetén - hogy meg kell találni a különböző címkékkel rendelkező adatponokat helyesen szeparáló hipersíkot az adatok által definiált térben. Különböző algoritmusokhoz különböző matematikai apparátus is tartozik, ezekről a későbbiekben még szót ejtek.

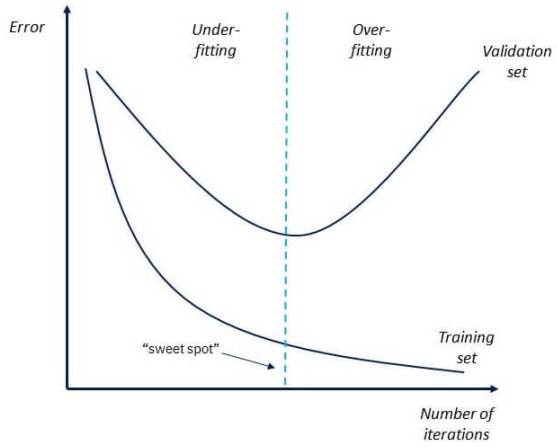
A gyakorlatban természetesen általában azzal kezdjük a munkát, hogy egy feltáró vizsgálati fázisban megismerkedünk a kapott adatbázissal. Ez a fázis különösen fontos, hiszen már ekkor is sok összefüggésre fény derülhet. Ezek után az adatok előfeldolgozási fázisa következik. Ennek a lépésnek az a célja, hogy felkészítsük az adatbázist arra, hogy a benne lévő összes rekord a tanuló

rendszerünk megfelelő be- és kimenete legyen. Például ebben a fázisban gyakran megtisztítjuk az adatsort a kiugró, vélhetően hibás mérés eredményeképpen megjelenő bejegyzésektől, a hiányos rekordoktól, és más egyéb problémás elemektől. Mindezek után a rendelkezésre álló adathalmazt kettéosztjuk, így létrehozva egy tanító és egy teszt halmazt. Emellett gyakran alkalmazott módszer, hogy létrehozunk egy úgynevezett validáló adathalmazt is, ennek célja hasonló a teszt halmazhoz, a különbség csupán annyi, hogy ezeket az adatokat már csak tanítás után használjuk a modell validálására, végső pontossági jellemzésére. A kimenetek minőségét egyébként sok különböző elterjedt metrikával leírhatjuk, feladatfüggő az, hogy mely pontossági jellemző maximalizálása a cél [11]. Miután megtörtént a modell tanítása, vagyis valamilyen numerikus eljárás segítségével megfelelő számú iterációban (epochban) feldolgoztuk a tanító mintahalmazt, valamint a kimeneti címkéknek megfelelően módosítottuk a belső súlyokat és paramétereket, ideális esetben következtetési időben megfelelő kimeneteket fog generálni a rendszer a bemenetek jelentős részére.

Rendkívül fontos az, hogy ez tényleg csak ideális esetben van így. A következtetési időben megfigyelhető pontosság rendkívül sokmindentől függ. Talán a két legfontosabb tényezőként a modell megválasztását és paraméterezését, valamint a rendelkezésre álló adatok mennyiségét és minőségét lehet megjelölni. Ha nincsen megfelelő mennyiségő adatpontunk, amelyet fel tudunk használni a tanítás során, a kész eszköztől elvárható pontosság érthető módon csökken. Léteznek úgynevezett augmentációs eljárások , melyek segítségével gyakorlatilag az adatbázis feldúsítását tudjuk elvégezni, vagyis újabb tanító mintákat tudunk generálni, azonban ezek alkalmazhatósága is korlátozott [12]. A rendelkezésre álló adathalmaz, és az általa leírt adatpontok tulajdonságai is kulcsfontosságúak, ezekre a tulajdonságokra röviden az adathalmaz minősége kifejezéssel szokás hivatkozni. Ha például ellentmondásos adatpontokkal rendelkezünk, vagy sok hiányos cella van a táblázatban, esetleg ha a rögzített adatpontok valamilyen ismeretlen, de jelentős torzítással vagy mérési bizonytalansággal terheltek, vagy bizonyos esetekben megjelennek indokolatlan módon kiugró adatok, melyek valamilyen rossz mérés eredményei, szintén nem várható el a tökéletes pontosság. Természetesen bizonyos előfeldolgozási lépésekkel, például az outlier adatpontok törlésével az esetek többségében elérhetünk némi minőségbeli javulást, azonban a legjobb matematikai módszerektől sem várható el az, hogy egy gyenge minőségű adatbázis felhasználásával tökéletes eszközt tudjunk létrehozni [13]. Ezekben az esetekben szinte mindig az volna a legjobb megoldás, ha az adatok rögzítési módszereit finomítjuk, pontosabb, megbízhatóbb mérési eljárást dolgozunk ki.

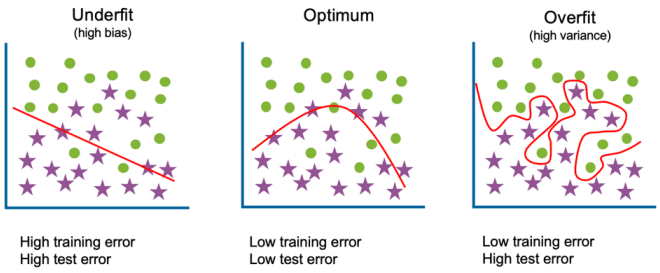
Mindezek mellett rendkívül fontos megemlékezni arról is, hogy szerencsésebb esetben is belefuthatunk az úgynevezett overfitting jelenségbe is, melyre egyébként a magyar szakirodalomban a "túltanulás" vagy a "túlilleszkedés" kifejezésekkel is szokás hivatkozni, és ami különösen a felügyelt modellekre jellemző, legfőképpen akkor, ha nem megfelelő hiperparaméterekkel látjuk azt el [14][15]. Amennyiben felügyelt tanítással elkezdünk egy eszközt tanítani, bizonyos epochszám után érdekes jelenségre figyelhetünk fel: a validációs adatokon mért hiba egy pont után nem csökken (ahogyan azt az underfitting szakaszban tapasztalhatjuk), hanem élesen növekedni kezd. A jelenség értelmezése, hogy rendszer gyakorlatilag túlilleszkedik a

tanító adathalmazra, és emiatt a validációs adathalmazra rosszul teljesít.



3. ábra. A validációs hiba növekedése az epochok számának növelésével[3]

Ezt a növekvő pontatlanságot, melyet általánosítási hibának hívunk az okozza, hogy ekkor a szeparálási hipersíkok - melyek a különböző címkékkel rendelkező tanító adatpontokat választják el egymástól, és amelynek a megtalálása a tanítási folyamat tényleges célja matematikai értelemben - túlzottan finoman rásimulnak a tanító adathalmaz pontjaira, ezáltal megtanulják azok pontatlanságait, mérési hibáit is, valamint a határpontok környezetében lévő tesztpontokat a sík rossz oldalára képezik le.



4. ábra. A szeparáló hipersík alul- és túlilleszkedésének szemléltetése egy egyszerű, kétdimenziós példán[4]

A túlilleszkedés jelensége azért is különösen érdekes, mert a hétköznapi intuíciónkból nem feltétlenül ez következne. Azt gondolnánk, hogy minél tovább tanítunk, minél tovább finomítjuk a belső súlyokat, annál jobb végeredményt fogunk kapni. A valóság azonban nem ez, és ebből már azt is könnyen láthatjuk, hogy a valós feladatok túlnyomó többségében - hacsak nem mesterségesen generált

[3]Forrás: `https://www.javatpoint.com/overfitting-in-machine-learning`
[4]Forrás: `https://www.javatpoint.com/overfitting-in-machine-learning`

adatpontokkal dolgozunk - szinte lehetetlen az, hogy 100%-os pontosságot érjünk el, hiszen a tanítási görbéből látható, hogy a validációs halmazon mért hiba egy bizonyos szint alá nem vihető. Azt, hogy ez probléma e, általában a feladat specifikációja határozza meg. Bizonyos alkalmazási területeken teljesen elfogadható a hiba, különösen akkor, ha az csak ritkán fordul elő, sokszor viszont szörnyű következményekkel járhat. Emiatt fontos, hogy az ilyen kritikus alkalmazások esetén mindig bizonyosodjunk meg arról, hogy nem a mesterséges intelligencia az egyetlen döntéshozó algoritmus. Általánosságban elmondható, hogy minél nagyobb paraméterszámmal rendelkezik a tanuló rendszer a tanító adatokhoz képest, annál valószínűbb a túltanulás jelensége. Kimnodható tehát, hogy az algoritmus megválasztásakor ez is egy igen fontos szempont, amit mindenképpen figyelembe kell vennünk. Egyébként lézetnek olyan eljárások, melyekkel jelentősen csökkenthetjük az overfitting valószínűségét, azonban ezek általában feladatspecifikus módszerek. Ilyen technika például a pruning ("metszés") vagy bizonyos augmentációs technikák [12][16]. Emellett pedig természetesen bizonyos modellek esetén a tanítási ciklusok csökkentésével is el tudjuk kerülni ezt a kellemetlenséget. Ezek azonban nem képezik ennek a dolgozatnak a tárgyát.

Ezen kérdéskör elvezet minket egyébként még egy érdekes és igen fontos szemponthoz. Érdemes belegondolni abba, hogy mit is értünk a modell végső pontosságán. Ez egy olyan pontossági metrika, melyet a rendelkezésre álló adathalmaz egy részhalmazán tapasztaltunk. Azonban nem mindegy az, hogy ezen adatpontok a lehetséges bemenetek értelmezési tartományának mekkora részét fedik le. Az összetett gyakorlati feladatok többségében erről sajnos kevés információnk van. Ha viszont valamit tudunk, akkor az álalában az, hogy a bemeneti változók értelmezési tartományának csak kicsi területét tudjuk pontokkal lefedni, még rendkívül nagy mennyiségű adat rendelkezésre állása esetén is. Ebből az következik, hogy gyakorlatilag a rendszer számos bemeneti kombinációval sem a tanítás, sem a tesztelés során nem találkozik. Ez azért igen nagy probléma, mert az elterjedt algoritmusok többségével semmilyen garanciát nem tudunk arra vonatkozóan vállalni, hogy ezekre a bemeneti kombinációkra milyen választ fog adni a megvalósított mesterséges intelligencia, ezekre a modellek válasza nem determinisztikus. Különösen nagy problémát okoz ez például az autonóm járművek fejlesztésének területén [17][18]. Egy átlagos autós olyan hatalmas számosságú különböző forgalmi szituációval talalkozhat, tehát olyan hatalmas a bemeneti változók kombinációinak tere, hogy gyakorlatilag teljesen lehetetlen minden esetet tanító és tesztelő adatpontokkal lefedni, még összetett szimulációs és augmentációs technikákkal is. Így a gyártók által létrehozott modell validálása a való életben gyakorlatilag lehetetlen, legalábbis jelenlegi tudásunk szerint. Ennek következtében a fejlesztők a felelősséget sem vállalják abban az esetben, ha baleset ér egy önvezető módban közlekedő autót, valamint valószínűsíthető, hogy a korábban megálmodott 100%-ban autonóm járművek víziója még hosszú ideig csupán utópia marad.

Fontos megjegyezni, hogy a fentebb felsorolt hátrányok és nehézségek nem a mélytanulási megoldások alkalmazása ellen szólnak. Ennek a területnek is, ahogyan mindennek megvannak a maga kihívásai és korlátai. Ezek felhívják a figyelmet arra, hogy a mesterséges intelligencia forradalmi elterjedése közben nem szabad elfelejteni azt, hogy a hagyományos algoritmusokkal is foglalkozzunk,

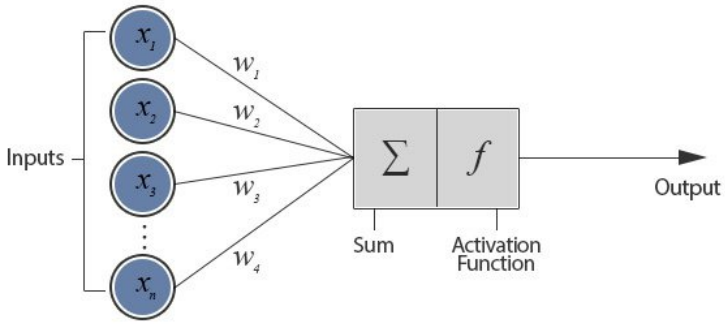
azokat is fejlesszük és alkalmazzuk, hiszen nem minden feladatot lehetséges vagy érdemes a mesterséges intelligenciára hagyni. Azokon a területeken viszont, ahol a hagyományos módszerek korlátaiból fakadóan gyengén teljesítenek, a tanuló algoritmusok rendkívül hasznosak lehetnek, annak ellenére is, hogy használatuk közben gyakran megoldadó nehézségekbe ütközünk.

Felügyelt tanítási eszközökből rengeteg létezik. Számos különböző módszerrel, sok érdekes és olykor igen komplex matematikai apparátust felhasználva születtek meg azok az algorimtusok, melyekkel manapság szinte minden területen találkozhatunk, akár osztályozási, akár regressziós feladatokban. Léteznek általánosabban felhasználható megoldások, mint például az MLP (Multi Layer Perceptron), az SVM (Support Vector Machine), vagy a Random Forest, de vannak specifikusabb algoritmusok is, mint például a természetes nyelvfeldogozásban használt transzformer hálók vagy a képfeldolgozásban használt konvolúciós neurális hálózatok. A megfelelő algoritmus kiválasztása, és annak hiperparamétereinek helyes megválasztása gyakorlatilag szinte egy önálló szakterületté nőtte ki magát, hiszen rengeteg szempontot figyelembe kell venni. Minden algoritmusnak és modellnek megvannak a maga előnyei és hátrányai, nem létezik olyan eszköz, mellyel minden problémát meg lehet oldani. A modell megválasztásánál figyelembe kell venni a rendelkezésre álló adatok mennyiségét és minőségét, a feladat komplexitását, az elvárt pontosságot, a tanításhoz rendelkezésre álló hardver eszközöket, a következtetési időben történő tévedés következményeit és még sok-sok tényezőt.

3.2.2. Az MLP

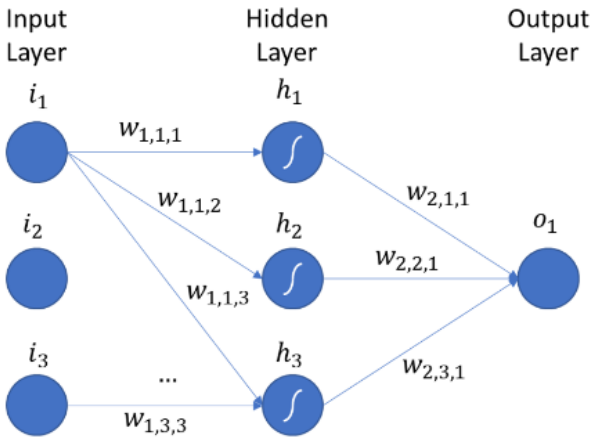
A többrétegű perceptron, vagyis az MLP az egyik legismertebb és legelterjedtebb eszköz a gépi tanulás területén [19]. Ennek a széleskörű elterjedtségnek az az oka, hogy egyszerűen alkalmazható, valamint hogy egy általános célú megoldás, számos osztályozási és regressziós feladatban használható, általános numerikus adatok feldolgozására megerősítéses tanulási feladatok során. Éppen emiatt azonban speciális területeken, mint például a hang- vagy képfeldolgozásban találhatunk ennél sokkal optimálisabb megoldásokat is. A tanítás során háló súlyait egy iteratív módszerrel hangoljuk a tanító minták alapján. A tanító algoritmus a tanító adatok címkéinek és a modell kimenetének a különbségeként egy költségfüggvényt képez. Ez a költségfüggvény tekinthető a leképezés hibájának is, mely egy úgynevezett hibafelületet feszít ki a mért változók terében. Az algoritmus célja, hogy megtaláljuk ezen hibafelület minimumpontját, ezzel maximalizálva a rendszer döntési képességének pontosságát. Ennek a módszernek egy elterjedt gyakorlati megvalósítása a Gradient Descent eljárás.

Az MLP felépítésének alapja a Rosenblatt perceptron, mely a lentebb található 5. ábrán figyelhető meg. A perceptron működése matematikai szempontból egyszerű: a bemenetek súlyozott összegét valamilyen aktivációs függvény bemeneteként felhasználva kapjuk meg a perceptron kimenetét. A tanító eljárás célja, hogy ezeket a súlyokat a tanító mintákon mért hibának megfelelően hangoljuk. Az aktivációs függvényeknek számos különböző fajtája létezik. Ezek közül egyébként a legelterjedtebbek a ReLu (Rectified Linear Unit), a tangens hiperbolikusz és a Sigmoid [20].



5. ábra. Az elemi perceptron felépítése[5]

Számos egyszerű szeparálási feladat akár egyetlen perceptron használatával is megoldható, ezeket lineárisan szeparábilis feladatoknak hívjuk. Természetesen a gyakorlati problémák többsége ennél jóval nagyobb bonyolultságú, ekkor azonban kézenfekvő módon adódik, hogy használjunk fel a feladatmegoldás során egy több perceptronból álló hálózatot. Ez a hálózat az MLP (Mulit Layer Perceptron), melynek egy egyszerűsített, sematikus képét a 6. ábrán láthatjuk.



6. ábra. Az MLP sematikus ábrája[6]

A háló architektúrája három külön szekcióra bontható: van egy bemeneti réteg, mely a bemeneti változók számosságával egyenlő számú pontot tartalmaz, van egy kimeneti réteg, mely az osztályozási feladat lehetséges kimeneteivel van kapcsolatban, illetve van egy vagy több rejtett réteg, mely a két fentebb említett részt köti össze. A rétegeket úgynevezett full-mesh módon kötjük össze, tehát adott rétegben lévő összes neuron a következő réteg összes neuronjának bemenetét képezi, mind-mind más és más súlyokkal. A modell két legfontosabb hiperparamétere, a rejtett rétegek száma, valamint azokban lévő neuronok száma. Ezeken felül pedig természetesen az aktivációs függvényt is meg kell választanunk. Ezek helyes megválasztása egyébként a legtöbb esetben nem feltétlenül triviális, gyakran olyan

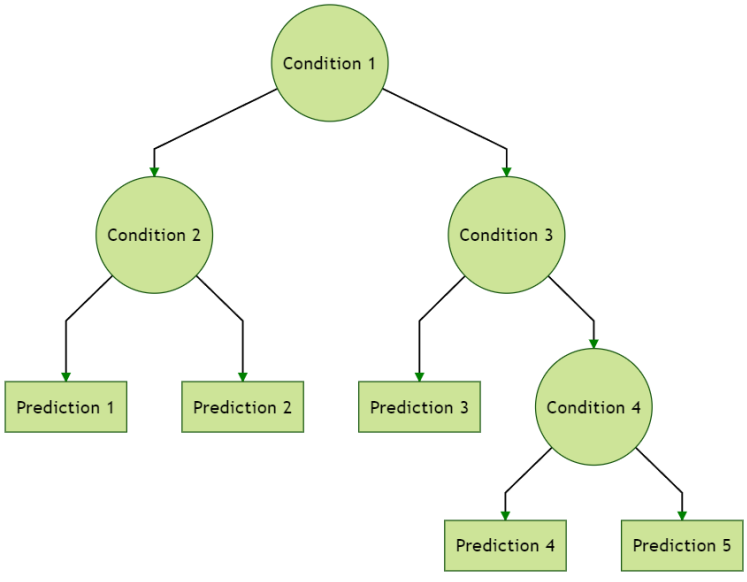
[5]Forrás: `https://www.quora.com/What-is-a-perceptron-in-neural-networks`

[6]Forrás: `https://www.analyticsvidhya.com/blog/2022/10/multi-layer-perceptrons-notations-and-trainable-parameters/`

megközelítéssel élünk, hogy több modellt is létrehozunk és tanítunk, és később a legjobban teljesítőt alkalmazzuk. Fontos figyelembe venni azt, hogy értelemszerűen minél nagyobb modellt használunk, a tanítás annál nagyobb hardverigényű lesz, valamint annál inkább megnövekszik a túlilleszkedés valószínűsége. Így nem teljesül az az elv, melyszerint a bonyolultabb modell pontosabb leképezést képes megvalósítani. Ezért gyakorlati problémák jelentős részében maximum egy vagy két rejtett réteget alkalmazunk, ezzel kellőképpen bonyolult leképezés is megvalósítható, miközben a tanítási idők nem növekednek meg drasztikus mértékben. Külön kategóriát képeznek azok a problémák, ahol két rejtett réteg nem elegendő, ezek az úgynevezett mély neurális hálózatok [21]. Ennek a területnek számos sajátos kihívása van, mint például a kihűlt neuronok kezelése, a felrobbanó gradiens probléma és a túlilleszkedés. Ezek nagyon izgalmas és érdekes témák, azonban most nem tárgyai ennek a dolgozatnak.

3.2.3. A Random Forest

A Random Forest az MLP mellett egy szintén igen széles körben alkalmazott, felügyelt módon tanítható megoldás, amellyel kiválóan megoldhatunk osztályozási feladatokat [22]. Azonban fontos, hogy működése a neurális algoritmusokhoz képest jelentős mértékben eltér. Ennek következtében egyébként könnyebben alkalmazható nagy adathalmazok esetén is, a tanítás jelentősen kevesebb hardveres erőforrást igényel a neurális alapú megoldásokhoz képest. A módszer legfontosabb tulajdonsága, hogy többségi döntésen alapul, melynek alapját a döntési fák képezik. A tanítás során létrehozunk több darab dötnési fát, melyeket az adatbázis egy-egy részhalmazán tanítunk.



7. ábra. Egy döntési fa sematikus ábrája[7]

A predikció úgy képződik, hogy minden egyes fa bemenete megkapja az adatokat, majd a legtöbb azonos kimenetet generáló címke lesz a prediktált címke.

[7]Forrás: `https://machinelearningtheory.org/docs/Random-Forest/tree/`

Alapvető fontosságú, hogy megértsük azt, hogyan is működik egy döntési fa [23]. Ennek egy egyszerűsített sematikus rajzát figyelhetjük meg a 7. ábrán. Ahogy az látható, a döntési fa gyakorlatilag egy fa gráf által reprezentálható, ahol az egyes csomópontokban bináris döntést hozunk a gyökér, mint bemenet által kapott adatpontról, majd a válaszként megjelenő címkét a levelek valamelyikén láthatjuk. A csomópontokban megjelenő döntésekhez, egy-egy változó (feature) küszöbértékét rendeljük a tanító minták alapján. Fontos, hogy ezeket a változókat úgy válasszuk ki, hogy az a lehető legjobban reprezentálják a tanító pontokat, ezáltal a helyesen megválasztott döntési küszöbök maximalizálják az úgynevezett information gain-t.

Számos egyszerű feladat megoldható akár egyetlen döntési fa felhasználásával is, azonban a gyakorlatban általában ennél jóval bonyolultabb feladatokkal találkozhatunk. Ekkor eljárhatunk úgy, hogy sok különböző döntési fát hozunk létre, amelyeket párhozamosan alkalmazva, egy szakértőegyüttesként használunk. A végső döntést az adja, hogy mely címke lett a legtöbb fának a kimenete. Természetesen nem biztos, hogy mindig minden fa kimenete helyes lesz, de várhatóan a többségé igen. A fák képzése, vagyis a tanítás speciális módon történik: ha minden alkalommal, minden adatpont minden tulajdonságát felhasználnánk az igen nagy számítási igényt jelentene, azonban erre nincsen szükség. A tanító algoritmus minden fa képzésekor a tanító adathalmaz véletlenszerűen kiválasztott elemeit veszi, majd ezeknek a pontoknak is véletlenszerűen kiválasztott néhány tulajdonságaiból képez egy részhalmazt, majd ebből képezi az adott döntési fát. Ezzel a megoldással megfelelő számú fa esetén várhatóan az összes feature több kombinációban is figyelembe lesz véve valamely fa által. Így várhatóan következtetési időben is helyes döntéseket fog hozni a létrejött szakértőegyüttes.

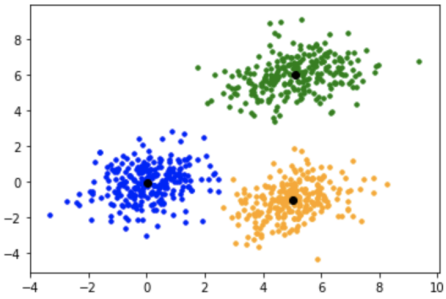
3.2.4. A felügyelet nélküli tanítás

A másik, szintén igen elterjedt út a felügyelt módszerek mellett a felügyelet nélküli tanító algoritmusok [24]. Ezeknél a módszereknél abban rejlik a nagy különbség a felügyelt megoldásokhoz képest, hogy a tanítás során felhasznált adathalmaz nem rendelkezik az elvárt kimeneti címkékkel. Ez nem feltétlenül jelenti azt, hogy a szükséges kimenettel egyátalán nem is rendelkezünk, csupán azt, hogy a tanító algoritmus ezeket nem kapja meg. Sőt, gyakran a kész modell teszteléséhez szükséges, hogy ezeknek az adatoknak birtokában legyünk. A felügyelet nélküli megoldások felhasználási területei nagyban különböznek a felügyelt modellekétől. Ennek több oka is van. Elsősorban az, ami ezeknek a metódusoknak igen nagy előnye, hogy olyan problémák esetében is használhatóak, ahol az adatbázis méretének következtében a címkézés kézzel történő elvégzése gyakorlatilag lehetetlen lenne. Emellett fontos differencia a felügyelt módszerekhez képest, hogy az elérhető pontosság általában korlátozottabb. A legfontosabb területek, ahol előszeretettel használnak felügyelet nélküli algoritmusokat, azok a klaszterezési és az anomália detekciós feladatok. Ezek tulajdonképpen rokon feladatoknak is nevezhetőek, gyakran láthatjuk, hogy bizonyos klaszterező algoritmusok anomáliák azonosítására is alkalmasak.

3.2.5. A klaszterezés és az anomália detekció

A klaszterezés kapcsán is fontos, hogy hasonló kitételeket tegyünk, mint a megerősítéses tanulás esetén. Rendkívüli módon meghatározza az elérhető eredményeket az, hogy milyen mennyiségű és minőségű adat áll rendelkezésre. Ha kevés vagy rossz minőségű adatunk van, érthető módon szerényebb elvárásokkal érdemes megkezdenünk a munkát. Ha adatbázisunkban az egyes pontok rendelkeznek olyan mért tulajdonsággal, melynek eloszlása véletlenszerű minden címke esetén, vagyis adott dimenzió mentén a különböző címkéjű csoportok nem válnak el megfelelő módon egymástól, akkor a klaszterező algoritmustól sem elvárható az, hogy ezt megtegye. Ilyen adatbázisok esetén szélsőséges esetben előfordulhat az, hogy bizonyos adatbázis oszlopok elhagyásával pontosabb eredményt kapunk. Emellett szintén nagyon fontos, hogy az adott feladathoz a megfelelő algoritmust válasszuk, a megfelelő hiperparaméterek beállításával. Azonban a gyakorlatban ez sokszor még a megerősítéses tanulás esetén történő modellezésnél is nehezebb feladat lehet. Ennek oka, hogy kellően nagyméretű adatbázisok esetén általában kevés előzetes információnk van az egyes adatpontok eloszlásáról, elhelyezkedéséről. Éppen emiatt mindenképpen szükséges egy előzetes feltáró adatelemzési munkafázis készítése, valamint a modellezés során erősen ajánlott, hogy több különböző megoldást is megvizsgáljunk. Ahogyan azt a megerősíteses tanulásnál láttuk, itt is széles választék áll rendelkezésünkre, sok-sok különböző algoritmus, melyeknek mind megvannak a maguk előnyeik és hátrányaik. A fejezet későbbi részében erre a témára még visszatérek.

A klaszterezés lényegének magyarázatához fontos, hogy tisztában legyünk azzal, mit is jelentenek az adatbázisunkban lévő bejegyzések. Ezek a sorok gyakorlatilag értelmezhetőek olyan pontokként, melyek a mért vagy rögzített tulajdonságok által meghatározott, sokdimenziós térben helyezkednek el. Természetesen ezt háromnál több dimenzió esetén igen nehéz elképzelni, de gyakorlatilag minden egyes rögzített sor az adatbázisban egy-egy ilyen pontot reprezentál. Intuitívan adódik az a tény, hogy mikor sok-sok különböző jellegű eseményt, tárgyat vagy bármi egyebet rögzítünk, az egymáshoz hasonló jellegű mérési pontok várhatóan egymáshoz közelebb fognak elhelyezkedni, hiszen az adott tárgyak vagy események tulajdonságai mind-mind leírhatóak valamilyen sűrűségfüggvénnyel, melyhez egyébként tapasztalati várható értéket és szórást is rendelhetünk. Eztuán már csak az a kérdés, hogy hány adatpontot tudunk rögzíteni, hiszen minél több adatunk van, annál jobban látható, hogy az azonos karakterisztikájú mérések eredményei egy-egy "felhőbe" csoportosulnak. A 8. ábra egyszerű példáján az figyelhető meg, hogy egy ilyen kevés dimenziós eloszlás esetén a klaszterezést gyakorlatilag szemmel is el tudjuk végezni, azonban ahogy az látszik, a K-means algoritmus ezt helyesen meg is tette nekünk.

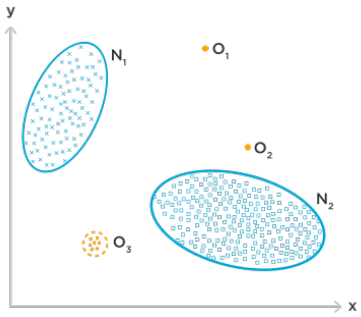


8. ábra. Egy egyszerű, kétdimenziós eloszlás klaszterezése K-means algoritmussal[8]

Természetesen ez a feladat nagyobb dimenziószám és más eloszlás esetén közel sem ennyire triviális. Számos gyakorlati területen felmerül az igény arra, hogy ezt a csoportokba soroló tevékenységet algoritmizáljuk. Például széles körben alkalmaznak klaszterezési megoldásokat a különböző marketinggel és kereskedelemmel foglalkozó szolgáltatások, valamint a közösségi oldalak is. A hasonló érdeklődésű felhasználók bizonyos mért metrikák alapján várhatóan egy-egy csoportba tömörülnek, például hasonló tartalmakat fogyasztanak, és hasonló termékeket néznek meg. Ezt remekül ki tudják használni arra, hogy a hírdetéseket és a tartalom ajánlásokat irányított módon juttassák el a felhasználókhoz, ezzel növelve azok relevanciáját az adott személyre nézve.

Az anomália detekció (vagy másnéven outlier detekció) valójában a klaszterezéssel rokon feladatként értelmezhető [25][26]. Ezekben a feladatokban is egyfajta klaszterezést végzünk el, de annak kimenetele a címkéket tekintve kevésbé releváns számunkra. Sokkal inkább az a cél, hogy megkeressük azokat az adatpontokat, amelyeket nem lehet egy-egy klaszterbe besorolni, amelyek valamilyen formában kiugró értékeket tartalmaznak, így a megszokott "felhőtől" távolabb helyezkednek el ebben a sokdimenziós térben. Számos különböző oka lehet annak, hogy ilyen rekordok előfordulnak: a gyakorlati problémák egy jelentős részében ilyenkor valamiféle mérési hiba eredményeiről beszélhetünk, azonban természetesen előfordulhat az is, hogy valamiféle egyedi, különös jelenség okozza az anomáliák megjelenését. Gyakori igény az, hogy ezeket a pontokat felismerjük, megtaláljuk és elhárítsuk ezek okozóját. Így nyilvánvalóvá válik, hogy egy ilyen anomália detekcióra alkalmas algoritmust például kibervédelmi események detektálására is lehetne alkalmazni, ugyanis számos esetben látható, hogy például a DDoS támadások egy része a normál forgalmi karakterisztikáktól bizonyos metrikákban jelentősen eltér. A 9. ábrán egy egyszerű példán be is tudom mutatni, hogy pontosan hogyan is kellene elképzelni ezeket a kiugró adatokat.

[8]Forrás: `https://www.geeksforgeeks.org/ml-k-means-algorithm/`



9. ábra. Egy egyszerű anomália detekciós példa[9]

Természetesen egy anomália detekciós feladat is több különböző erre alkalmas algoritmussal is megoldható. Emellett kiemelendő, hogy számos klaszterező algoritmus is képes kiugró adatpontok azonosítására, azonban fontos, hogy ez nem mindig igaz. Számos olyan algoritmus is létezik, mely minden pontot mindenféleképpen valamely klaszterbe sorol, attól függetlenül, hogy az mennyire távol helyezkedik el az adott csoporttól. Ahogyan azt a korábbi fejezetekben is már kifejtettem, itt is rendkívüli mértékben meghatározza az elérhető pontosságot az adathalmaz minősége és mérete, hiszen a klaszterező algoritmusok sem tudnak hiányos, vagy nem megfelelő pontokból álló klasztereket helyesen azonosítani, így nyilvánvaló, hogy ilyen esetekben az anomália detekciótól is csak szerényebb teljesítményt várhatunk el. Nyilvánvalóan a gyakorlati feladatokban a 9. ábrán látott esetnél jóval bonyolultabb és nagyobb dimenziószámú problémákkal találkozunk, így különösen érdekes kérdés annak vizsgálata, hogy egy-egy adatpont osztályozása során hol húzzuk meg a határvonalat a klaszterekhez tartozó és a kiugrónak tekintett adatok között. Ha túlzottan szigorú határokat szabunk, a modellünk nagy valószínűséggel sokkal több pontot fog anomáliaként értékelni, ennek eredményeképpen nagy pozitív falsrátát fog produkálni. Amennyiben viszont túlzottan gyenge megkötéseket teszünk, valószínűleg nem fogjuk tudni az anomáliák jelentős részét detektálni, így a pontosság szintén csökken, a negatív irányban. Ezt egyébként általában valamely általunk megadott hiperparaméter szabályozza, ezáltal itt is elmondható, hogy érdemes lehet kísérletezni különböző paraméterezéssel inicializált modellekkel is.

3.2.6. Választás a különböző klaszterező megoldások között, valamint néhány elterjedt algoritmus bemutatása

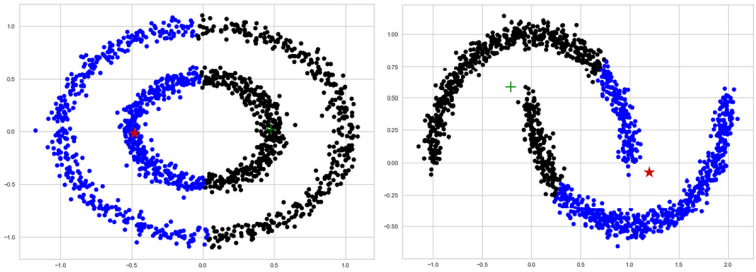
Ahogyan a felügyelt tanító algoritmusoknál is láttuk, a felügyelet nélkül működő, klaszterezési feladatokra alkalmazott metódusok esetén is igen széles kínálatból választhatunk. Ezeknek részletes, teljeskörű bemutatása akár különálló dolgozatot is kitenne [27], így ezt nyilvánvaló okokból most nem teszem meg, azonban ebben a fejezetben szeretném röviden összefoglalni és összehasonlítani néhány elterjedt algoritmus legfontosabb tulajdonságát. Ezeknek a tulajdonságoknak a sokszínűsége leginkább annak tudható be, hogy minden megoldás más és más numerikus eszköztárat használ, máshogy valósul meg a "motorháztető alatti" működés. Az

[9]Forrás: `https://www.spotfire.com/glossary/what-is-anomaly-detection`

alkalmazott modell megválasztásakor számos szempontot figyelembe kell vennünk. Ilyen természetesen a rendelkezésre álló adatok minősége és mennyisége. Vannak olyan algoritmusok, melyek könnyen alkalmazhatóak sok elemből álló adatsorokon is, míg mások működésükből adódóan nem túlzottan skálázhatóak. Másik fontos tényező az, hogy milyen hiperparaméterekkel rendelkezik az adott modell. Gyakoriak azok a megoldások, ahol a felhasználótól élőre elvárt hiperparaméter az, hogy hány klasztert szeretne azonosítani a tanító minták között, míg más algoritmusok önállóan határozzák meg az azonosítható klaszterek számát. Hogy ezt előnyként vagy hátrányként értékeljük e, az feladatfüggő. Ha előre ismert az, hogy hány különböző csoportból kerültek ki a tanító minták, akkor érdemesebb lehet olyan algoritmust választani, ahol ez megadható hiperparaméterként megjelenik. Azonban, ha nem tudjuk, vagy nem vagyunk biztosak abban, hogy az adathalmaz hány csoportot tartalmazhat, érdemes lehet olyan klaszterező megoldást használni, ahol ez nem elvárt bemenet. A fent említett kérdéshez szorosan kapcsolódik a következő fontos szempont, méghozzá az, hogy az algoritmus képes e kiugró adatokat, anomáliákat detektálni. Egyes algoritmusok erre képesek, ekkor általában a kimeneten egy különálló címkét kapnak ezek az outlier adatpontok, míg más megoldások minden egyes mintát mindenképpen valamely klaszter tagjaként azonosítanak, függetlenül attól, hogy az milyen távol van a klaszter többi tagjától. Ezen felül még az utolsó fontos szempont, amit kiemelnék az az, hogy érdemes figyelembe venni a minták eloszlását, a klaszterek várható alakját. Ez a szempont kicsit nehezen megfogható, különösképpen azért, mert erről a gyakorlatban általában igen kevés információnk van, különösen akkor, hogyha sokdimenziós problémát vizsgálunk. Léteznek bizonyos nevezetes eloszlások, melyeket néhány klaszterező algoritmus képes, míg mások képtelenek helyesen osztályozni a mély matamatikai működésükből kifolyólag. Ezek a nevezetes eloszlások például a spirál alakú, az egymásba fonódó, és a körkörös elrendezésű klaszterek. Mivel általában tényleg semmilyen előzetes információnk nincsen arról, hogy mégis nagyjából hogyan néz ki az adatsorunkban megjelenő minták eloszlása, és hogy fellelhető e ilyesfajta speciális eloszlás valahol, általában a legjobb megoldás az, ha több különböző modellt is készítünk és tesztelünk, majd ezek közül a legjobb teljesítményűt alkalmazzuk a továbbiakban.

Az egyik legelterjedtebb megoldás a K-means klaszterező algoritmus, melyet a vizsgálatok során én is aktívan alkalmaztam [28]. A K-means fontos tulajdonsága, hogy a felhasználónak hiperparaméterként specifikálnia kell a klaszterek számát a szóban forgó adatbázisban. Ezen felül az algoritmus nem képes kiugró adatpontok, anomáliák azonosítására, minden egyes pontot egy-egy klaszterbe sorol. Ha röviden össze akarnánk foglalni azt, hogy matematikai szempontból hogyan működik ez a megoldás, azt mondhatjuk, hogy lényegében egy minimumkeresésről van szó. Az algoritmus véletlenszerűen helyez el a térben a hiperparaméterként megadott klaszterek számának megfelelő pontot, melyeket iteratív számítással úgy mozgat a térben, hogy a tanítóminták által meghatározott pontoktól mért euklideszi távolságaik összege a lehető legkisebb legyen. Ezáltal elérhető az, hogy ezek a pontok bekonvergálnak egy-egy olyan lokációba, amely az adott klaszter középpontjának tekinthető. A modell ezek után az osztályozást úgy végzi, hogy egy adott mintapont abba a klaszterbe fog tartozni, amely klaszter középpontjához

a legközelebb esik. Ennek a működésnek több következménye is van. Egyrészről a működés igen gyorsnak mondható, még nagy adatbázisok esetén is, ami hatalmas előnyt jelent. Azonban azt is láthatjuk, hogy például a korábban említett speciális eloszlások osztályozása esetén ez a modell komoly korlátokba ütközik. Ezt a 10. ábrán is megfigyelhetjük egy példán. Az ábrán azt is láthatjuk, hogy hol helyezte el a modell az általa azonosított klaszterek középpontjait.



10. ábra. A koncentrikusan és a spirálisan elhelyezkedő klaszterek helytelen azonosítása K-means klaszterezéssel[10]

Természetesen ez nem azt jelenti, hogy a K-means nem jó klaszterező algoritmus, csupán érdemes lehet figyelembe venni azt, hogy nem minden eloszlás esetén fogunk vele helyes eredményt kapni, bizonyos esetekben más algoritmusok felhasználásával lehet, hogy jobb eredményeket érhetünk el. Alternatív megoldás lehet például a DBSCAN (density-based spatial clustering of applications with noise) algoritmus [29]. Ez a megoldás a K-means-hez hasonlóan jól skálázható nagy adatbázisokra is, azonban előnye ez előbbivel szemben, hogy nem szükséges specifikálni a felfedezendő klaszterek számát. Emellett alkalmas arra is, hogy anomáliákat, kiugró adatokat ismerjen fel. Azonban hátránya, hogy az eredmény pontossága igen érzékeny arra, hogy mekkora adathalmazzal dolgozunk, valamint, hogy milyen hiperparamétereket állítunk be. Vagy szintén népszerű megoldás az OPTICS (ordering points to identify the clustering structure) algoritmus, mely szintén jól skálázható, és hiperparaméterként csupán az elvárt minimális klaszterméretet kell megadnunk [30]. A K-means-től jelentősen eltérő numerikus működésükből adódóan mindkét algoritmus képes arra, hogy a fentebb említett koncentrikus és spirális mintaeloszlásokat helyesen kategorizálja. Ezek mellett pedig természetesen még számos megoldás létezik, mind-mind különböző előnyökkel és hátrányokkal.

Az általam felhasznált és fentebb bemutatott algoritmusok kiválsztásakor a legfőbb motivációm az volt, hogy olyan megoldást válasszak, melyeket gyakran alkalmaznak a gépi tanulási feladatok minden területén, és amelyek ezáltal felhasználóbarát implementációval valamint széles körben elérhető és jól dokumentált elméleti ismeretanyagokkal is rendelkeznek. Emellett az is fontos szempont volt, hogy a modell könnyen alkalmazható és skálázható legyen, hiszen igen nagy méretű adatbázissal dolgoztam. Ezért esett a választás az MLP-re, a Random Forest-re, valamint a K-means és a DBSCAN algoritmusokra.

[10]Forrás: `https://towardsdatascience.com/explain-ml-in-a-simple-way-k-means-clustering-e925d019743b`

4. Az adatbázis feltáró vizsgálata

A feladat lényegi részének megkezdése előtt kulcsfontosságú, hogy a rendelkezésre álló adatbázissal mélyrehatóan megismerkedjünk, megértsük azt, hogy mit írnak le az egyes jellemzők. Éppen ezért fontos, hogy egy feltáró adatelemzési fázis során megvizsgáljuk a kapott adatokat, néhány hagyományos statisztikai eszköz segítségével. Ahogy az már korábban elmítésre került, a munkám során az Aitia International Zrt. által rendelkezésemre bocsátott adatbázissal dolgoztam. A cég az ügyfelei által üzemeltetett szolgáltatások és szerverek kommunikációját a nap 24 órájában monitorozza. Ezek a monitorozó megoldások rengeteg elemi eseményt detektálnak, melyekről folyamatosan rekordokat készítenek. Ezeknek a rekordoknak egy, körülbelül az elmúlt másfél évet lefedő részét bocsátották rendelkezésemre. Naponta akár sok ezer ilyen eseményt is láthatunk a hálózaton, így értelemszerűen a teljes adatbázis hatalmas méretű: ha csak egy-egy hónap adatait vizsgáljuk, akkor is 250-300 ezer rekorddal van dolgunk. Ez a nagyságrend már egyértelműen Big Data jellegű problémává teszi ezt a feladatot. Emellett rendkívül fontos kiemelni azt a tényt, hogy ezek az adatok valós ipari körülmények között rögzített forgalmi mérésekből származnak, aktuális forgalmi mintázatokat tartalmaznak. A teljes adatbázis valójában két különálló táblázatból áll: az egyik egy aggregált tábla, amely minden egyes támadást, gyanús forgalmat, és jóindulatú nagyobb forgalmi eseményt csupán egyszer, aggregált formában tartalmaz. Ennek egy rövid részlete látható a 11. ábrán. A különösen nagy jelentőséggel rendelkező paraméterek és metrikák fejlécét az ábrán világoskék színnel jelöltem meg, valamint a érintett IP címeket kitakartam.

Attack ID	Card	Victim IP	Port number	Attack code	Detect count	Significant flag	Packet speed	Data speed	Avg packet len	Avg source IP count	Capture link	Start time	End time	White list flag	Type
100001	sga10gq0		60742	High volur	1	0	61100	75	1296	3	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100002	sga10gq0		58203	High volur	1	0	105500	127	1254	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100003	sga10gq0		60136	High volur	1	0	61200	75	1296	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100004	sga10gq0		62369	High volur	2	0	90750	109	1254	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100005	sga10gq0		51492	High volur	1	0	56400	73	1317	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100006	sga10gq0		64664	High volur	3	0	67067	82	1296	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100007	sga10gq0		54199	High volur	108	0	70028	88	1284	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100008	sga10gq0		61342	High volur	3	0	69800	86	1286	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100009	sga10gq0		52597	High volur	3	0	89734	108	1258	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100010	sga10gq0		51947	High volur	3	0	105034	126	1265	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100011	sga10gq0		36800	High volur	3	0	69634	86	1290	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100012	sga10gq0		49450	High volur	4	0	68301	84	1291	2	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100013	sga10gq0		59218	High volur	2	0	68100	83	1296	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100014	sga10gq0		36800	High volur	1	0	98900	122	1296	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100015	sga10gq0		60232	High volur	1	0	67300	83	1296	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100016	sga10gq0		52910	High volur	1	0	74800	90	1278	1	https://gig	2022-04-1	2022-04-1	0	Normal traffic
100017	sga10gq0		47021	High volur	1	0	60700	81	1395	2	https://gig	2022-04-1	2022-04-1	0	Normal traffic

11. ábra. Egy rövid részlet a támadások és események táblájából

Látható tehát, hogy egy címkézett adatbázissal van dolgunk. Az egyes rekordok mind-mind valamilyen biztonsági eseményt reprezentálnak, melyek 3 osztályba kerültek besorolásra: normál forgalom, gyanús forgalom, és DDoS támadás. Hogy jobban megértsük, mit is jelentenek ezek az események, az 1. táblázatban ezeknek egy rövid összefoglalója olvasható.

Típus	**Rövid leírás**	**Szükséges reakció**
Normal traffic	Jóindulatú forgalmi karakterisztikával rendelkező, azonban a megszokottnál nagyobb volumenű hálózati forgalmat reprezetnál	Nem szabad beavatkozni
Suspicious traffic	Határterület a támadások és a normál események között, valamely metrikában eltér a normál karakterisztikáktól, azonban nem jelentős mértékben	Operátori felülvizsgálat szükséges
DDoS attack	Jelentős mértékben eltér a jóindulatú felhasználók tevékenységétől, egyértelműen rosszindulatú forgalom	Egyértelmű beavatkozás és tiltás szükséges

1. táblázat. A különböző rekordtípusok rövid összehasonlítása

A második tábla tartalma az első táblában látható rekordok, mint esemény vektorok elemeit tartalmazza egyesével. Ennek egy rövid részletét mutatja be a 12. ábra, ahol a 10001-es eseményrekord azonosítótól a 10004-es rekordig láthatjuk a különböző elemek listáját. A különösen fontos adatok fejlécét itt is világoskék színnel jelölöm. Itt megfigyelhető az is, hogy míg egyes események csupán egy-egy elemből állnak, addig számos olyan eseményt is találhatunk, melynek elemszáma ennél jóval nagyobb.

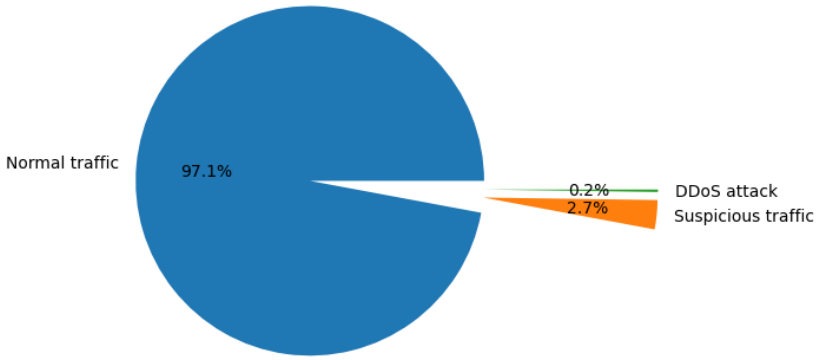
Attack ID	Detect count	Card	Victim IP	Port number	Attack code	Significant flag	Packet speed	Data speed	Avg packet len	Source IP count	Time
10001	1	sga10gq0		0	High volume	0	25200	33	1396	1	2022-03-31T07:06:39
10002	1	sga10gq0		0	High volume	0	36200	44	1278	1	2022-03-31T07:06:45
10003	1	sga10gq0		49667	High volume	0	24300	29	1287	1	2022-03-31T07:06:48
10004	1	sga10gq0		0	High volume	0	24000	29	1296	1	2022-03-31T07:06:48
10004	2	sga10gq0		0	High volume	0	25600	31	1277	2	2022-03-31T07:07:06
10004	3	sga10gq0		53186	High volume	0	32300	39	1296	1	2022-03-31T07:08:11
10004	4	sga10gq0		53186	High volume	0	33500	41	1296	1	2022-03-31T07:08:37
10004	5	sga10gq0		59257	High volume	0	33200	44	1388	1	2022-03-31T07:08:44
10004	6	sga10gq0		0	High volume	0	25200	33	1403	1	2022-03-31T07:09:18
10004	7	sga10gq0		0	High volume	0	38600	47	1296	1	2022-03-31T07:10:04
10004	8	sga10gq0		53186	High volume	0	32000	39	1274	2	2022-03-31T07:10:22
10004	9	sga10gq0		0	High volume	0	28200	34	1296	1	2022-03-31T07:10:58
10004	10	sga10gq0		62977	High volume	0	26800	35	1403	1	2022-03-31T07:11:10
10004	11	sga10gq0		53186	High volume	0	34500	42	1277	1	2022-03-31T07:11:20
10004	12	sga10gq0		0	High volume	0	36200	48	1403	1	2022-03-31T07:11:39
10004	13	sga10gq0		53186	High volume	0	24600	30	1284	2	2022-03-31T07:11:40
10004	14	sga10gq0		49348	High volume	0	33300	44	1397	1	2022-03-31T07:11:59
10004	15	sga10gq0		0	High volume	0	32200	43	1403	1	2022-03-31T07:12:17
10004	16	sga10gq0		62977	High volume	0	32200	42	1373	1	2022-03-31T07:12:17
10004	17	sga10gq0		0	High volume	0	29100	38	1403	1	2022-03-31T07:12:23

12. ábra. Egy rövid részlet az események táblájából

Amennyiben elkezdjük megvizsgálni a táblázatokat, észrevehetjük, hogy az első táblázat eseményeinek túlnyomó többsége több, gyakran akár több száz elemből is

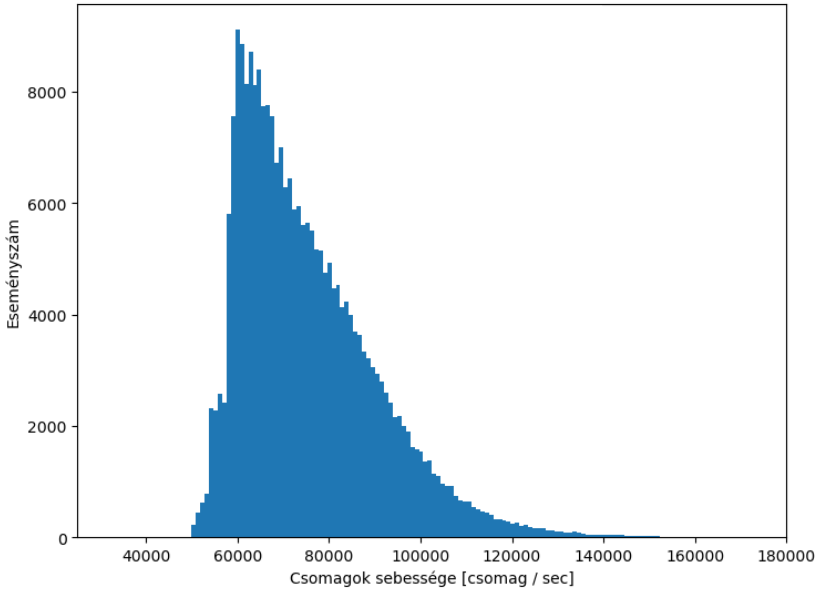
állhatnak. Ezekben az elemi rekordokban mért számadatokból származtathatók az aggregált táblában látható adatok. Emellett az első tábla "Attack code" mezőjének értéke is innen képződik úgy, hogy a legsúlyosabb vektorkomponens kódját vesszük figyelembe. Az események elemeit tartalmazó tábla azért is lesz igen hasznos nekünk, mert ennek segítségével szükség esetén további metrikákat is be tudunk vezetni, ha a modell pontosságát tovább szeretnénk növelni. Ahogyan arra már utaltam, az adatbázis mérete egyébként igen nagy, amennyiben csak egyetlen hónap adatait vizsgáljuk, akkor is nagyságrendileg 250-300 ezer eseményrekordot látunk, amikhez a másik táblában látható komponensek száma ennek többszöröse. Ebből belátható, hogy a teljes adatbázis feldolgozása igen nagy számítási kapacitást igényelne, ennek hiányában azonban én a mélytanulási viszgálatok többségét egy véletlenszerűen kiválasztott hónap adatbázisán végeztem. Az adatbázis teljeskörű ismertetése gyakorlatilag egy teljes külön dolgozatot is kitenne, amit itt nyilvánvaló okok miatt nem fogok megtenni, azonban szeretném a legfontosabb tulajdonságokat bemutatni, vizuális reprezentációkkal támogatva.

Amennyiben megnézzük a tárolt adatokat, szembetűnik az adathalmaz talán legfontosabb tulajdonsága: a "Type" mezőben megjelölt címkék. Ezek számunkra rendkívül fontosak, hiszen felhasználásukkal tudjuk majd később a mesterséges intelligencia modellt felügyelt eszközökkel tanítani és tesztelni, valamint segítségükkel a klaszterezés megbízhatóságáról is képet kaphatunk. Éredekes lehet megnézni, hogy hogyan is oszlanak el az adatbázis "Type" mezőjében megjelenő különböző hálózati forgalom címkék. Ha a lentebb látható, 13. ábrára pillantunk megfigyelhetjük, hogy a rekordok túlnyomó többsége normál, jóindulatú forgalmat jelent, míg csupán a bejegyzések 0,2%-ában láthatunk DDoS támadásokat. Emellett az is látható, hogy a gyanús forgalmak is csak egy kisebb hányadát jelentik a forgalmi mintáknak. Ez a tény megerősíti azt az állítást, melyet a bevezetőben tettem: hatalmas volumenű normál adatforgalom között kell felismernünk azokat az eseményeket, melyek veszélyt jelenthetnek az általunk üzemeltetett szolgáltatásra. Később, különösen a felügyelt modellek esetén ez az asszimmetria érdekes jelenségeket fog okozni.



13. ábra. A különböző forgalomtípusok megoszlása a teljes adatbázisban

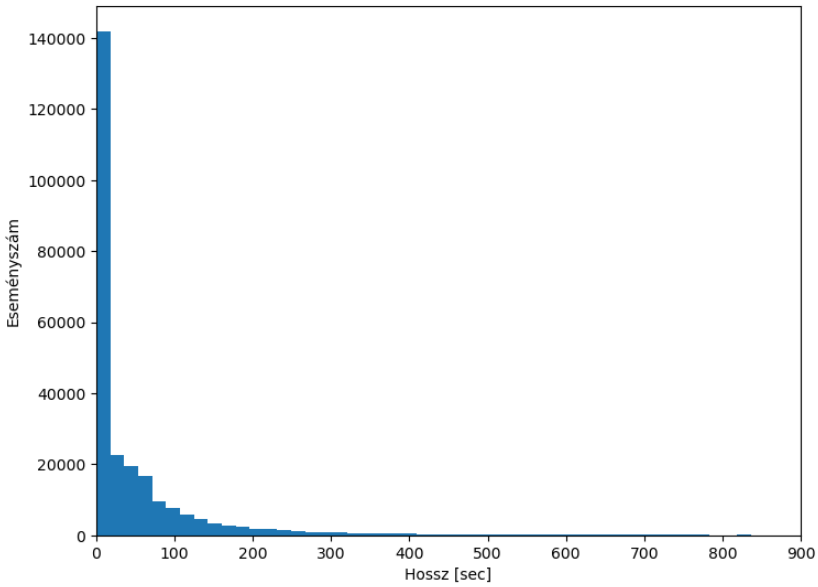
Az adatbázisban fontos elem még, amelyet érdekes lehet megvizsgálni az a csomagsebességek rekordjai. Itt érdekes módon egy normál-szerű eloszlást láthatunk, melyet a következő ábra szemléltet. Ebből is levonhatunk hasznos következtetéseket: az 14. ábrán jól látszik, hogy nem láthatóak kiugróan magas értékek, melyek egyértelműen DDoS támadásra utalnának. Ez is felfedi a terület egyik fontos nehézségét: nehezen tudjuk megkülönböztetni a normál forgalmakat a DDoS támadásoktól. Gyakran történnek olyan támadások, melyek látszólag az idő túlnyomó részében nem okoznak jelentős többlet forgalmi terhelést, valamint természetesen léteznek olyan események is, amikor a jóindulatú felhasználók terhelik túl a szolgáltatást (ezt egyébként organikus DDoS eseménynek hívjuk). Az ilyen eseményeket rendkívül nehéz helyesen detektálni. Természetesen egy igen szükséges tudás hiányzik az ábráról: nem mindegy, hogy mely forgalmi rekord melyik IP-címhez tartozik. A különböző IP-címek alatt működő különböző szolgáltatások nyilvánvaló módon más és más normál forgalmi karakterisztikával rendelkeznek. Így lehetséges az, hogy egy kis volumenű normál felhasználói adatforgalomra tervezett szolgáltatást kisebb volumenű DDoS támadással is túl lehet terhelni, míg egy nagy forgalomra tervezett rendszerben egy nagyobb támadás sem feltétlenül okoz fennakadást.



14. ábra. Az eseményekhez tartozó átlagos csomagszámok eloszlása a teljes adatbázisban

Egy másik érdekes információ lehet az, ha megvizsgáljuk azt, hogy mennyi ideig tart egy-egy esemény, hiszen az adatbázis minden egyes eseményhez rendel egy kezdő és egy végső időbélyeget. Gyakoriak az olyan támadások, melyek csupán rövid ideig jelentenek többletterhelést a hálózaton, hiszen ezzel a támadók könnyebben végezhetnek észrevétlenül tevékenységeket. Azonban ismertek olyan események is, ahol (általában kis volumenű) hosszabb ideig történő forgalmi jelenséget láthatunk. Ha a lentebb látható 15. ábrára pillantunk megállapíthatjuk, hogy az események túlnyomó többsége csupán néhány másodpercig tart, de elenyésző

számban megjelennek hosszú ideig elnyúló jelenségek is.



15. ábra. Az események időbeli hosszának eloszlása a teljes adatbázisban

A fentebb bemutatott ábrák természetesen közel sem mutatják be teljes mértékben ezt az adatbázist. A cél azonban nem is igazán ez volt, csupán az, hogy meghatározzuk a feladat szempontjából legfontosabb tulajdonságokat.

5. A létrehozott modellek és az elért eredmények bemutatása

Az eredmények részletes bemutatása előtt fontos megjegyzéseket kell tennem. Az adatbázis mérete hatalmas. Az iparban üzemeltetett detektáló eszközök által rögzített hálózati forgalomnak egy két éves időszakot lefedő kivonatát kaptam meg, amit ha idő szerinti eloszlásban megvizsgálunk láthatjuk, hogy egy hónap nagyságrendileg 250-300 ezer eseményrekorddal rendelkezik. A másik, az események elemeit tartalmazó tábla pedig értelemszerűen ennek a többszörösével rendelkezik. A teljes adatbázis feldolgozásához és egyben történő kezeléséhez igen nagy hardveres kapacitásra lenne szükség, azonban sajnos a munka során csupán korlátozott számítási erőforrásokhoz volt hozzáférésem, éppen ezért a vizsgálatok során egy leszűkített adatbázissal dolgoztam. Az adatbázisból minden alkalommal egy-egy hónapra leszűkített kivonatokat képeztem, ez a nagyságrend jelentette a felső határát annak, amit a saját eszközeim még képesek elfogadható futási időkkel kezelni. A modellek tanítása és tesztelése során ezeket használtam fel.

5.1. A Random Forest becslő

A modell létrehozásakor fontos előre eldöntenünk azt, hogy hány fával szeretnénk dolgozni, hiszen ez a paraméter jelentősen befolyásolhatja azt, hogy milyen eredményeket érhetünk el. Sajnos a szakirodalomban nem igazán lelhető fel arra vonatkozó egyértelmű irányelv, hogy mikor hány fával érdemes dolgozni. Természetesen érdemes figyelembe venni azt, hogy milyen kapacitású hardver áll a rendelkezésünkre, valamint hogy a bemenő adatok hány dimenzióval rendelkeznek. A bemenő adatokból egy alsó minimumot meg tudunk becsülni, azonban felső határt nem. Ahogy azonban a legtöbb mesterséges intelligencia alapú megoldás esetén lenni szokott, nem is feltétlenül visz minket előre az, ha minél nagyobb modellekkel dolgozunk. Ez persze akkor is igaz, ha nagy kapacitású hardverek állnak rendelkezésünkre, hiszen az elérhető pontosság ettől bizonyos komplexitás felett már teljesen függetlennek tekinthető, valójában sokkalinkább az adatoktól és a választott modelltől függ. Több okból sem érdemes a modellt túlzottan nagy komplexitásra inicializálni. Az egyik fontos ok az "Ockham borotvája" nevű, inkább filozófiai jellegű elv, mely a technológiai szektornak is egy fontos iránymutatást ad. Ennek lényege, hogy két azonos feladat megoldását ellátó eszköz vagy metódus közül mindig az egyszerűbbet érdemes választani, amennyiben a két megoldás között jelentős minőségbeli különbség (esetünkben ez a pontosság) nincsen. Tehát hiába rendelkezünk nagy kapacitású hardverekkel, amin nagyméretű modelleket futtatunk, ez igen nagy erőforrás-pazarlásként aposztrofálható, amennyiben ugyanezt a feldatot hasonlóan jó eredménnyel egy sokkal egyszerűbb eszközzel is meg lehetne oldani. A másik szintén fontos dolog, amit érdemes figyelembe venni, hogy szinte minden felügyelt mesterséges intelligencia modell hajlamos a túltanulásra, amennyiben túl sok paraméterrel rendelkezik, és ennek kivédésére nem vezetünk be semmilyen extra mechanizmust (pl. pruning) a tanító algoritmusba.

A fentebb említett tényezők figyelembevételével kezdtem neki a modellezési folyamatnak. Kezdetben kisebb számú fával dolgoztam, majd később iteratív

jelleggel változtattam a modell komplexitását, így végül egy pontossági és egy harveres igénybevételi optimumra sikerült jutni. Az első vizsgálatok során kevés (nagyságrendileg 10-20) fával dolgoztam. Ezek a tanítási idő tekintetében természetesen igen kedvezően alakultak, azonban a teljesítményük igen gyenge volt. Ennek oka a tanító algoritmus működésében keresendő. Az algoritmus minden fa létrehozásakor véletlenszerűen feature-öket választ ki a tanító halmaz mintáiból, amelyeket felhasznál a fa létrehozására. Ha viszont túl kevés fával inicializáljuk a modellünket, lesznek olyan dimenziók, amik várhatóan csupán néhány fában fognak megjelenni, mint figyelembe vett döntési küszöb. Ennek az a következménye, hogy várhatóan a végső döntési pontosság kicsi lesz. Ezen nyilvánvalóan úgy tudunk változtatni, ha növeljük a fák számát, így várhatóan minden adatbázis dimenzió kellően sok számú fában figyelembevételre kerül. Én is így tettem, a következő modellezési lépésben növeltem a fák számát egy nagyságrenddel nagyobbra, mely 100-150 fát jelentett. Ezek pontossága drasztikusan nőtt az előző vizsgálatokhoz képest, azonban a tanítást megnehezítette a modell intenzív hardverigénye. Legvégül az optimumot egy 60 fából álló modellel sikerült megtalálni. Ez a saját eszközeimen optimális futási időket eredményezett, miközben a fák növelésével a pontosság további növelését már nem lehetett elérni, viszont csökkentésükkel a becslés minősége romlott. A következőkben azt mutatom be, hogy ezzel a modellel milyen eredményeket sikerült elérni.

A Random Foresttel történő munka következő fázisában az adatbázist annak nyers formájában, a bemenetek bővítése nélkül használtam fel. Ahogy azt már említettem, egy hónapot lefedő adatot mintavételeztem, majd ennek egységesen egy 80-20%-os tanító és teszt felosztásával dolgoztam. A modell tanítása során a 2. táblázatként látható tévesztési mátrixot sikerült a teszt adatokon produkálni.

Valós \ Becsült	**DDoS támadás**	**Normál forgalom**	**Gyanús forgalom**
DDoS támadás	99	3	15
Normál forgalom	0	49209	55
Gyanús forgalom	15	116	1257

2. táblázat. A nyers adatokkal tanított random forest modell tévesztési mátrixa a 2022. májusában rögzített adatok teszthalmazán

Ha megfigyeljük a 2. táblázatot láthatjuk, hogy a támadások egy részét nem sikerült helyesen detektálni, több támadást is normál forgalomként azonosított a modell. Azonban a normál forgalmak jelentős részét sikerült jól detektálni, ezek közül egy sem kapott támadásra utaló címkét. Ezek az eredmények ígéretesnek tűnnek, azonban a feladat megkívánja azt, hogy tovább növeljük a modell pontosságát. Ezt úgy tudjuk megtenni, ha további metrikákkal bővítjük a bemeneti változók terét.

Ha megnézzük a támadás vektorok elemeit tartalmazó táblát, mely a 12. ábrán figyelhető meg, észrevehetjük, hogy sok számszerű metrikát tartalmaz. Ilyen metrikák például a csomagok méretére és az adatkapcsolat sebességére vonatkozó mérések. Ezeknek a metrikáknak csupán az egy-egy azonosító alatt futó eseményekre vett átlagát tartalmazza a támadás tábla, melyet a 11. ábrán

is láthatunk. Az átlag képzés önmagában véve igen veszteséges metódusnak mondható, hiszen egy-egy vektor elemben megjelenő mérések szórása igen nagy is lehet. Ezt az információt felismerve, kézenfekvő megoldás, ha minden egyes támadás esetén kiszámoljuk, hogy milyen szórásokkal rendelkeznek ezek a különböző változók. Ezek fontos extra információkat tartalmazhatnak a modell számára, így felhasználhatjuk őket a továbbiakban a bemenetek bővítésére. Ezen felül szintén fontos észrevenni az eseményeket tartalmazó táblában, hogy az "Attack code" mező erősen egyszerűsített adatokat tartalmaz. Az elemenket leíró táblában számos elem esetén láthatunk különböző forgalomtípusokat, protokollokat, melyek gyakran egy-egy azonosítóhoz tartozó esemény során is változnak. Jó volna, ha ezek is valamilyen formában belekerülnének a modell bemeneteiként megjelenő változókba. Ezt úgy tudjuk megtenni, hogy minden egyes vektorra kiszámoljuk a benne lévő forgalomtípusok relatív gyakoriságát. Ehhez a gyakorlatban számos új oszlopot fel kell, hogy vegyünk, egészen pontosan annyit, ahány típusú protokoll vagy forgalomtípus megjelenik az események táblájában. A kibővített adatbázis felhasználásával tanított modell pontossága kis mértékben, de tovább nőtt, ezt a 3. táblázatban látható tévesztési mátrixon is megfigyelhetjük.

Valós \ Becsült	**DDoS támadás**	**Normál forgalom**	**Gyanús forgalom**
DDoS támadás	102	3	11
Normál forgalom	0	49276	0
Gyanús forgalom	16	0	1360

3. táblázat. A kiegészített adatokkal tanított random forest modell tévesztési mátrixa a 2022. májusában rögzített adatok teszthalmazán

Ígéretes eredmény, hogy a falsrátát sikerült lentebb faragni, a normál forgalmak 100%-át sikerült helyesen besorolni. Azonban a DDoS támadások közül számos esetben így sem sikerült helyes döntést hozni. Különös probléma az a három DDoS támadás, melyet normál forgalomként értelmezett a modell. Mindezt figyelembevéve a modell pontossága jónak értékelhető, azonban nem szabad elfelejteni, hogy ez csupán egy kisebb teszthalmazon történő kiértékelés eredménye. Érdekes jelenségeket láthatunk, ha ugyanezt a már betanított modellt egy másik hónap adataival próbáljuk meg tesztelni. Ezt a 4. táblázat tévesztési mátrixa mutatja be.

Valós \ Becsült	**DDoS támadás**	**Normál forgalom**	**Gyanús forgalom**
DDoS támadás	104	28	3
Normál forgalom	6160	307090	1
Gyanús forgalom	568	8	12

4. táblázat. A kiegészített adatokkal tanított random forest modell tévesztési mátrixa a 2023. januárjában rögzített adatok teszthalmazán

Azt láthatjuk, hogy a falsráta jelentős mértékben megnőtt. Számos DDoS

támadást nem sikerült detektálni, és rengeteg normál forgalom támadásra utaló címkét kapott. Emellett a gyanús forgalmak jelentős részét is DDoS támadásként sikerült értékelni. Egyébként ezen felül más hónapokban mért adatokon is teszteltem ugyanezt a modellt, a tendencia minden esetben igen hasonló volt. Önamagában véve a modell ezekkel az eredményekkel nem nevezhető használhatatlannak jelenlegi formájában, hiszen így is sikerült a normál forgalmak és a támadások túlnyomó részét helyesen szeparálni, azonban ennél komolyabb teljesítményt várnánk el. Megállapítható az, hogy a modell általánosító képessége nem megfelelő. A modell nem képes helyesen detektálni ismeretlen forgalmi karakterisztikákat, ezek egy részét látszólag véletlenszerűen osztályozza. Az is megállapítható, hogy ez a megoldás rendkívül érzékeny arra, hogy az adatbázis mely részét használjuk fel a tanítás során. Természetesen felhasználhatnánk szélesebb időablakban rögzített adatokat a tanításra, azonban semmi nem garantálná azt, hogy azzal megbízhatóbb modellt tudnánk alkotni, valamint nem reális elvárás az, hogy kvázi végtelenbe nyúló tanító halmazt használjunk fel. Ez a problémakör erősen kapcsolódik a bevezetőben részletezett nehézséghez: a bemeneti változók olyan hatalmas teret határoznak meg, melyet szinte lehetetlen volna tanító adatpontokkal teljes mértékben lefedni. Azonban a következtetési időben megfigyelhető működésre nehezen tudunk garanciákat vállalni, ugyanis a modell korábban nem látott, teljesen ismeretlen bemeneti kombinációra várhatóan véletlenszerű választ generál.

5.2. Az MLP

A munkát azzal kezdtem, hogy itt is nyers formában, a bemeneti metrikák bővítése nélkül próbáltam meg az adatok feldolgozását. A Keras API segítségével könnyen felépíthetünk különböző modelleket, ezért én ezt a függvénykönyvtárat használtam. A modellezési munkafázis során több különböző architektúrájú megoldással is próbálkoztam. Ahogyan az már az elméleti fejezetben is bemutatásra került, az MLP legfontosabb paraméterei a rejtett rétegek száma és az azokban lévő neuronok. Több kombinációval is futtattam teszteket, a rejtett rétegek számát 1 és 2 között változtattam, míg azok elemszámát a be- és a kimeneti réteg elemszámának intervallumában módosítgattam. Már a kezdetektől fogva sok problémával szembesültem. A modell paraméterszáma a számos bemenő változó miatt mindenféleképpen igen nagy, valamint a tanító minták halmaza is jelentős méretű, ez a tanítást meglehetősen lassúvá tette. A tanítás során minden epoch több, mint 20 percig futott. Ennek ellenére a 80-20%-os tanító és teszt adathalmaz felosztás esetén, a legjobban teljesítő modell alkalmazásával egy hónapnyi adat feldolgozásakor 90%-os pontosságot sikerült elérni. Ez igen jó eredménynek mondható, azonban ha egy másik hónap adataival teszteltem, az eredmény közel sem volt ennyire szép. Ennek oka, hogy az MLP-vel is egyfajta túlilleszkedési jelenséget sikerült megfigyelni. A Random Foresthez hasonlóan, az MLP modell általánosítási képesssége is gyenge ebben a feladatban. Emellett fontos megfigyelni azt, hogy itt is érvényes a korábban megfogalmazott állítás, miszerint nem tudunk garanciát vállalni annak kapcsán, hogy hogyan fog válaszolni a rendszer az anomália-szerű, kiugró, korábban nem látott karakterisztikával rendelkező bemenő adatokra.

Érdemes lehet belegondolnunk abba, hogy mi lehet az oka ezeknek a nem túl kielégítő eredményeknek. A felügyelt tanulási modellek esetén ideális esetben érdemes olyan adathalmazzal dolgozni, melyben a különböző kimeneti címkékkel rendelkező adatpontok nagyságrendileg egyenlő számban reprezentáltak. Ez jelenleg ebben az esetben nincsen így, hiszen ahogyan azt láthattuk az adatbázis bemutatásáról szóló 4. fejezetben, az adatsorban rögzített eseményeknek csupán 2 ezreléke jelentett DDoS támadásokat, és a gyanús forgalmak is csak 3% alatti arányban találhatóak meg. Ez a terület adottsága, melyen változtatni nehezen tudnánk, hiszen pont az a célunk, hogy a vizsgálatokat valós környezetben rögzített adatokon végezük el. Viszont akkor volna lehetséges a felügyelt modellekkel (különösen az MLP-vel) történő munkában jelentős minőségbeli javulást elérni, ha ezen a masszív asszimmetrián valahogyan mégis változtatni tudnánk. Az első nyilvánvaló megoldási lehetőség lenne az, ha csak a támadásokkal egyenlő mennyiségű normál forgalmi rekorddal tanítanánk. Azonban ez sem vinne minket jelentősen előre. Fontos látni ugyanis azt, hogy általánosságban elmondható, hogy a normál forgalmak rendkívül sokfélék lehetnek, míg a DDoS támadások kevésbé. Ennek az a fő oka, hogy a normál forgalmak karakterisztikái jelentős kapcsolatban állnak az adott IP-cím alatt működő szolgáltatás típusával. A tanítás során rendkívül fontos, hogy a modell a lehető legtöbb ilyen normál forgalmi karakterisztikára rátanuljon, hiszen azokat nem szeretnénk korlátozni. Emiatt nem tehető meg, hogy bizonyos rekordokat teljesen kihagyjunk a tanító adatok halmazából, hiszen nagy valószínűséggel arra következtetési időben a háló véletlenszerű osztályozási kimenetet fog adni. Mindezek fényében elmondható, hogy a felügyelt modellekkel erős korlátokba ütköztünk, melyeket látszólag csak további támadásrekordok rögzítésével, augmentációs technikákkal, vagy inkább felügyelet nélküli modellek alkalmazásával tudunk áttörni.

5.3. A K-means és a DBSCAN klaszterezés

A klaszterezés során a legtöbbet a K-means klaszterező algoritmussal dolgoztam. Ennek oka, amit már az elméleti fejezetben is kifejtettem, hogy ez az egyik legelterjedtebb, általános célú klaszterező eszköz, mely jól skálázható nagy méretű adatbázisokra is, valamint könnyen és egyértelműen használható. A scikit-learn függvénykönyvtár renkívül felhasználóbarát implementációját tartalmazza ennek az algoritmusnak. Később azonban rátérek a DBSCAN algoritmus alkalmazására is.

Fontos kiemelni, hogy a munka ezen szakaszán már csupán a szórásokkal és relatív gyakoriságokkal kiegészített adatbázissal dolgoztam. Ezeken felül egy másik megjegyzendő dolog, hogy az adatbázisban megjelenő időbélyegek felhasználása sok kérdést vet fel. Ezek önmagukban véve ebben az esetben nem volnának túlzottan informatívak a klaszterezés számára, sőt, valószínűleg az eredményeket rossz irányba vinnék el. Emiatt bevezettem egy új metrikát, mely egyszerűen csupán a vektorokként megjelenő események kezdő és végpontja között eltelt időt tartják számon. A vizsgálatokat azzal kezdtem, hogy a K-means modellt 3 klaszter azonosítására inicializáltam, majd figyeltem, hogy a bemeneti mintahalmazon az egyes címkével rendelkező események hányas címkét kaptak a modelltől. Ezt a lentebb látható 5. táblázatban figyelhetjük meg.

Címke \ Típus	DDoS támadás	Normál forgalom	Gyanús forgalom
0	21	164737	281
1	0	0	9
2	17	7265	22

5. táblázat. 3 klaszter azonosítása K-means klaszterezéssel

Ha megnézzük a 5. táblázatot látható, hogy a K-means klaszterezés nem volt képes helyesen csoportokba sorolni a különböző eseménytípusokat. A DDoS események a normál forgalmi eseményekkel azonos címkéket kaptak. Ha jobban belegondolunk, erre a jelenségre találhatunk logikus magyarázatot. Az adatbázis többszáz különböző IP-cím mögött lévő szolgáltatás hálózati kommunikációs mintáit tartalmazza. Ezek az IP-címek számos különböző típusú szolgáltatást takarnak, így ezek normál hálózati forgalmat jelentő karakterisztikái is jelentős különbségekkel bírhatnak. Például jelentősen más egy levelezési szolgáltatást ellátó szerver és egy multimédia tartalom szolgáltató szerverének a normál forgalmi mintázata. Nem elvárható az a K-means modelltől, hogy ezek között különbségeket ismerjen fel, majd ezek alapján ezeket a jelentősen különböző normál forgalmú mintákat helyesen osztályozza.

Azonban érdekes azt megvizsgálni, hogy milyen eredményeket láthatunk akkor, ha nem ragaszkodunk a három csoportra történő bontásra. A K-means klaszterező algoritmus paramétereként tetszőleges számú klasztert megadhatunk, az algoritmus el fogja végezni az adott klaszterszámba történő besorolást. Több kíséretet is végeztem, ahol lépésről lépésre növeltem a klaszterek számát. Ennek során nem minden eredmény volt külön-külön túlzottan informatív, ezért csupán példákat mutatok be, azonban érdekes tendenciákat láthatunk. A 6. táblázaton látható, hogy hogyan sikerült 12-es klaszterszám esetén az egyes forgalomtípusok osztályzása.

Címke \ Típus	DDoS támadás	Normál forgalom	Gyanús forgalom
0	1	45748	1
1	1	46721	0
2	21	0	165
3	0	0	9
4	0	44189	0
5	0	963	6
6	1	538	0
7	10	296	9
8	0	0	86
9	0	6328	1
10	3	27219	29
11	1	0	6

6. táblázat. 12 klaszter azonosítása K-means modellel

Jól megfigyelhető az, hogy az egyes típusok címkéi elkezdtek elválni egymástól. A mintában szereplő DDoS támadások jelentős része például 2-es címkét kapott,

melyet egyetlen normál forgalmi esemény sem. Ezen felül a 7-es címkében is számos támadás van, melytől a normál forgalmak túlnyomó többségét szeparálni tudtuk. Ha tovább növeljük a klaszterek számát, ezt a tendenciát mégjobban megfigyelhetjük. A 7. táblázaton azt láthatjuk, hogy hogyan teljesített a modell, ha 18 klaszter azonosítását kértük tőle.

Címke \ Típus	**DDoS támadás**	**Normál forgalom**	**Gyanús forgalom**
0	1	11268	6
1	1	423	0
2	0	33263	0
3	0	618	0
4	0	0	1
5	0	0	86
6	2	12627	19
7	0	0	8
8	1	36146	0
9	1	0	6
10	21	0	165
11	0	38087	0
12	0	1697	0
13	0	4235	1
14	1	17013	5
15	0	961	6
16	10	190	9
17	0	15474	0

7. táblázat. 18 klaszter azonosítása K-means modellel

Itt már az is megfigyelhető, ahogyan a normál forgalmak egyes csoportjai elválnak egymástól, vélhetően a korábban említett karakterisztikákbeli kölünbségek miatt. A fentebb látott táblázatok eredményeiből levonhatunk sok következtetést. Első és legfontosabb, hogy megállapítható, hogy a K-means klaszterezés pontosságában elértünk valamiféle korlátot. Ezzel az algoritmustól nem igazán tudnánk nagyobb pontosságot elvárni ennek az adatbázisnak a használatával. Sajnos az sem könnyítő tényező, hogy nem tudunk optimális klaszterszámot meghatározni. Emellett implementációs szempontból érdekes kérdést vet fel az, hogy hogyan kezeljünk ennyi különböző klasztert, hogyan aggregáljuk a különböző címkét kapó, ámde egyforma típusú forgalmakat. Erre akár valamiféle hierarchikus jellegű megoldás használata is elképzelhető. Azonban az összességében elmondható, hogy a klaszterezés még így is megbízhatóbb és konzisztensebb eredményeket produkál a felügyelten működő modellekhez képest.

Egy másik, általam használt klaszterező megoldás a DBSCAN volt. Ennek a megoldásnak a mély algoritmikus működése jelentősen más, mint a K-meansnek, azonban alapvetően az algoritmus itt is a különböző adatpontok közötti távolság alapján hoz döntést. Nagy előnyként értékelhető, hogy a DBSCAN algoritmus

nem várja el a felhasználótól, hogy a klaszterek számát előre megmondja, ennek optimális kialakítására önállóan képes. Ennek következtében tudunk anomáliákat is azonosítani, ugyanis azok, amik egyetlen klaszter közelében sincsenek, anomáliaként értelmezett pontok lesznek. A modell legfontosabb hiperparaméterei is ezt a kérdéskört befolyásolják: meg kell adnunk, hogy minimum hány közel elhelyezkedő adatponttól tekintünk egy csoportot különálló klaszternek (*min_samples* paraméter), valamint befolyásoljuk, hogy milyen távolság az, amely után már egy-egy pontot kiugró adatként értelmezünk (*eps* paraméter). Az algoritmus skálázhatósági szempontból igen jó, azonban fontos, hogy nem olyan tökéletes, mint a K-means. Ennek következményeként a vizsgálatot egy kisebb, véletlenszerűen mintavételezett adatbázison végeztem, ugyanis a saját személyes eszközeim korlátjai ezt tették lehetővé. Ez az adatbázis egy hónap adatainak a véletlenszerűen mintavételezett 15%-át jelentette. Az eredmények érdekesen alakultak. A modellt természetesen több különböző paraméter kombinációjával is teszteltem, azonban a főbb tendenciákban nem lehetett nagy különbségeket látni. A 8. táblázatban láthatjuk az egyik teszt eredményét.

Címke \ Típus	**DDoS támadás**	**Normál forgalom**	**Gyanús forgalom**
anomália	11	320	231
0	1	33306	4
1	0	3278	3
2	76	0	725
3	0	78	0
4	0	0	42

8. táblázat. A DBSCAN modell alkalmazása a 2022. májusában rögzített adatok egy részhalmazán, *eps*=2.0 és *min_samples*=10 paraméterekkel

Számos tanulság levonható a fentebb látott eredmények kapcsán. Megfigyelhető, hogy számos normál és gyanús forgalmat is sikerült a DDoS támadások egy részén túl anomáliaként azonosítani. A gyanús forgalmak esetén ez nem feltétlenül probléma, azonban a normál forgalmak esetén igen. Azonban azt is érdemes látni, hogy ez a tanító halmazban megtalálható normál forgalmaknak csupán egy kis százalékát jelenti. Ezt a fals negatív rátát azonban mindenképpen szükséges volna tovább csökkenteni. Emellett azt is láthatjuk, hogy a DDoS forgalmak jelentős részét anomáliaként, míg jelentős részét egyetlen klaszterként sikerült azonosítani. Ez kimondottan ígéretes eredményként értékelhető, ugyanis pontosan ez volt a célunk: a DDoS események szeparálása a jóindulatú eseményektől. A modell más paraméterekkel történő alkalmazása szintén igen hasonló eredményeket mutatott, így ezeket most részletesen nem mutatom be. Összességében a DBSCAN teljesítménye is ígéretesként értékelhető. Látható, hogy a szeparáció helyes irányban történik meg, azonban a pontosság növelése további vizsgálatok szükségességét vetíti előre. Emellett érdekes megoldandó feladatkör az, hogy hogyan tudnánk még több tanító mintát felhasználni a munka során, valamint érdekes viszgálati kérdéskör volna az, hogy hány mintával volna szükséges tanítanunk a lehető legeredményesebb szeparáció elérése érdekében.

6. Összegzés, további vizsgálati lehetőségek

A fentebb ismertetett eredmények fényében az elvégzett munka sikeresnek mondható, el tudtam érni az előzetesen meghatározott célokat. Ezen felül messzemenő következtetéseket is megállapíthatunk. Kimutattam, hogy mind a felügyelt, mind a felügyelet nélküli megoldásokkal valamiféle korlátot sikerült elérni. A vizsgált modellek támadásfelismerő képessége korlátozott, a falsráta bizonyos határ alá történő csökkentése ilyen formában nem lehetséges. Azonban ez nem azt jelenti, hogy a mesterséges intelligencia által történő DDoS detekció reménytelen próbálkozás volna, sőt. Az eredmények szépnek értékelhetők, különösképpen amiatt, mert mindezt valós, ipari környezetben rögzített, aktuális adatokon sikerült elérni úgy, hogy valójában közel sem voltunk minden információnak a birtokában. Egy valós környezetben ennél több információ áll rendelkezésre, ezek tovább növelhetik a pontosságot. Emellett azt is fontos kihangsúlyozni, hogy valós ipari környezetben egy ilyen megoldás csupán egyfajta kiegészítő védelmi mechanizmusként működne. Egészen egyszerűen azért, mert számos támadás hagyományos módszerekkel is felismerhető, sőt, általában ezek jóval hatékonyabb megoldást is jelentenek. A mesterséges intelligencia legfőbb feladata az volna, hogy a hagyományos eszközökből (scrubber, tűzfal, IDS/IPS) álló védelmi rendszer által nem detektálható, ismeretlen jellegű támadások felismerésében tovább segítse az adatközpontok biztonságát, támogassa az operátori döntéshozatalt. Az az egyértelmű irány azonban kimondható, hogy a felügyelet nélküli modellek jelentősen jobb eredményeket produkáltak felügyelt társaikhoz képest, így valószínűleg a jövőben is ezekkel érdemesebb próbálkozni. Ennek egészen egyszerűen az az oka, hogy rendkívül nehéz volna egy olyan tanító mintahalmazt képezni, amely reprezentatívnak tekinthető minden szempontból, és ami széleskörben lefedi a probléma értelmezési tartományát. Természetesen egy ipari környezetben jóval komolyabb hardver eszközök is elérhetőek, így a nagy adathalmazt alkalmazó tanítás során felmerülő nehézségek ilyen környezetben nem feltétlenül lépnek fel. Azonban mindezek ellenére nehezen kerülhető meg az a probléma, hogy a következtetési időben történő működésre igen nehezen tudnánk garanciákat vállalni a felügyelt modellek esetében. Viszont mindezek fényében sem állítható, hogy a felügyelt algoritmusok használatát teljsesen el kellene vetni. Ebben a feladatban az is drasztikusan nehezítette a felügyelt modellek dolgát, hogy nagyságrendjét tekintve rendkívül kevés DDoS támadást jelentő minta állt a rendelkezésünkre. Ahogyan azt láttuk az adatbázis feltáró vizsgálatát tartalmazó 4. fejezetben, a rendelkezésre álló adatoknak csupán 2 ezreléke jelentett DDoS támadást. Fontos azonban azt látni, hogy ez a nehézség a terület sajátjának mondható. A vizsgált adatok valós ipari környezetből származnak, így teljes mértékben reprezentatívnak tekinthetők az általános adatközponti halózati forgalom karakterisztikáit tekintve. Ideális esetben egy felügyelt modell tanítása során érdemes nagyságrendileg azonos mennyiségű egy osztályba tartozó mintával dolgozni, azonban itt ez nem volt lehetséges. Érdekes további vizsgálatokra ad lehetőséget az, ha megvizsgálnánk, hogy augmentációs módszerekkel hogyan tudnánk ezen az adatbázis asszimmetrián változtatni. Ennek segítségével fel tudnánk dúsítani az adatbázisunkat, további támadásmintákat tudnánk generálni. Természetesen azonban mindez nehezen

tudna módósítani azon a nehézségen, hogy ismeretlen karaktersztikára a felügyelt modellek nem determinisztikus módon válaszolnak, azonban a pontosságon valószínűleg valamennyit javítani tudnánk.

Mindezen túl a munkámban bemutatott megoldásokon kívül rengeteg további lehetőség van még, különösen akkor, ha a rendelkezésre álló hardver eszköztár nem képez szűk keresztmetszetet. Számos egyéb, felügyelet nélküli és felügyelt algoritmus felhasználása is lehetséges volna. Az anomália detekcióban széles körben alkalmazott Isolation Forest algoritmus például igen hatékonyan tudja szeparálni a kiugró adatokat az adatbázisokból. Ezen kívül sok egyéb klaszterező algoritmus vizsgálata is lehetséges volna, nem csak a K-means használata. Ezek akár olyan megoldások is lehetnek, melyek skálázhatósági szempontból nem ennyire tökéletesek, azonban valamely más tulajdonságuk előnyösebb a K-means-hez képest. Ezen felül akár néhány felügyelt megoldással is lehetséges, hogy nagyobb pontosságot érhetnénk el. Érdekes vizsgálat volna megnézni, hogy ha a hálózat forgalmat mintáit egyfajta idősoros szekvenciaként kezelnénk, számos cella alapú neurális hálóval lehetséges volna ezek feldolgozása. Ilyenek megoldás lehetne például a szintén széleskörben alkalmazott LSTM (Long Short Term Memory) cellákból épített hálók. Persze ezek jóval komplexebb algoritmusok, természetesen így a tanításuk is jelentősen nehezebb feladat, de számos előnnyel rendelkeznek az itt bemutatott megoldásokhoz képest. Emellett felmerül az a fontos kérdés is, hogy hogyan lehetne egy már implementált, aktívan működő modellt továbbfejleszteni, tesztelni. Erre a megerősítéses tanítás eszköztárát volna szükséges alkalmazni. Azonban mindezek részletes vizsgálata már egy következő munka tárgyát képezik. A legfontosabb kérdés azonban az, hogy hogyan lehetne egy ilyen megoldást ipari körülmények között ténylegesen használni. Érdemes látni azt, hogy bár nem tűnik soknak, valós ipari környezetben néhány százalékos hibaarány sem elfogadható teljesítmény, hiszen ahogy az a bevezetőben elhangzott, egyre több és súlyosabb támadás észlelhető világszerte. Egyértelműen kimondható, hogy ezzel a detekciós aránnyal nem lehetünk elégedettek, tovább kell dolgozni azon, hogy a falsrátát a lehető legkisebb mértékűre csökkentsük.

Köszönetnyilvánítás

Elsősorban szeretnék köszönetet mondani egyetemi konzulensemnek, Dr. Orosz Péternek a témaválasztásban való segítségéért, a kimagasló segítőkészségéért, valamint a rengeteg időért, amivel a dolgozatom létrejöttét segítette. Emellett köszönettel tartozom Nagy Balázsnak, aki rendelkezésemre bocsátotta a felhasznált adatbázist, valamint Dr. Skopkó Tamásnak, aki sokat segített a programozási és egyéb technikai problémák megoldásában.

Irodalomjegyzék

[1] R. Sanjeetha, A. Shastry és H.R. Cheta. „Mitigating HTTP GET flood DDoS attack using an SDN controller". *5th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)* (2020).

[2] P. Cheskidov, K. Nikolskaia és A. Minbaleev. „Choosing the Reinforcement Learning Method for Modeling DDoS Attacks". *Internationa Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)* (2019).

[3] A. Aljuhani. „Machine Learning Approaches for Combating Distributed Denial of Service Attacks in Modern Networking Environments". *IEEE Access* (2021).

[4] S. Dong és M. Sarem. „DDoS Attack Detection Method Based on Improved KNN With the Dergee of DDoS Attack in Software-Defined Networks". *IEEE Access* (2020).

[5] H. Beitollahi, DM. Sharif és M. Fazeli. „Application Layer DDoS Attack Detection Using Cuckoo Search Algorithm-Trained Radial Basis Function". *IEEE Access* (2022).

[6] S. Yeom, C. Choi és K. Kim. „LSTM-Based Collaborative Source-Side DDoS Attack Detection". *IEEE Access* (2022).

[7] Ai Lei Tao. *How traffic scrubbing can guard against DDoS attacks.* URL: `https://www.computerweekly.com/news/252456702/How-traffic-scrubbing-can-guard-against-DDoS-attacks`. (megtekintve: 2023.10.31.)

[8] John Graham-Cumming. *No Scrubs: The Architecture That Made Unmetered Mitigation Possible.* URL: `https://blog.cloudflare.com/no-scrubs-architecture-unmetered-mitigation/`. (megtekintve: 2023.10.31.)

[9] Coursera. *3 Types of Machine Learning You Should Know.* URL: `https://www.coursera.org/articles/types-of-machine-learning`. (megtekintve: 2023.10.31.)

[10] IBM. *What is supervised learning?* URL: `https://www.ibm.com/topics/supervised-learning`. (megtekintve: 2023.10.31.)

[11] Aditya Mishra. *Metrics to Evaluate your Machine Learning Algorithm.* URL: `https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234`. (megtekintve: 2023.10.31.)

[12] Cem Dilmegani. *Top Data Augmentation Techniques.* URL: `https://research.aimultiple.com/data-augmentation-techniques/`. (megtekintve: 2023.10.31.)

[13] Amal Joby. *What Is Data Preprocessing? 4 Crucial Steps to Do It Right.* URL: `https://learn.g2.com/data-preprocessing`. (megtekintve: 2023.10.31.)

[14] AWS. *What is overfitting?* URL: `https://aws.amazon.com/what-is/overfitting/`. (megtekintve: 2023.10.31.)

[15] Jason Brownlee. *Overfitting and Underfitting With Machine Learning Algorithms*. URL: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. (megtekintve: 2023.10.31.)

[16] Souvik Paul. *Pruning in Deep Learning Model*. URL: https://medium.com/@souvik.paul01/pruning-in-deep-learning-models-1067a19acd89. (megtekintve: 2023.10.31.)

[17] Michael Wagner Philip Koopman. „Challenges in Autonomous Vehicle Testing and Validation". (). URL: https://users.ece.cmu.edu/~koopman/pubs/koopman16_sae_autonomous_validation.pdf. (megtekintve: 2023.10.31.)

[18] Micahael Weyrich Christof Ebert. „Validation of Automated and Autonomous Vehicles". *ResearchGate* (). (megtekintve: 2023.10.31.)

[19] Caroline Bento. *Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis*. URL: https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141. (megtekintve: 2023.10.31.)

[20] Pragati Baheti. *Activation Functions in Neural Networks*. URL: https://www.v7labs.com/blog/neural-networks-activation-functions. (megtekintve: 2023.10.31.)

[21] Marc Mercier. *What is a deep neural network? Learn more about deep neural networks and how deep learning works*. URL: https://botpress.com/blog/deep-neural-network. (megtekintve: 2023.10.31.)

[22] javatpoint. *Random Forest Algorithm*. URL: https://www.javatpoint.com/machine-learning-random-forest-algorithm. (megtekintve: 2023.10.31.)

[23] John F. Magee. *Decision Trees for Decision Making*. URL: https://hbr.org/1964/07/decision-trees-for-decision-making. (megtekintve: 2023.10.31.)

[24] IBM. *What is unsupervised learning?* URL: https://www.ibm.com/topics/unsupervised-learning. (megtekintve: 2023.10.31.)

[25] Avi Networks. *Anomaly Detection*. URL: https://avinetworks.com/glossary/anomaly-detection/. (megtekintve: 2023.10.31.)

[26] Harika Bonthu. *Detecting and Treating Outliers*. URL: https://www.analyticsvidhya.com/blog/2021/05/detecting-and-treating-outliers-treating-the-odd-one-out/. (megtekintve: 2023.10.31.)

[27] Scikit-learn. *Clustering*. URL: https://scikit-learn.org/stable/modules/clustering.html#clustering. (megtekintve: 2023.10.31.)

[28] Education Ecosystem. *Understanding K-means Clustering in Machine Learning*. URL: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1. (megtekintve: 2023.10.31.)

[29] Avya Gajjar. *Cluster Analysis with DBSCAN : Density-based spatial clustering of applications with noise.* URL: https://medium.com/analytics-vidhya/cluster-analysis-with-dbscan-density-based-spatial-clustering-of-applications-with-noise-6ade1ec23555. (megtekintve: 2023.10.31.)

[30] Yufeng. *Understanding OPTICS and Implementation with Python.* URL: https://towardsdatascience.com/understanding-optics-and-implementation-with-python-143572abdfb6. (megtekintve: 2023.10.31.)

Ábrák jegyzéke