MNB-BME Együttműködés 2020/2021
Digitalizáció, mesterséges intelligencia és adatkorszak Műhely

Biztonságos és a személyes adatokat védő mesterséges intelligencia megoldások

Készítette:

Gáspár Csaba

10.1.2.4. Személyazonosság, személyazonosítás online lehetőségei

Budapest, 2021

Tartalomjegyzék

1. Vezetői összefoglaló

Az adatokban rejlő erő kiaknázásának igénye miatt egyre nagyobb szükség mutatkozott olyan eljárásokra, amikben több kooperáló fél együttesen tudja feldolgozni azokat. Ezekben a szkenáriókban kiemelt szerepet kap az adatok védelme – nemcsak harmadik fél számára nem kívánják azokat megosztani, hanem gyakran az együttműködő szereplők is redukálni szeretnék a közöttük áramló információk mennyiségét. Ebbe a gondolati körbe ugyanúgy bekapcsolható a személyes adatok vagy a privacy védelme, a GPDR direktíváknak való megfelelés, mint az együttműködő pénzügyi szervezetek közötti adat- és tudáscseréje.

Tanulmányunkban elején áttekintjük, hogy a biztonságos és privacy védő mesterséges intelligencia megoldások milyen kihívásokkal szembesülnek, illetve mik azok a az alapvető építőkövek, melyekre támaszkodva biztosítani lehet az adatok védelmét.

A „nagy kép” felvázolását követően három kiemelt módszert tárgyalunk. Ezek kiemelt szerepet kaphatnak egy biztonságos adatelemzési környezet kialakításában, ugyanakkor mindhárom új, nem triviális megközelítést hoz az adatelemzés területére. A három téma rendre a következő:

(A) Differenciált adatvédelem (DA) módszere – A technika alapgondolata, hogy a megosztásra kerülő adattagokat úgy módosítsuk (tipikusan úgy terheljük meg zajjal), hogy a módosított adatok birtokában a támadóknak kevés esélyük legyen az eredeti adatpontokra vonatkozó alapadatok visszafejtésére, miközben az eredeti elemzési célt mégis jól el tudjuk érni. Tipikus példája a területnek, mikor különböző statisztikákat kívánunk publikálni miközben vigyázni szeretnénk a kis elemszámú adatpontból számolt részstatisztikák adatszivárgása ellen.

(B) Federált gépi tanulási (FGT) algoritmusok – Ezen eljárások képesek úgy felépíteni gépi tanulási modelleket, hogy a tanító adathalmazok elosztottan vannak jelen az adattulajdonosoknál. A módszerrel úgy építhetők adatbányászati modellek, hogy az adatokat birtokló intézetek, felhasználók (például adatpénztárca tulajdonosok) az adataikat nem osztják meg az algoritmus futtatása során. Hasonló módon a módszerrel akár több pénzintézet építhet közös kockázati modelleket, miközben saját ügyfeleinek adatait az üzleti titok megtartása és a GDPR rendelkezéseknek megfelelően rejtve tudja tartani az együttműködő partnerek előtt.

(C) Homomorf titkosítás – Ennek a megközelítésnek az ideája szerint úgy rejtjük el az adatokat, hogy a feldolgozó algoritmusok műveletekeiket a titkosított adaton tudják, az eredeti adatokat nem látják. Ennek megvalósításához egy speciális homomorf titkosítási módszert használ, ami magas számítási kapacitás igény mellett képes biztosítani magas fokú adatbiztonságot, de jelentősen korlátozza a megvalósítható elemzési műveleteket. A módszer még kiforratlan, az első implementációkkal nemrég léptek elő a nagy szoftvergyártó cégek.

A módszerek többségéhez megfelelő implementációk is elérhetők már, de a megfelelő védelem biztosítása érdekében gyakran több módszer kombinálására van szükség. Összességében elmondható, hogy a privacy és az adatok védelme az adatelemzés területén komoly technológiai és szervezési kompetenciák felvonultatásával ma már jól megoldható.

2. Bevezető - Adatvédelmi módszerek áttekintése

Napjainkban az innovatív megoldások jelentős részében kap szerepet a mesterséges intelligencia (MI). Miközben módszerei egyszerre jelentenek kimagasló lehetőséget, úgy alkalmazásuk számtalan kockázattal is jár. Korunk egyik kihívása megtalálni az egyensúlyt az MI felhasználása és a kockázatok és veszélyek kezelése között.

Az MI üzleti felhasználása tipikusan ott jelenik meg, ahol jelentős a digitalizáció szintje, illetve ahol jól kimutatható a rá épülő megoldások üzleti hasznosulása. Ilyen értelemben a pénzügyi szektor az MI egyik legizgalmasabb területe. Az elmúlt évtizedben robbanásszerűen terjedt el a gépi tanulási algoritmusok használata, ami sok tekintetben felgyorsította és átalakította az üzleti folyamatokat. Az adatokban rejlő érték felismerésével azonban azon kybertámadások száma is ugrásszerűen megemelkedett, ahol kiemelten fontos szerepet kapott az adatokkal való visszaélés.

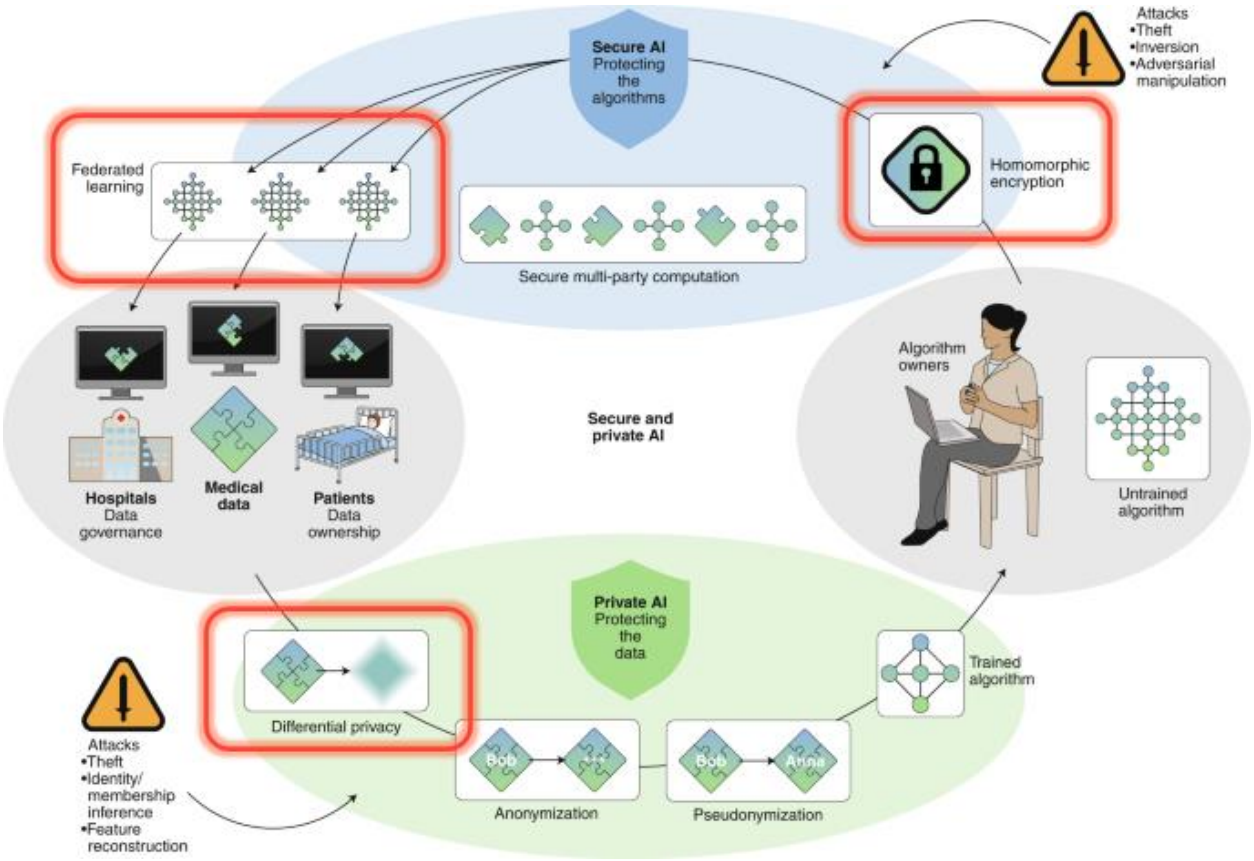
Az MI és a big data forradalmába belesimulva új, korábban nem ismert technológiai és etikai problémák merültek fel. Emiatt nem beszélhetünk adatelemzésről, sem gépi tanulási algoritmusokról anélkül, hogy ki ne térnénk annak adatbiztonsági aspektusaira.

Ebben a tanulmányban áttekintést nyújtunk a magánélet megőrzését biztosító MI jelenlegi és kialakulóban lévő technikáiról megvitatva azok előnyeit, hátrányait és technikai megvalósításait. Kitérünk az egyes módszerek robusztusságára, valamint a személyes adatok sérelmére irányuló lehetséges gyengeségeikre és támadási felületeikre.

A tanulmányunk készítése során áttekintettük a gépi tanulással kapcsolatos specifikus adatbiztonsági kérdéseket. Általánosan elmondható, hogy többek publikáltak átfogó tanulmányokat a témában, de maga a terület még nem önállósodott, a publikációk tipikusan vagy a mesterséges intelligencia vagy a gépi tanulás irányából jövő szakmai közösségekből, vagy az adatvédelemre specializálódott szakemberektől származik. A terület nem nevezhető kanonizáltnak abban az értelemben, hogy nincs tudományos konszenzus arra vonatkozóan, hogy az adott aspektussal kapcsolatban melyek a legfontosabb részproblémák.

Messze a legjobb és legátfogóbb felosztást az 1 is készítő Georgios A. Kaissis és társai adták [1]. Bár munkájuk során ők kifejezetten az egészségügyi adatokra fókuszálva fogalmazták meg a szükséges adatvédelmi elveket, mégis az ő összeállításuk a legteljesebb, legátgondoltabb, lényegében minden más általunk vizsgált áttekintő anyag elemei hozzárendelhetők ezen fogalmi keret egyes attribútumaihoz.

Az 1 ennek a tanulmánynak az összefoglaló diagramja látható. Ez négy fő területet, és azon belül módszercsaládokat definiál.



1. ábra - Az adatok, algoritmusok, szereplők és technikák közötti kapcsolatok és kölcsönhatások sematikus áttekintése a biztonságos és a személyes adatok védelmét szem előtt tartó mesterséges intelligencia területén [1].

MI megoldások esetén valamilyen adatokban rejlő összefüggéseket keresünk, általában gépi tanulási algoritmusok bevetésével. Ennek megvalósításához szükség van egy felől az elemezni kívánt adatokra, más felől pedig az analízist végző algoritmusra. Az esetek többségében ez a két információ más-más piaci szereplőknél található, így szükség van ezek valamilyen formában való megosztására. Ugyanakkor az információ csere minden esetben kockázatos, hiszen nem csak a másik félben kell maradéktalanul megbíznia az adat birtokosának, hanem az esetleges külső támadásoktól is védenie kell információt. Ez éppúgy igaz a nyers felhasználói adatokra (Adatok), mint a gépi tanulási algoritmusok modellparamétereire (Modellek). Ezen felosztás mentén megkülönböztethetünk biztonságos MI (Secure AI) és magán MI (Private AI) megoldásokat, melyek közül az első az algoritmusokat védő, míg a második a felhasználói adatok védelmét biztosító megoldásokat foglalja össze.

Az alábbi táblázat az ábrán is szereplő módszereket mutatja be egy rövid leírással és példával. Ezek közül a vastagon szedett hármat a következő fejezetekben részletesen tárgyaljuk.

Módszer neve	Leírás	Példa
Alapértelmezett biztonság (az ábrán nem szerepel)	Azokat a rendszereket, amelyeket az alapoktól kezdve a magánélet és személyes adatok védelme szem előtt tartásával terveztek és legjobb esetben nem igényelnek speciális adatkezelést.	Ahelyett, hogy utólagosan próbálnánk biztonságossá tenni az adott rendszert, az adatvédelem már a kezdetektől az implementáció része.
Federált gépi tanulás (Federated machine learning)	Az adatok egy számítógépre való összegyűjtése helyett azok helyben, a végfelhasználók saját gépén való feldolgozása, majd csak az elemzési eredmények visszagyűjtése és egyesítése.	Egy több szereplős orvosi projekt esetén a résztvevő kórházak saját rendszerükön tanítják be a saját adataikon a gépi tanulási modelleket és csak a kialakított modellt, mint eredményt küldik el az aggregáló szervezetnek.
Biztonságos többszereplős számítás (Secure multi-party computation)	Technikák és protokollok gyűjteménye, ami lehetővé teszi két vagy több fél számára, hogy az adatokat felosztva közös számításokat végezhessenek anélkül, hogy bármelyik fél betekintést nyerhetne az adatokba.	Két kórház közös betegeinek meghatározása anélkül, hogy bármelyik fél kiadná pácienseinek listáját.
Homomorf titkosítás (Homomorphic encryption)	Az adatok titkosítása oly módon, hogy azokat a gépi tanulási algoritmusok dekódolás nélkül is képesek legyenek helyesen feldolgozni, de egyénre szabott adatok ne legyenek kiolvashatók belőle.	Gépi tanulási eljárás alkalmazása titkosított adatokon azok előzetes dekódolása nélkül.
Differenciált adatvédelem (Differential privacy)	Az egyes adatpontok módosítása azok konkrét értékének elfedése érdekében a globális mintázatok megőrzése mellett. Algoritmusokra is alkalmazható.	Az adatok véletlenszerű keverése az egyének és az adatbevitelük közötti kapcsolat megszüntetése érdekében.
Anonimizálás (Anonymization)	Egy adathalmazban az azonosításra szolgáló adatok, adattagok eltávolítása.	A nevek, személyes azonosítók, illetve korhoz vagy nemhez köthető adatok eltávolítása.
Pszeudonimizálás (Pseudonymization)	Egy adathalmazban az azonosításra szolgáló adatok felülírása.	Nevek, személyes azonosítók következetes helyettesítése véletlen karakterek sorozatával.
Hardveres biztonsági megvalósítások (Hardware security implementation)	Speciális hardver architektúrák, amik garantálják a személyes adatok biztonságát.	-

Az adatelemzés során mind az elemzőt (algoritmus), mind az adatokat érhetik támadások, ezért az információt érdemes minden esetben titkosítani. A **homomorf titkosítás** lehetővé teszi, hogy az adatok dekódolás nélkül is elemezhetőek legyenek gépi tanulási algoritmusokkal. Az adatszivárgás lehetősége azonban mindig fennáll és

gyakran az adat birtokosa, legyen az szervezet vagy magánszemély, még az elemző félnek sem szeretné kiadni az adatokat, ezért sok esetben a felek törekszenek az adatmozgást minél inkább lecsökkenteni. Ebben nyújtanak segítséget a biztonságos többszereplős számítási eljárások (*secure multi-party computation*), többek közt a **federált gépi tanulás** (federated learning), ami lehetővé teszi az adatok helyben való elemzését és a különböző forrásokból származó eredmények aggregálását. A személyes adatok védelme szempontjából fontos még megemlíteni, hogy az elemzések eredményének ismeretében lehetséges visszafejteni a bemenő adathalmaz részét vagy egészét. Éppen ezért, az eredmények publikálásakor is körültekintően kell eljárni. Megoldás lehet a **differenciált adatvédelem**, ami lehetőséget biztosít a mintázatok bemutatására a valódi adatpontok bemutatása nélkül.

A dolgozat további részében az itt kiemelt három témára fektetünk hangsúlyt: differenciált adatvédelem, a federált gépi tanulás és a homomorf titkosítás módszerét tárgyaljuk részletesebben kitérve azok működési elvére és megvalósítási lehetőségeire.

3. Differenciált adatvédelem (DA)

A Differenciált adatvédelem (Differencial Privacy) a védelmet biztosító rendszer kimenő adatainak védelmét célzó adatvédelmi eljárás. Új megközelítése azt jelenti, hogy a megosztásra vagy publikálásra került kimeneti adatokból az egyes bemeneti adatpontok (például felhasználók tulajdonságai) azonosíthatóságát hivatott megakadályozni. Példaként képzeljük el, hogy valamilyen statisztikát, aggregált táblázatot szeretnénk megosztani az ügyfeleinkről, és azt szeretnénk, hogy az így publikussá vált értékekből senki se tudjon egyértelműen következtetni egy-egy felhasználó által képviselt értékre.

A módszer mélyebb megértéséhez néhány alapfogalom definiálása szükséges: akkor tekintünk egy *A* folyamatot, vagy algoritmust **differenciáltan privát**nak, ha egy bemenő adatpont kicserélése mellett a kimenete „nagyjából" változatlan marad. Ez alatt azt értjük, hogy a kimenet közel ugyanakkora valószínűséggel vesz fel egy adott értéket a két esetben.

Ha az adatokat publikáló szereplő az adatait ilyen differenciáltan privát művelettel dolgozza fel, majd publikálja, akkor nagyon kis eséllyel tud egy támadó ebből az adatból érdemi új információhoz jutni. Tegyük fel, hogy 1000 felhasználó életkorának statisztikáit osztja meg az adatgazda, de a támadó ebből 999 felhasználó adatait, és magát a statisztika készítő eljárást is ismeri. Ekkor az általa egyedül nem ismert felhasználó életkorát különböző értékkel kitölti, majd lefuttatja rá a statisztika készítő megoldást, végülösszeveti az eredményt a publikált eredménnyel. Ha a statisztika készítő megoldás differenciáltan privát, akkor a kimenet lényegében alig változik, illetve ha az egyik kimenete megegyezik a publikált értékkel nem lehet biztos benne, hogy tényleg ki tudta találni az adott felhasználóhoz tartozó értéket. Például az átlagszámítás ebben az értelemben biztosan nem differenciáltan privát művelet, mert ott bizonyosan ki lehet találni mi a bemeneti érték, de ha az átlagolás után kerekítjük az értéket, vagy ha az eredményhez egy nem zavaró méretű zajt adunk hozzá, az már előrelépés lehet a témában.

Formálisan, egy *A(D)* folyamat vagy algoritmus ε - differenciáltan privát, ha minden D_1 és D_2 egyetlen elemben eltérő adathalmazra igaz, hogy

$$e^{-\varepsilon} \cdot \mathbb{P}[A(D_2) = O] \leq \mathbb{P}[A(D_1) = O] \leq e^{\varepsilon} \cdot \mathbb{P}[A(D_2) = O],$$

bármely tetszőleges *O* kimenet esetén. Itt ε > 0 a hasonlósági tényező. Minél kisebb értéket vesz fel, a kimenetek valószínűsége annál kevésbé térhet el, hiszen, ha ε közel 0, akkor e^{ε} közel 1. Vagyis ε csökkentésével az eljárás biztonságosabbá tehető. A fenti képletben $\mathbb{P}[A(D_2) = O]$ annak a valószínűsége, hogy egy D_1 adathalmazon *A* algoritmust lefuttatva *O* kimenetet kapunk.

Ha tehát van egy adatfeldolgozó eljárásunk, ami különböző statisztikákat, műveleteket készít az adatainkon (ezek akár lehet egy gépi tanulási algoritmus, ami eredményét akarjuk publikálni), érdemes lehet megvizsgálni annak differenciáltan privát jellegét, illetve érdemes azzá tenni. Ezzel matematikailag levezethető erős biztosítékokat tudunk adni arra, hogy a publikált kimenetből esetleges támadók nem tudnak az eredeti adathalmaz adatpontjaira érdemben következtetni.

Vizsgáljuk meg a módszer néhány kiemelt előnyét:

1. **Nincs szükség a lehetséges támadások modellezésére**, a támadás típusától függetlenül védelmet biztosít a bemenő adatoknak, vagyis nem szükséges a lehetséges támadási módok feltérképezése és egyenkénti kivédése. Tehát függetlenül attól, hogy a támadó célja a célszemély beazonosítása az adathalmazban, számára korábban ismeretlen információkat szerezni róla vagy kideríteni, hogy szerepel-e benne egyáltalán, a differenciáltan privát eljárás minden információt biztonságban tart. Még akkor is, ha néhány adatpontról már tudja a támadó, hogy az adathalmaz része.

2. **Számszerűsíthető egy támadás esetén kiszivárgó maximális információ.** Tegyük fel, hogy van egy *A* mechanizmusunk, ami ε - differenciáltan privát. Ezt az algoritmust futtatjuk egy *D* adathalmazon, majd publikáljuk az *A(D)* kimeneteként kapott eredményt. Tegyünk fel, hogy egy támadó szeretné kideríteni, hogy egy konkrét személy része-e a *D* adathalmaznak.

   Ha *A* differenciáltan privát, akkor erre nem sok esélye van. Vegyük a legrosszabb esetet és feltételezzük, hogy a támadó az összes adatpontot ismeri, egyedül a célszemélyről nem tudja, hogy szerepel-e az adatbázisban. Ekkor a támadó célja, hogy kiderítse melyik volt a valós kiindulási adathalmaz: ami tartalmazza a célszemélyt (D_{in}), vagy ami nem (D_{out}).

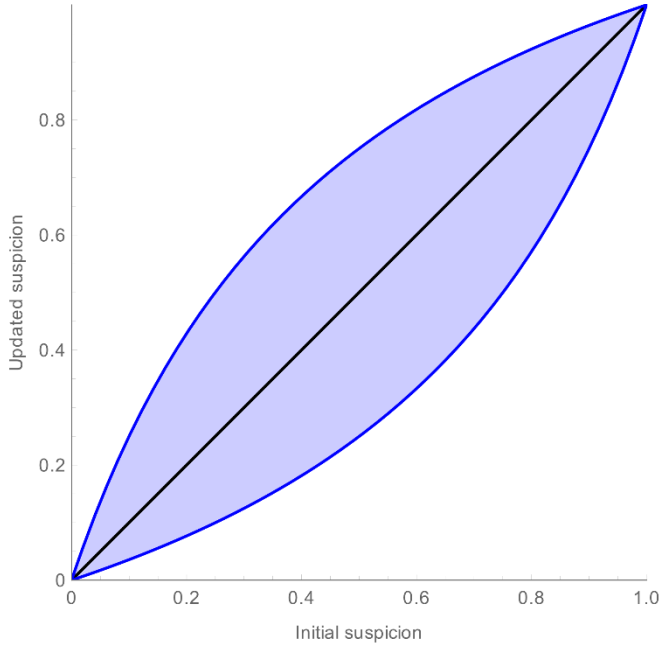
   A támadó rendelkezik egy előzetes gyanúval a célszemélyt illetően, amit egy $\mathbb{P}[D = D_{in}]$ valószínűséggel (egy 0 és 1 közötti számmal) jellemezhetünk. Azt mondjuk, hogy a gyanú 0,9, ha a támadó úgy véli a szélszemély benne volt az adathalmazban, 0,01, ha valószínűtlennek tartja és 0,5, ha tanácstalan. Ehhez hasonlóan definiálhatjuk annak a valószínűségét is, hogy a célszemély nem volt benne az adathalmazban és mivel a két esemény közül valamelyik mindenképpen bekövetkezik:

   $$\mathbb{P}[D = D_{out}] = 1 - \mathbb{P}[D = D_{in}].$$

   Az *O* eredmény publikálásával a támadó nyilvánvalóan többlet információhoz jut, de nem mindegy, hogy mennyivel tud meg többet, mint amit már eredetileg is sejtett. Az új valószínűséget formálisan $\mathbb{P}[D = D_{in}| A(D) = O]$ alakban írhatjuk fel. A Bayes tétel segítségével kifejtve az alábbi egyenlőtlenséget kapjuk:

   $$\frac{\mathbb{P}[D = D_{in}]}{e^{\varepsilon} + (1 - e^{\varepsilon}) \cdot \mathbb{P}[D = D_{in}]} \leq \mathbb{P}[D = D_{in}|A(D) = O] \leq \frac{e^{\varepsilon} \cdot \mathbb{P}[D = D_{in}]}{1 + (e^{\varepsilon} - 1) \cdot \mathbb{P}[D = D_{in}]}.$$

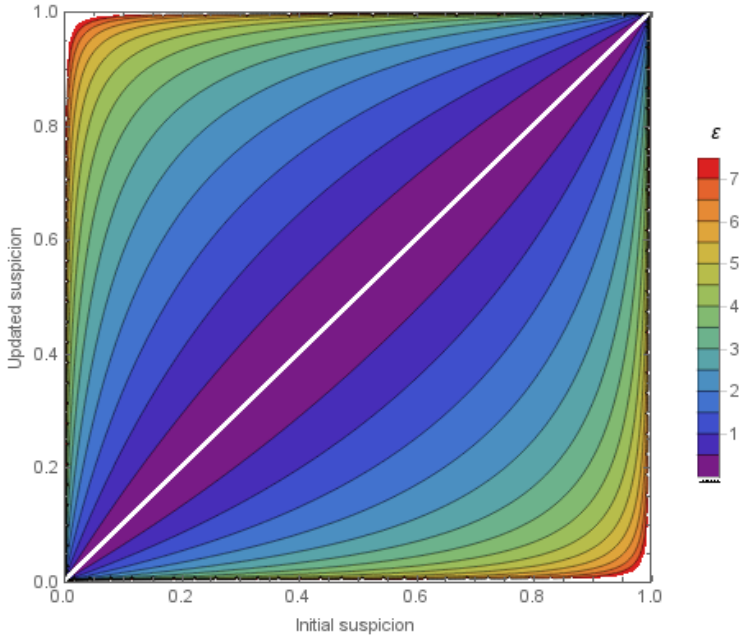
Ez tulajdonképpen a támadó gyanújának mértéke a publikált adatok ismeretében. Ebből látszik, hogy differenciált adatvédelem esetén a kimenet ismeretében érvényes valószínűség sosincs messze az eredeti gyanútól. Például ε=1.1 esetén a 2. ábrának megfelelően változik a kezdeti gyanú mértéke az *O* kimenet ismeretében.



2. ábra - Egy esetleges támadás esetén a publikált kimenetből nyerhető többlet információ valószínűsége ε=1.1 esetén [28]**.**

A fekete vonal mutatja az eredeti valószínűséget, a kék görbék pedig az alsó és felső határt, amik közé esik a többlet információ ismeretében frissített gyanú. Vagyis ha a támadó eredetileg 50 %-os bizonyossággal feltételezett valamit az adathalmazról, akkor a publikált eredmények figyelembevételével ez a bizonyosság 25-75 % között alakul.

Fontos, hogy az eredmény nagyban függ az ε hasonlósági tényezőtől. A többlet információt az eredeti sejtés valószínűségének függvényében ábrázolva különböző ε értékekre a 3. ábra látható grafikont kapjuk.



3. ábra -- Egy differenciáltan privát algoritmus kimenetéből nyerhető többlet információ az eredeti sejtés függvényében különböző hasonlósági tényezők esetén[28].

3. **A differenciált adatvédelem kompozíciós tulajdonsággal rendelkezik**. Egy adathalmazon különböző számításokat végezhetünk, aminek eredményét minden esetben közzé is tehetjük. Amennyiben minden számításunk differenciáltan privát, úgy az egyes megoldások együttese is differenciáltan privát. Vagyis a publikált adatok összességéből maximálisan kinyerhető információ mennyiség számszerűsíthető.

Tegyük fel, hogy *A* és *B* ε - differenciáltan privát algoritmusok. A kettőt kombinációját nevezzük C algoritmusnak: $C(D) = C\big(A(D), B(D)\big)$. Vagyis *C* kimenete kizárólag *A* és *B* kimenetétől függ, amit egy $O = (O_A, O_B)$ kimenet párral jellemezhetünk. Mivel az algoritmusainkat kizárólag saját véletlenszerűségeik befolyásolják, a rendszerünket felírhatjuk független valószínűségi eseményekként. Azaz

$$\begin{aligned}\mathbb{P}[C(D_1) = O] = \ & \mathbb{P}[A(D_1) = O_A] \cdot \mathbb{P}[B(D_1) = O_B] \\ & \leq e^{2\varepsilon} \cdot \mathbb{P}[A(D_2) = O_A] \cdot \mathbb{P}[B(D_2) = O_B] \\ & \leq e^{2\varepsilon} \cdot \mathbb{P}[C(D_2) = O],\end{aligned}$$

vagyis *C* 2ε - differenciáltan privát. Tehát gyengül ugyan a titkosítás mértéke, de továbbra sem teljesen feltörhető az adathalmaz. Mitöbb, pontosan számszerűsíteni tudjuk, hogy a legrosszabb szcenárió esetén kiszivárogható adatok maximális mennyiségét.

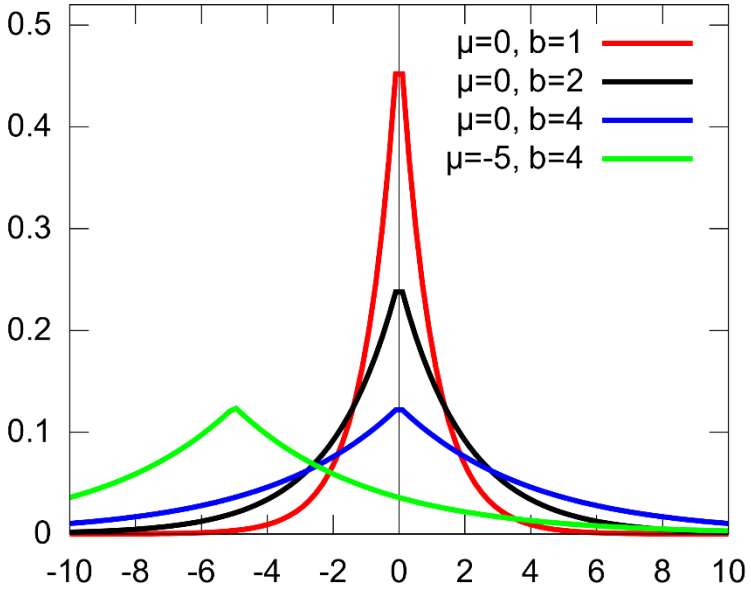
3.1. Differenciált adatvédelem a gyakorlatban

Differenciált adatvédelem esetén mindig azt feltételezzük, hogy egy esetleges támadó egy kivételével (célszemély) az összes adatpontot ismeri, és a célszemélyről próbál információt nyerni. Értelemszerűen ebben az esetben nem publikálhatjuk a valós eredményeket, hiszen abból egyértelműen adódna a célszemélyre vonatkozó információ. Viszont ha az eredményekhez zajt keverünk, akkor az adataink globális mintázata megmarad, viszont számszerűen nem követhetőek vissza az egyes adatpontok. Megfelelő paraméterezéssel tetszőleges valószínűség-eloszlás (pl. normál eloszlás, exponenciális eloszlás, geometriai eloszlás, egyenletes eloszlás stb.) bevethető a valós adatok torzítására használt zaj kikeveréséhez. A következő példákban Laplace-eloszláson keresztül mutatjuk be a differenciált adatvédelem menetét. Ez az eloszlás egyszerű paraméterezhetősége miatt jól használható, ugyanakkor népszerű a DA publikációk levezetéseiben előnyös matematikai tulajdonságai miatt.

A Laplace-eloszlás sűrűségfüggvénye

$$f(x|b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}},$$

ahol μ a lokációs paraméter, ami az eltolást adja meg és b a skálaparaméter, ami meghatározza az eloszlás „elkentségét”, vagyis hogy mennyire lapos a függvény. A 4. ábra a Laplace-eloszlás sűrűségfüggvényét mutatja meg különböző paraméterekkel.



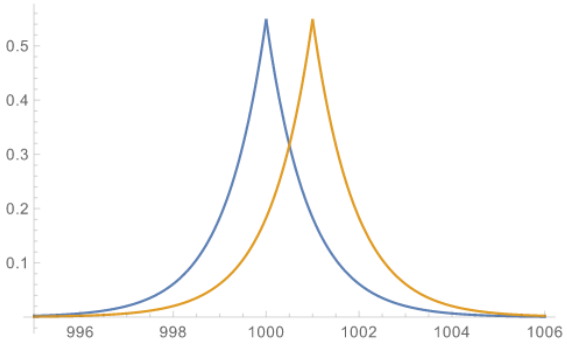
4. ábra - Laplace-eloszlás sűrűségfüggvénye különböző paraméterekkel [29].

Nézzünk meg néhány egyszerűbb statisztikai függvényhez köthető példát, illetve azt hogy ezekben az esetekben hogyan lehet az adott eljárást differenciáltan priváttá tenni.

3.1.1. Egyedi értékek megszámlálása

Tegyük fel, hogy van egy adathalmazunk, ami emberek szemszínét tartalmazza. Egy támadó ebből egy kivételével minden adatpontot ismer, és a célszemélyről szeretné megtudni, hogy zöld-e a szeme. Ha közzétesszük a zöld szeműek valós *k* számát, akkor ezt az értéket a támadó összevetheti az általa ismerttel és ebből már tudja is, hogy a célszemély szeme milyen színű. Hiszen ha a támadó által ismert személyek közt *k*-1 zöld szemű található, akkor a célszemély zöld szemű, ha viszont *k*, akkor nem az.

Ha a valós értékhez Laplace(1/ε) eloszlású zajt keverünk (ebből az eloszlásból egy véletlen értéket kiválasztunk és azt a valós értékhez adjuk), akkor a támadónak sokkal nehezebb dolga van. Az 5. ábra szemlélteti a közzétett érték eloszlását annak függvényében, hogy a valós érték *k* = 1000 (kék vonal) vagy *k* = 1001 (sárga vonal) volt-e.

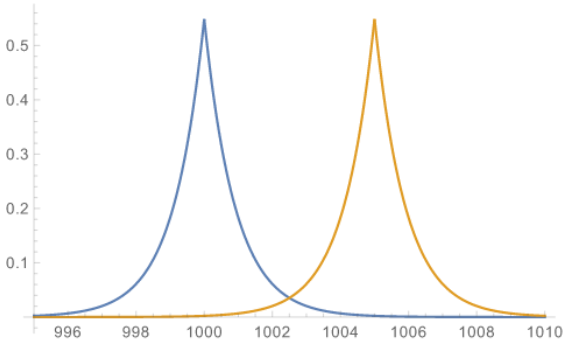


5. ábra - Laplace-eloszlás 1/ε = 1/1,1 skálaparaméterrel, k=1000 és k=1001 valós kimeneti értékek esetén [28].

Képzeljük el, hogy a valódi érték *k* = 1001, zajjal eltolva pedig 1003-at kapunk, amit közzé is teszünk. Ekkor egy esetleges támadó nehezen tudja kitalálni, hogy valójában 1000 vagy 1001 volt-e a mért szám. A két eset valószínűségének hányadosa éppen e^{ε}-t ad vissza, ami az ε - differenciáltan privát mechanizmus definíciója.

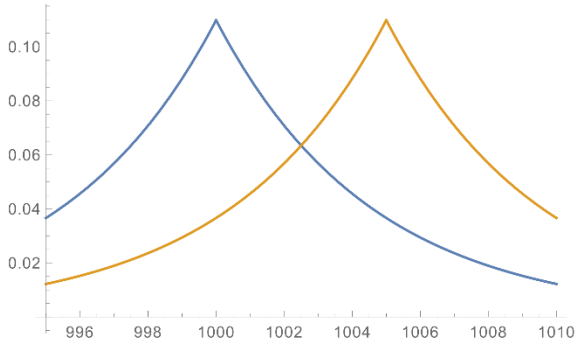
3.1.2. Eltérő értékek kezelése

Az előző példában csupán két lehetséges érték megkülönböztethetetlenségét kellett biztosítanunk, amik 1 egység távolságra voltak egymástól. Könnyen előfordulhat azonban, hogy a publikálni kívánt eredmények több értéket is felvehetnek. Ilyen esetben egy esetleges támadónak annál könnyebb dolga van, minél nagyobb az



6. ábra – Laplace-eloszlások 1/ε = 1/1,1 skálaparaméterrel, k = 1000 és k = 1005 valós kimeneti értékek esetén [28].

értékek közti eltérés. Például egy igényfelmérő űrlap esetén tegyük fel, hogy mindenki csupán egy választ küldött be, kivéve egy személyt. Ha a támadó éppen e személy válaszainak számára kíváncsi, akkor annál könnyebb dolga van, minél több válasz érkezett a célszemélytől. Ha az illető 5 választ küldött be, akkor a fenti eloszlások helyett a 6t kapjuk. Ebben az esetben a valószínűségek aránya már nem e^{ε}, hanem lényegesen nagyobb: $e^{5\varepsilon}$, vagyis a metódus nem ε - differenciáltan, hanem 5ε - differenciáltan privát. Ahhoz, hogy visszakapjuk az ε - differenciáltan privát esetet, a Laplace-eloszlás skálaparaméterét 1/ε helyett 5/ε-nak kell választanunk. Így az 7 szerinti eloszlásokat kapjuk, amik közt láthatóan lényegesen nagyobb az átfedés. Viszont ez csak akkor igaz, ha a kérdőívünkre adott maximál válaszméret 5. Vagyis minden esetben ismernünk kell az értékkészlet maximumát, vagy egy adott értékben fixálnunk kell azt, a kiugró értékeket pedig ennek mentén levágni.



7. ábra - Laplace-eloszlások 5/ε = 5/1,1 skálaparaméterrel, k = 1000 és k = 1005 valós kimeneti értékek esetén [28].

3.1.3. Több statisztikai elemzés publikálása egyszerre

Általában egynél több információt teszünk közzé egyszerre (pl. az egyes szemszínek előfordulásának a száma, egy kérdőív esetén a válaszok eloszlása), ilyenkor pedig máshogyan kell eljárnunk az adatvédelem tekintetében. Érdemes két esetet elkülöníteni aszerint, hogy az adatpontok csak egy statisztikában szerepelnek-e vagy lehetséges átfedés.

Tegyük fel, hogy egy kérdőívre adott válaszokat szeretnénk publikálni. Első lépésben a válaszadók életkor szerinti eloszlását mutatjuk meg, ahol értelemszerűen nem lehet átfedés a csoportok között, hiszen mindenki pontosan egy kategóriába sorolható. Ebben az esetben minden egyes korcsoport valós értékéhez hozzáadhatunk Laplace-zajt 1/ε skálaparaméterrel, így biztosítjuk, hogy a publikált statisztika ε - differenciáltan privát legyen. Arra kell csak figyelnünk, hogy a tört értékek kerekítsük egészre, az esetleges negatív számokat pedig 0-ra. Kis értékek esetén (például ha egy kategóriába csak 1 személy esik) nehezebb dolgunk van, mert nem biztos, hogy a zaj megfelelően elfedi a valódi értéket. Ezért érdemes definiálni egy küszöbértéket, ami alatti esetszámokat nem publikálunk.

Az űrlap egyes kérdései között viszont már ismétlődnek a szereplők, így ebben az esetben – a fent tárgyalt kompozíciós tulajdonság miatt - nem elegendő az egyes kategóriákat 1/ε paraméterű Laplace-zajjal torzítani. Ha mindent összevetve *C* darab kategóriánk van (összeszámolva minden publikálandó kérdés lehetséges válaszait), akkor ezek között kell elosztanunk a rendelkezésünkre álló ε „költségvetésünket” ahhoz, hogy a végeredmény ε – differenciáltan privát legyen. Ezt eloszthatjuk egyenletesen (minden kategóriához *C*/ε paraméterű Laplace-zajt keverünk) vagy önkényesen módosítva az egyes kategóriákra eső skálaparamétert

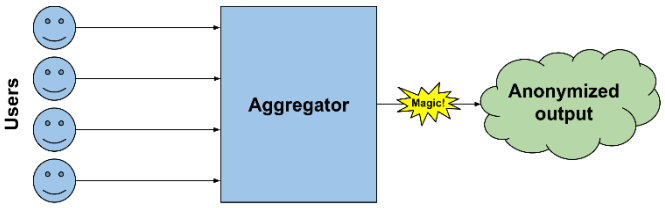
azok védelmének fontossága szerint. A lényeg, hogy az egyes kategóriák ε-jait összeadva az eredetileg definiált értéket kapjuk.

3.2. Differenciált adatvédelmi architektúrák

A korábban leírt differenciált adatvédelmi (DA) módszerekkel tehát biztosíthatjuk matematikai értelemben az adatok védelmét, de érdemes körüljárni azt is, milyen architektúrában érdemes megvalósítani ezt a folyamatot: a kérdés itt az, hogy ha a védendő felhasználók birtokolják saját adataikat (vagy a védendő felhasználó több adatpontot együtt kezel és ismer), ki és hogyan gyűjtse össze az adatokat, ki és hogyan hajtsa végre a differenciáltan privát műveleteket, hogy a végén előálljon a megfelelő kimenet, illetve ezen folyamat mentén melyik egység biztonságát kell garantálni, mi történik, ha ezt az egységet támadás éri.

3.3.1. Globális DA

A globális differenciált adatvédelmi modellben, ahogy azt a 8 szemlélteti, egy központi összesítő, más néven *aggregátor*, kezeli az adatokat. Az *adatforrások* (users), akik lehetnek individuumok vagy adatgyűjtő szervezetek is, közvetlenül az *aggregátornak* küldik el a valós (zajjal nem terhelt) adatokat. Az adatgyűjtést követően a központi rendszer kezeli az adatokat és elvégzi rajtuk a szükséges számításokat. Mielőtt az eredmények publikálásra kerülnének, az *aggregátor* differenciáltan privát metódust hajt végre az adatok védelme érdekében. Ezt az eljárást alkalmazták például a 2020-as amerikai népszámláláson [2].

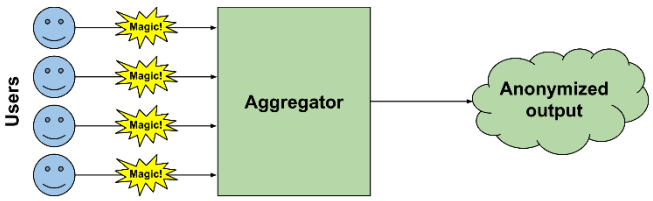


8. ábra - Globális differenciált adatvédelmi modell sematikus rajza [28].

A globális modell legnagyobb előnye a pontossága. A rendszerben egyetlen egyszer kerülnek az adatok torzításra és a nagy adatmennyiség következtében viszonylag kis zajjal is nagyfokú biztonságot érhetünk el. Hátránya viszont, hogy az egy helyen tárolt nagyszámú adat miatt a rendszer könnyen támadások célpontjává válhat. Ha pedig egy támadás sikerrel jár, akkor az összes adat kiszivároghat. Ebből adódóan az adatszolgáltatókat is nehezebb meggyőzni az *aggregátor* biztonságosságáról és így rávenni őket az adatok megosztására.

3.3.2. Lokális DA

A lokális differenciált adatvédelmi modellben szintén egy *aggregátor* gyűjti össze, dolgozza fel és végül publikálja az adatokat, viszont ebben az esetben a bemenő adatok zajjal keverve, vagyis anonimizálva érkeznek. Ebből adódóan nincs szükség az adatok további titkosítására, sem a statisztikák torzítására. Ezt az eljárást használja például az Apple az iOS billentyűzet-használtra vonatkozó adatok gyűjtéséhez [3].



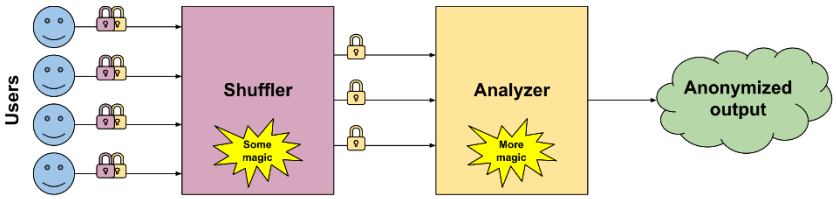
9. ábra - Lokális differenciált adatvédelem sematikus ábrája [28].

Nagy előnye ennek a modellnek, hogy – a globális modellel szemben – az *aggregátor* megbízhatósága nem feltétele a működésnek. Mivel a felhasználók saját maguk védik az adataikat, így egy rosszindulatú szervezet esetén is biztonságban tudhatják azokat. Tehát a lokális modell jól használható olyan esetekben, ahol nehéz

elnyerni a felhasználók bizalmát. Ilyen lehet például bármely törvénybe ütköző tevékenységre (pl. droghasználat, adócsalás stb.) irányuló kutatáshoz való adatgyűjtés. A séma hátránya a zajos adatokban rejlik. Mivel minden felhasználó saját maga torzítja az adatait, a teljes adathalmazra vonatkozó zaj lényegesen nagyobb. Éppen ezért, sokkal több adatpontra van szükség ahhoz, hogy az eredmények értékelhetőek legyenek. Segíthet továbbá, ha nagyobb ε értéket definiálunk.

3.3.3. ESA: Encode, Shuffle, Analyze

Az előző két modell esetén láthattuk, hogy bár vitathatatlan előnyökkel rendelkeznek, az adatbiztonság vagy az adatok zajossága miatt használatuk korlátokba ütközik. Ezen problémák megoldására született az ESA (Encode, Shuffle, Analyse – magyarul: kódolás, keverés, elemzés) architektúra, ahogy azt a 10 mutatja. Az előzőekben látott aggregátort ebben a modellben két másik egység, a *keverő* (shuffler) és az *elemző* (analyzer) váltja fel a bemenő zaj csökkentése és az adatok biztonságos tárolása érdekében. A *kódoló*, vagy felhasználó (encoder, vagy user) egység pedig a korábbi felhasználóknak és adatforrásoknak felel meg, kihangsúlyozva, hogy az információ már kódolva érkezik a *keverő* egységbe.



10. ábra - ESA differenciált adatvédelmi modell sematikus rajza [28].

Első lépésben a felhasználók (vagy adatgyűjtők) kétszeresen kódolják az adataikat, két rétegben, mielőtt azt továbbítanák a keverőnek. A keverő egy köztes szereplő, aki ebből kizárólag az első réteget tudja dekódolni. Ez azt jelenti, hogy hozzáfér az egyedi és a csoport azonosítókhoz (user ID és group ID), de a konkrét értékekhez nem. Például láthatja, hogy az adatok a testmagasságra vonatkoznak, de az egyes felhasználók magasságához nem fér hozzá.

Ezt követően a keverő összegyűjti az adathalmazban előforduló összes osztályt a csoport azonosítók felhasználásával és megszámlálja az egyes kategóriákba eső adatpontokat. Az elemszámokat összeveti egy előre definiált küszöbértékkel és csak azon csoportok adatait továbbítja az elemzőnek, melyekben elegendő adatpont szerepel. Fontos, hogy a felhasználók azonosítóit ilyenkor már teszi bele az elküldendő csomagba. Majd az elemző dekódolja az adatok második biztonsági rétegét, így hozzájut a valós adatokhoz, amiken végül elvégzi a kívánt elemzéseket és publikálja az eredményt.

A kétlépcsős titkosításnak köszönhetően különválasztásra kerül maga az adat az őt leíró metaadatoktól. A keverő hozzáfér a felhasználói azonosítókhoz, viszont az általuk közölt információhoz nem. Az elemző pedig pont fordítva: az adatokat csomagokban látja, de azt nem tudja visszakövetni, hogy melyik adatpont kitől származik. Ha a keverő szintjén megfelelően megválasztott véletlenszerűséget viszünk a folyamatba, akkor azzal biztosíthatjuk, hogy a kimenet differenciáltan privát legyen.

Az ESA modell kulcsa a különválasztott keverő és elemző egység. A kivitelezés szempontjából kényelmesebb lenne a két folyamatot egy helyen kezelni, de ezzel lényegében a globális modell egy lépéssel továbbfejlesztett

változatához jutunk, vagyis az adatszivárgás kockázata továbbra is fennáll. Tehát a teljes adatbiztonság két módon valósulhat meg:

- A keverő és az elemző egységet külön szervezetek biztosítják. A bizalom növelése érdekében célszerű, ha a keverő szerepét egy nonprofit ügynökség látja el.
- Amennyiben a két szerepet ugyanaz a szervezet látja el, a bizalmat növelheti, ha a keverés egy különös védelemmel ellátott biztonságos hardveren megy végbe, amiről a felhasználók távoli bizonyítvány (remote attestation) segítségével bizonyosodhatnak meg.

Forrás:

- Big Data UN Global Working Group: UN Handbook on Privacy-Preserving Computation Techniques 2019, http://publications.officialstatistics.org/handbooks/privacy-preserving-techniques-handbook/UN Handbook for Privacy-Preserving Techniques.pdf
- D. Desfontaines, "Ted is writing things On privacy, research, and privacy research.," 2018. https://desfontain.es/privacy/differential-privacy-awesomeness.html.

3.3. További megfontolások

Az differenciált adatvédelmet biztosító módszer nyilvánosságra hozása

A módszer egyik sajátossága, hogy a támadó ismerheti a számítási folyamatot végző módszert, sőt a védelemnek valamilyen szinten része, hogy ezt kötelező a tudomására hozni. Képzeljük el azt az esetet, hogy az egyszerű statisztikák publikálása előtt differenciális adatvédelmi módszerként különböző zajokat adtunk hozzá az adatokhoz, de ezt nem fedtük fel a publikusan, például azt állítjuk, hogy a publikált adatok a valós adatok.

Ebben az esetben egy adatvédelmi érzékenységgel rendelkező állampolgár, ügyfél méltán hiheti azt, hogy az adati feltörhetők, úgy érzi az adatait elárulták, jelentősen csökken az adatpublikáló iránti bizalma. Sőt egyes esetekben a problémára való figyelem felkeltése céljából maga is támadóvá válik, könnyűszerrel visszafejti az adatokat (a hasonló példában itt is egyetlen felhasználó adatát nem ismeri, de a többit igen), és bár várhatóan rossz eredmények birtokában van, a publikus térben hangoztatni tudja, hogy a rendszer feltörhető, ő is hozzájutott az egyik felhasználó adott adatpontjához. Egy ilyen szituációból csak rosszul tud kijönni az adatokat publikáló szervezet.

A támadó rendelkezésére álló háttéradatok reális mértéke

A leírt példákban azt feltételeztük, hogy a támadó a teljes adathalmazt egyetlen felhasználó kivételével ismeri. Ez nyilván nagy méretű adathalmazok esetén nem reális, illetve ha ártó szándékai vannak az adatokkal, akkor valószínűleg a nagy adathalmazból fog találni magának megfelelő célpontot. A differenciált adatvédelem valójában egy másik szituációt vesz górcső alá: mikor egy nagy sokaság adott részcsoportjáról beszélünk, mely részcsoport már lehet kellően kicsit ahhoz, hogy a támadónak megfelelő mennyiségű adat áll rendelkezésére.

Ha a januári budapesti lakáseladások négyzetméterre vetített árait vizsgáljuk, és publikáljuk azok átlagát, akkor valószínűleg nem lesz olyan szereplő, aki ismerni fogja egy kivételével az összes tranzakciót. De ha ugyanezt az adatot kerületi vagy irányítószám szerinti felbontásban is közöljük, ott már állhat elő olyan szituáció, ahol az

elemszámok kis mérete, vagy egy-egy ingatlanközvetítő a saját belső adatbázisa ismeretében már tud következtetni korábban nem ismert adásvételek árszintjére.

Tehát ebből a szempontból a differenciált adatvédelem egy elég szigorú feltételrendszer mentén igyekszik védeni az adatokat, aminél egy igen ritka, erős támadás esetére készül fel, azaz a módszer a gyengébb támadási módszerek esetén a határhelyzetkez képest jelentősen jobb védelmet biztosít.

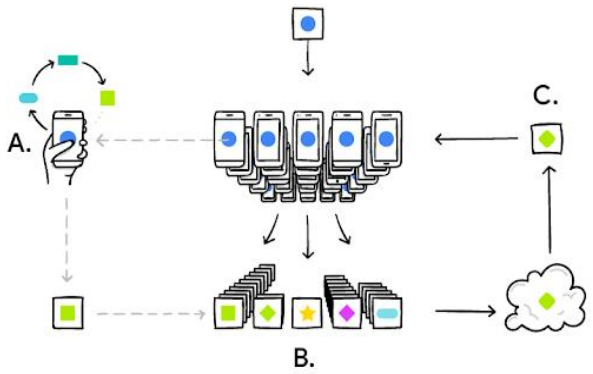
Gépi tanulási algoritmusok felépített modelljei és a differenciált adatvédelem

Az általunk áttekintett irodalom alapján egyre több gépi tanulási algoritmusnak születik differenciáltan privát implementációja. Az egyszerűbb statisztikai modelleken túl regressziós problémák [4], osztályozási feladatok [5]-[6], klaszterezés [7],[8],[9] és dimenzió redukció [10] megoldására is vannak már differenciáltan privát algoritmusok, amik az adott algoritmusnak megfelelő módon kevernek zajt az eredményekhez. Ezen túl a *deep learninghez* használt neurális hálózatoknak is van már differenciáltan privát változata [11]. Az elméleti megfontolásokon túl, 2021. januárjától már elérhető az IBM Differential Privacy Library python csomag, ami a neurális hálózat kivételével az említett gépi tanulási problémák mindegyikére kínál legalább egy modellt, ami eleget tesz a differenciált adatvédelemnek [12], [13].

4. Federált gépi tanulás (FGT)

A federált gépi tanulás (FGT) lehetővé teszi egy modell decentralizáltan tárolt adatokkal való betanítását. Ez a gyakorlatban azt jelenti, hogy egy központi adatgyűjtő és -elemző rendszer helyett az elemezni kívánt adatok nem mozdulnak, hanem keletkezési helyükön kerülnek feldolgozásra. Erre az architektúrális elrendezésre tekinthetünk gráfként, éppen ezért az adatforrásokat szokás (lokális) csúcsoknak nevezni. A gyakorlatban ezek bármilyen adatgyűjtésre, adatfeldolgozásra és adatátvitelre alkalmaz ezközök lehetnek, például okostelefon vagy IoT (*Internet of Things* – dolgok internetje) eszközök.

A federált gépi tanulási folyamatot a 11 szemlélteti. A központilag inicializált gépi tanulási modell paramétereit minden egyes résztvevő (A) megkapja és saját adatainak felhasználásával lokálisan frissíti, tanítja a modellt. Az eredményt visszaküldi a központi szerverre (B), ahol az egyes modellek aggregálásával létrejön egy új, globális modell (C). Ezt tekintjük egy iterációs lépésnek. A kellő pontosság elérése érdekében érdemes több iterációs lépést végezni. Továbbá az adatok a felhasználóknál való folyamatos termelődése is szükségessé teszi a modell



11. ábra - Federált gépi tanulás elvi működése [14].

folyamatos újra tanítását. Ezzel az architektúrával az adattárolás és a modellépítés problémája különválasztható, aminek köszönhetően sokkal nagyobb biztonságban tudhatjuk az adatokat, hiszen azt követlenül nem osztottuk meg senkivel [14], [15].

A rendszer egyik nagy előnye, hogy az adatok végig lokálisak maradnak, nem kerülnek központi tárolásra. Vagyis a felhasználóknak nem kell attól tartaniuk, hogy adataik avatatlan kezekbe kerülnek, vagy olyan más, általuk nem támogatott modell vagy statisztikai kimutatás készítésére használták fel azt.

Ugyanakkor élvezhetik egy globális modell előnyeit, a gépi tanulási algoritmusok pontossága jobb lehet, mintha csak a lokális adatokon történt volna azok hangolása, ennek megfelelően várhatóan javul a felhasználói élmény. A tanulási folyamat csúcsoknak való kiszervezése által a rendszer számítási kapacitása is nagyobb, hiszen a tanítás párhuzamosan történik az egyes perifériákon. Így a központi szerver szemszögéből nézve a federált gépi tanulás számítás- vagy energiaigénye lényegesen alacsonyabb más eljárásokhoz képest. Ez közvetlenül adódik a csúcsokon végzett tanításból, hiszen ekkor a felhasználók eszközét terheli a számítási kapacitás- illetve energiaigény. Nyilvánvalóan a felhasználók nézőpontjából szemlélve ez hátránynak minősül, ami akár az adott rendszerben való részvételt, vagyis az adatszolgáltatást is meghiúsíthatja.

A párhuzamosított folyamatok gyorsabb előrejelzést tesznek lehetővé. A központi szerver kizárólag aggregálást és az adatok továbbítását végzi, ezért lényeges számítási kapacitást spórol meg a hagyományos gépi tanulási eljárásokhoz képest. A csúcsok függetlenségéből adódóan akár arra is van lehetőség, hogy, a lassabb eszközök válaszának megvárása nélkül, a gyorsabb eszközök frissítése hamarabb megtörténjen.

A továbbiakban a leírásunk főleg az elosztott felhasználókat a legtöbb esetben mobiltelefon felhasználóként fogjuk tekinteni, de az itt tárgyalt kihívások és lehetőségek nagyon hasonló módon fogalmazhatók meg, ha más számítási környezetben képzeljük el az együttműködést. A fejezet végén külön kitérünk a módszerek pénzpiaci alkalmazhatóságára.

Fontos azonban szem előtt tartanunk, hogy a perifériáknak képesnek kell lenniük az adatgyűjtésen túl adatelemzésre is. A mai okostelefonok szinte már kivétel nélkül alkalmasak bonyolult gépi tanulási modellek futtatására, azonban még számos korábbi gyártmányú készülék van forgalomban, amik nem feltétlenül rendelkeznek a szükséges paraméterekkel. Amennyiben nem a végfelhasználók készülékein tanítjuk a gépi tanulási algoritmusokat, hanem egymástól független adatbázisokon, abban az esetben is meg kell győződnünk róla, hogy a tárolási forma lehetőséget biztosít az algoritmusunk futtatására.

Amennyiben a rendszer csúcsait a végfelhasználók jelentik, akkor számolnunk kell az adatforrások folyamatos fluktuációjával. Bármikor előfordulhat, hogy egy adott eszköz kikerül a hálózatból valamilyen kommunikációs vagy energiaellátási probléma miatt. Továbbá a felhasználók bármikor megváltoztathatják korábbi döntésüket és megtilthatják az adatokhoz való hozzáférést. Ezzel egyidőben pedig folyamatosan csatlakozhatnak újabb kliensek a rendszerhez, akik előre be nem látható ideig szolgáltatnak adatokat. Ugyancsak van esélye annak, hogy egy felhasználó ártó szándékkal lép be a közös számítási térbe, és szabotálni vagy módosítani akarja a közösen kialakítandó modell paramétereit.

Az FGT egyik legnagyobb erőssége, hogy kifejezetten alkalmas heterogén adathalmazok feldolgozására. Ugyanakkor minden esetben egyedileg kell megoldani az eltérő perifériák megfelelő integrálását, hiszen minden eszköz eltérő számítási és kommunikációs kapacitásokkal rendelkezik. Például, ha egyszerre szeretnénk mobiltelefonoktól és IoT eszközöktől származó adatokat feldolgozni, akkor minden eszközre biztosítanunk kell

egy megfelelő lokális tanító algoritmust, ami jól illeszkedik egy komplexebb globális modellbe. Ezen felül, a rendszerbe integrált IoT eszközök számítási kapacitása sok esetben limitált, ezt figyelembe kell vennünk.

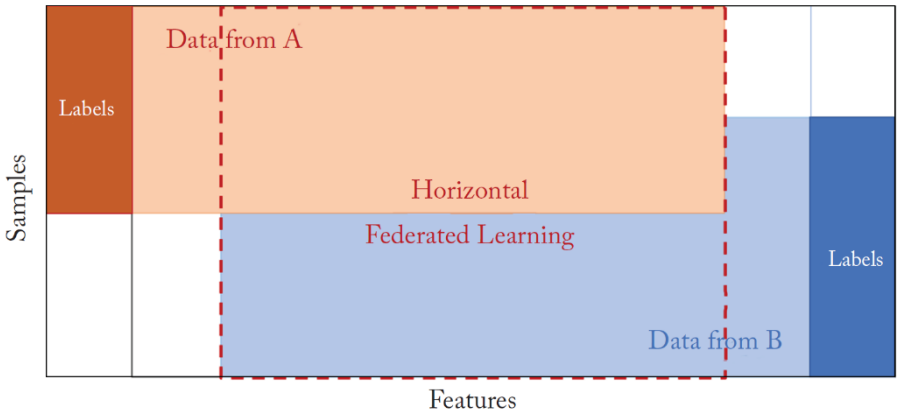
Az elosztott tanulási rendszerekkel ellentétben, federált gépi tanulás esetén nem feltételezzük a lokális adathalmazok egységes eloszlását. Éppen ellenkezőleg, FGT esetén heterogén adathalmazokat feltételezünk, amik mérete is nagyságrendeket átölelő skálán változhat. Könnyen elképzelhető, hogy nem minden felhasználó ugyanazt a funkcionalitást használja egy applikáción belül, illetve nem ugyanakkora intenzitással használja azt, így az eltárolt adatok jellege és mennyisége nagyban eltérhet az egyes csúcsok között. Ebből adódóan mindenképpen egy dinamikusan testre szabható modellre van szükség, ami könnyedén adaptálható minden adatforrás típusra.

A federált gépi tanulási architektúrák rugalmasan alakíthatók az adott problémának megfelelően. Eltérés lehet többek közt a kommunikációs csatornákban (pl. az egyes csúcsok kapcsolatban állnak-e egymással, vagy kizárólag a központi szerverrel cserélnek információt), a perifériás eszközökben (pl. kizárólag azonos típusú eszközök vagy heterogén rendszer) és abban is, hogy milyen további adatvédelmi eljárások és gépi tanulási modellek kerülnek a rendszerbe. Ebben a tanulmányban két nagyobb struktúrát tárgyalunk, amik felépítésüknek köszönhetően eltérő problémák megoldására alkalmasak. Ezek a horizontális, illetve vertikális federált gépi tanulási architektúrák [16].

4.1. Horizontális FGT

A horizontális FGT architektúrát szokás minták szerint felosztott federált gépi tanulásnak (*sample-partitioned federated learning* vagy *example-partitioned federated learning*) is nevezni, mivel abban az esetben alkalmazható, ha a különböző forrásból származó adatok tulajdonságai (*features*) átlapolnak, viszont a minták (*samples*) eltérnek. Például két pénzügyi szervezet várhatóan nagyon hasonló adatokat gyűjt az ügyfeleiről, viszont valószínűleg kevés közös klienssel rendelkeznek. A horizontális megnevezés a táblázatos leírásmódból ered, ahol a sorok (minta pontok) vízszintesen vannak különböző csoportokba osztva, ahogy azt a 12 szemlélteti. Formálisan a következő feltételeknek kell teljesülniük:

$X_i = X_j, Y_i = Y_j, I_i \neq I_j, \forall D_i, D_j, i \neq j$, vagyis a két adathalmaz adatjellemzői és cimketere (*labels*) páronként – (X_i,Y_i) és (X_j, Y_j) – azonosak, valamint az ügyfélazonosítók (I_i és I_j) különböznek bármely két különböző D_i és



12. ábra – Horizontális federált gépi tanulás szemléltetése [16].

D_j adathalmazra. Azaz egy ügyfél egyetlen részadat-halmazban szerepel csak, mindegyikben ugyanaz az előrejelzendő célváltozó (címketér) megtalálható, és kialakítható egy olyan közös bemenő változó készlet, ami minden részadathalmazban megtalálható. Amennyiben egyes szereplőknél többletadat is megjelenik, azt ez az architektúra figyelmen kívül hagyja. Ha egy ügyfél mégis mindkét adathalmazban szerepel, ez a rendszer külön entitásnak tekinti őket. Ez akkor elfogadható megközelítés, ha ezek aránya elenyésző az adathalmazban.

A horizontális modell definíciójából adódóan a minták közt nincs átfedés, minden adatpont külön adatbázisból érkezik. Tehát ez a struktúra ideális olyan esetekben, amikor közvetlenül a felhasználói adatok feldolgozása a cél, vagyis B2C terméknél. Erre egy jó példa az okostelefonokon használható intelligens billentyűzet, a Google fejlesztésű *Gboard on Android*.

A két legelterjedtebb architektúra a horizontális federált gépi tanulás megvalósításához a kliens-szerver, illetve a *peer-to-peer* struktúra.

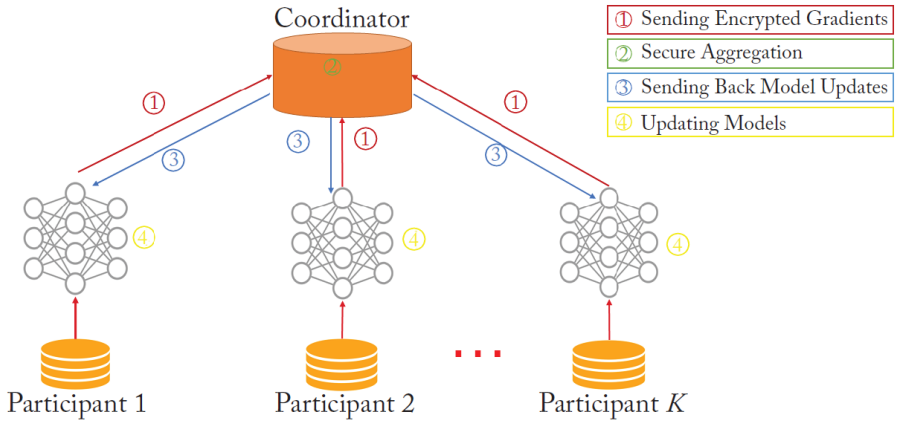
4.1.1. Kliens-szerver architektúra

A kliens-szerver architektúrára, más néven centralizált federált gépi tanulási struktúrára egy tipikus példa látható a 13. Centralizált FGT esetében a perifériák kizárólag a központi szerverrel állnak kapcsolatban. Ez a szerver a felelős a kezdeti modell inicializálásáért, a csúcsok kiválasztásáért és feléjük való adattovábbításért, valamint a visszaérkező módosított modellek összegyűjtését és aggregálását is ez az egység végzi. Mivel minden csúcs közvetlenül a központi szervernek küldi meg az adatokat, ezért sok periféria esetén a szerver kapacitása lehet a rendszer szűk keresztmetszete.

A rendszerben résztvevő *K* felhasználó, vagy csúcs mindegyike közvetlenül a központi szerverhez, vagy aggregátorhoz kapcsolódik. Az adatok minden csúcson azonos struktúrában tárolódnak. A tanulási folyamat lépései a következők:

1) A felhasználók lokálisan lefuttatják a gépi tanulási modellt, majd annak megváltozott paramétereit továbbítják a központi aggregáló egységnek.
2) A központi szerver valamilyen biztonságos módszerrel aggregálja a csúcsokról beérkezett adatokat.
3) A frissített modellparamétereket a szerver visszaküldi a felhasználóknak.
4) Az egyes felhasználók frissítik a modelljüket az új paraméterekkel.

Az iterációs lépések addig ismétlődnek, amíg a modell el nem éri a kívánt pontosságot, vagy valamilyen más határfeltétel (pl. megengedett iterációs lépésszám, számítási idő elérése) be nem következik. Mivel a modell



13. ábra – Példa kliens-szerver architektúrára horizontális federált gépi tanulási rendszer esetén [16].

frissítése párhuzamosan történik az egyes csúcsokon, ezzel az architektúrával lényegesen gyorsabb tanítási folyamatok futtathatók, mint egy központi adatbázis esetén.

Ez az architektúra nagyon hasonló az elosztott gépi tanulási (*distributed machine learning*) modell felépítéséhez. Ugyanakkor fontos elvi különbségek vannak. Az elosztott tanulási rendszerekben az adatok perifériákon tárolódnak, ahonnan egy központi szerver utasítására kerülnek kiolvasásra és feldolgozásra. Ezzel ellentétben FGT esetén a csúcsok önálló entitásoknak tekinthetőek, amik nem adják ki az adatokat és maguk rendelkeznek a tanulási modell elfuttatásáról. További különbség, hogy a federált rendszernek nem feltétele, hogy az adatok egyenletesen oszoljanak meg a csúcsok között, ahogy az sem feltétlenül, hogy homogén struktúrában legyenek eltárolva.

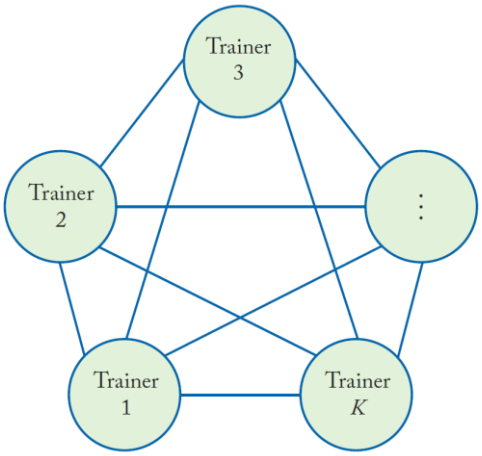
Mivel a kliens-szerver architektúrában a felhasználók közvetlenül az aggregátorral osztják meg a számított értékeket, az adatok biztonsága érdekében szükséges további titkosítási eljárás használata. Ennek hiányában a központi rendszer a modellparaméterek megváltozásából vissza tudná fejteni az eredeti adatokat. A leggyakrabban használt eljárások a differenciált adatvédelem, valamilyen kódolási eljárás (pl. homomorf) vagy a titkos megosztási eljárások egyike.

4.1.2. Peer-to-peer architektúra

Egy másik lehetséges elrendezés horizontális federált gépi tanulás esetén a *peer-to-peer* struktúra, ahol az előzővel ellentétben nem szerepel központi szerver, hanem az adatszolgáltató csomópontok, más néven (elosztott) tanítók vagy munkások, egymás közt küldik a gépi tanulási modell frissített súlyait, ahogy azt a 14 mutatja. Ezt a modellt szokás decentralizált federált gépi tanulási modellnek is nevezni, hiszen a csúcsok egymással is közvetlen kapcsolatban állnak, képesek egymásnak továbbítani a frissített modellparamétereket. Ennek köszönhetően kisebb az esélye, hogy egy-egy csúcs kapcsolati problémák miatt kiesik a rendszerből. Valamint az adatok is nagyobb biztonságban vannak, hiszen nem egy központi algoritmus paramétereit

módosítják az egyes csúcsok, hanem láncszerűen adják tovább az információt, így a központi szerver (ha van) sem tudja, hogy melyik periféria hogyan befolyásolta a folyamat végén létrejött globális modellt.

Aggregátor hiányában az adatszivárgás veszélye lényegesen csökken, ugyanakkor az adatok biztonsága érdekében a kommunikációs csatornák védelme elengedhetetlen. Erre alkalmazható például bármely privát kulcsra épülő titkosítási eljárás. Mivel ez a struktúra teljesen mellőzi a hierarchiát, a dolgozóknak előzetesen meg kell állapodniuk az adattovábbítás sorrendjében. Lényegében két mód áll rendelkezésre: a tanítók meghatározott sorrendben követik egymást, vagy pedig minden tanító egyenlő valószínűséggel véletlenszerűen választja ki, hogy melyik másik csúcsnak küldi tovább a frissített modell súlyokat. Az tanítási folyamat mindkét esetben valamilyen előre meghatározott határ esemény (pl. konvergencia, maximális iterációs lépésszám, maximális idő elérése) bekövetkeztéig tart. A kliens-szerver struktúrával ellentétben, ebben az esetben a súlyok nem párhuzamosan, hanem sorozatosan frissülnek az egyes csúcsokon, ami lényegesen lassítja a tanítási folyamatot.

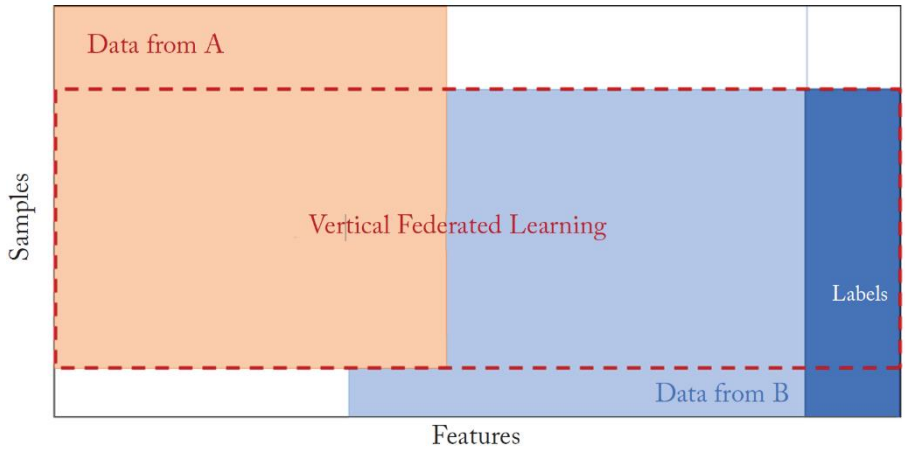


14. ábra – Példa *peer-to-peer* architektúrára horizontális federált gépi tanulási rendszer esetén [30].

4.2. Vertikális FGT

Vertikális FGT modellben, a horizontálissal ellentétben, az adathalmazok azonos szereplőket, de eltérő adatjellemzőket tartalmaznak, ahogy azt a 15 szemlélteti. Ez olyan esetekben fordulhat elő, ahol egy adott szereplő különböző szokásait szeretnénk összekötni, hogy személyre szabottabb szolgáltatást nyújthassunk. Például egy felhasználó vásárlási szokásait egy adott szolgáltatónál kiegészítve ugyanazon felhasználó banki adataival, célzott ajánlatokat tehetünk az illetőnek. Éppen ezért a vertikális, vagy függőleges FGT kifejezetten alkalmas szervezetek közti adatelemzésre irányuló együttműködések megvalósítására. Ebben az esetben az adatok nem a végfelhasználóktól érkeznek, hanem már valamelyest aggregált formában, adatbázisokban találhatók. Az egyes intézmények engedélyezhetik az adatokon való tanítást anélkül, hogy azokat kiadnák a

kezükből vagy akár csak felfednék azt. Ebben a felállásban a résztvevő felek célja egy közös gépi tanulási modell létrehozása a rendelkezésre álló heterogén adatjellemzők felhasználásával. A közös modell felhasználásával mindegyik fél további információhoz juthat, amivel relevánsabb szolgáltatást nyújthat ügyfeleinek.



15. ábra -- Horizontális federált gépi tanulás szemléltetése [16].

Formálisan a következőképpen írhatjuk le a vertikális FGT adatokra vonatkozó feltételeit:

$$X_i \neq X_j, Y_i \neq Y_j, I_i = I_j, \forall D_i, D_j, i \neq j,$$

ahol *X* és *Y* az adatjellemzők és a címkék terét adja meg ebben a sorrendben, *I* a mintákhoz tartozó azonosítók, valamint *D* jelöli az egyes szereplők birtokában lévő adathalmazokat.

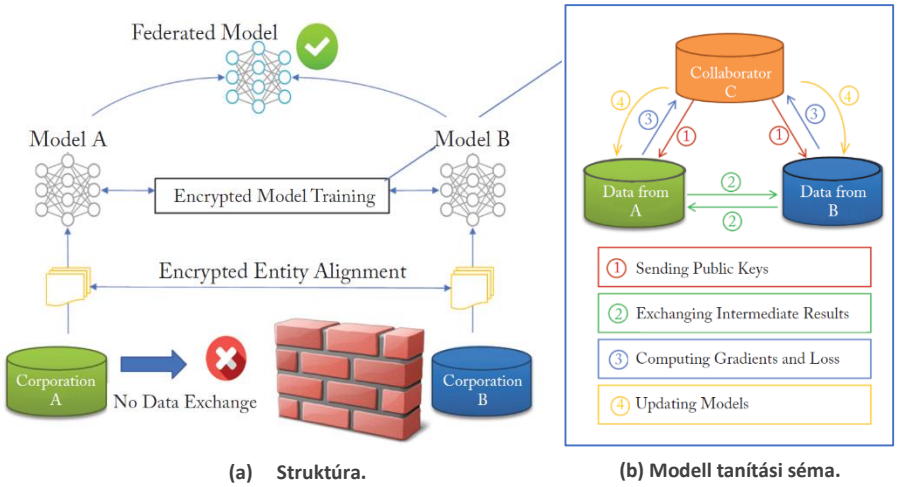
A közös együttműködés záloga a vertikális FGT rendszerben, hogy a résztvevők őszinték-de-kíváncsiak (honest-but-curious), vagyis megosztják az adataikat (valamilyen maszkolt formában), ugyanakkor a lehető legtöbb információt igyekeznek kinyerni a másik féltől kapott adatokból. Mivel szorosan együtt működnek, ezért nincs szükségük a szabályokat megkerülendő összeesküvésre. Fontos még, hogy az adatok és a kommunikációs csatornák megfelelően védettek legyenek, vagyis ha az egyik felet támadás éri, akkor a másik fél adatai semmiképpen ne tudjanak kiszivárogni. A számítások biztonságos elvégzése érdekében előfordulhat, hogy egy köztes szereplő is megjelenik. Az ő feladata, hogy begyűjtse a két fél titkosított adatait, elvégezze a közös modell tanítását, majd a modellparamétereket visszaküldje az adathalmazok tulajdonosainak.

A vertikális FGT modell struktúráját a 16 mutatja be. Az egyszerűség kedvéért csak két szereplőt tekintünk, akik közös gépi tanulási modellt szeretnének építeni. Jelölje *A* és *B* a két szervezetet, akik mindketten rendelkeznek saját adatbázissal, illetve *B* adatai közt szerepel a célváltozó is, amit szeretne prediktálni. Biztonsági és adatvédelmi okok miatt a két szereplő nem oszthatja meg egymással az adatait, ezért szükség van *C*-re, egy harmadik szereplőre, aki biztosítja az adatok védelmét. Feltételezzük, hogy a folyamat résztvevői őszinték és nem konspirálnak egymás közt az adatok kiszivárogtatása érdekében. A tanulási folyamat két főbb lépésből áll:

I. Titkosított entitások összepárosítása (*encrypted entity alignment*): A közös felhasználók azonosítása titkosított ügyfél-azonosítók mentén történik, a személyes adatok védelme érdekében. Csak olyan felhasználók kerülnek a tanító rendszerbe, akik mind *A*, mind *B* adatbázisában szerepelnek. Ennek megvalósíthatóságához elengedhetetlen, hogy a titkosításra használt metódus távolságtartó legyen, vagyis a titkosítás után az adatpontok távolsága ne változzon. Ezt követően egy párosító eljárásra van szükség, ami beazonosítja az adathalmazok közti közös metszetet az adatpontok felfedése nélkül.

II. Titkosított modell tanítása (*encrypted model training*): A beazonosított közös szereplők adatainak felhasználásával történik a közös gépi tanulási modell betanítása. A folyamat négy fő lépésre osztható:

1) *C* létrehoz egy titkosítási kulcs párt és elküldi a nyilvános kulcsot *A*-nak és *B*-nek.
2) *A* és *B* külön-külön inicializálja az algoritmushoz tartozó súlyokat, majd *A* titkosítja és elküldi eredményeit *B*-nek. Így *B* a közös rendszer veszteségfüggvényét is ki tudja számolni, amit szintén titkosítva továbbít *A*-nak.
3) A közös veszteségfüggvény ismeretében *A* és *B* kiszámítja a rá vonatkozó gradiens értékét. A kapott (már titkos) eredményeket egy további maszkkal elfedve elküldik *C*-nek.
4) *C* dekódolja az adatokat, majd visszaküldi mind *A*-nak, mind *B*-nek. Végül *A* és *B* a maszkot eltávolítva frissíti a saját modellparamétereit.



(a) Struktúra. (b) Modell tanítási séma.

16. ábra -- Vertikális FGT (a) struktúra sematikus rajza és (b) a modell tanításának lépései [16].

A teljes algoritmus levezetését, a különböző lépések létének indoklását a forrásként megjelölt cikkből érdemes átlátni, annak részletes tárgyalása meghaladja a tanulmány kereteit.

4.3. Federált gépi tanulásra alkalmas algoritmusok

Az utóbbi évek kutatásainak köszönhetően ma már minden gépi tanulási alapfeladat ellátására létezik federált gépi tanulási algoritmus, így lineáris problémákat és osztályozási feladatokat is meg tudunk oldani vele.

Módszer	Előny	Alkalmazás
Lineáris modellek	Könnyen implementálható, hatékonyan alkalmazható FGT-ra	Lineáris regresszió, Ridge regresszió
Osztályozó modellek	Pontos, stabil és nem lineáris problémák megoldására is alkalmas	Logisztikus regresszió, Döntési fa, Random forest, Gradient boosting machine
Neurális háló alapú modellek	Tanulási képességek, rendkívül robusztus és hibatűrő	Mintázat felismerés, Intelligens vezérlés

1. táblázat - Federált gépi tanulásra alkalmas algoritmusok [17]

A legtöbb implementáció nem csupán a FGT-t teszi lehetővé, de a legtöbb esetben további személyes adatokat védő eljárást is alkalmaz az adatok biztonsága érdekében. Például a korábban tárgyalt differenciált adatvédelmi eljárással, vagy a következő fejezetben bemutatott homomorf titkosítással is jól kombinálhatóez a módszer.

A népszerű gépi tanulási csomagok algoritmusai közül több is képes arra, hogy egy részben illesztett modellt tovább finomítson, futtasson (erre szolgál például az sklearn csomag partial_fit függvénye). Fontos kiemelni ugyanakkor, hogy önmagában ezen feltételek birtokában létrehozható olyan kódbázis, amikor a különböző szereplők közösen építenek fel egy modellt, de ez magában nem biztosítja azt, hogy az egyes résztvevők ne tudják visszafejteni a másik szereplő adathalmazának egyes elemeit. Ez is egy külön érv, miért érdemes a célfeladatra fejlesztett FGT eljárásokat alkalmazni.

A federált gépi tanulás alkalmazása pont a pénzügyi szektor számára a legkézenfekvőbb, az MNB mint szabályozó, komoly szerepet játszhat ezen megoldásokra épülő többletszolgáltatások kialakításában. A különböző kockázati tényezők kiszámítása során ugyanis a pénzintézetek tipikusan csak saját ügyfélkörük historikus adataira támaszkodhatnak, a FGT technikák lehetőséget teremtenének arra, hogy jóval szélesebb körből származó adatok alapján történjék meg a becslés. Ez különösen azon pénzintézetek számára adna előnyt, melyek kis piaci részesedésük miatt kevésbé tudnák reprezentatív adatokkal lefedni a teljes lakosságot.

5. Homomorf titkosítás

Ahogy az az előző fejezetekből is kiderül, az adatok titkosítására szinte minden esetben szükség van, amikor valamilyen információ megosztásra kerül, még ha ez nem is a nyers adat, hanem a gépi tanulási modellhez tartozó súlyok. Egyre gyakoribb a felhő alapú adatelemzés, ami lehetőséget nyújt nagy adatmennyiségek feldolgozásán túl akár több szereplő együttműködésére is. Erre azért van szükség, mert a legtöbb esetben nem az adatokat előállító intézmény építi fel a gépi tanulási modelleket, hanem egy másik, kifejezetten erre szakosodott szervezet. Abban az esetben pedig, ha egy felhő alapú megoldást választunk, az adatszivárgás (legyen az szándékos vagy véletlen) lehetősége mindig fennáll, ezért érdemes az adatokat minél átfogóbb védelemmel ellátni.

A homomorf titkosítás (homomorph encryption) egy nyilvános kulcsú titkosítási séma. A felhasználó létrehoz egy titkos és egy nyilvános kulcsot, majd a nyilvánosat felhasználja adatai titkosítására. A titkosított adatokat tovább küldi az elemzést végző félnek, aki a speciális titkosítási módszernek köszönhetően azok visszafejtése nélkül képes elemezni az adatokat. Teljes homomorf titkosítás esetén mind összeadás, mind szorzás elvégezhető az adatokon a titkosított adatok terében, ami gyakorlatilag tetszőleges logikai és számtani műveletek sorozatát lehetővé teszi az adatok visszafejtése nélkül. Végül a korábbi titkosítást végző adatgazda megkapja a számítások még mindig titkosított formában levő eredményét, amit saját titkos kulcsával tud visszafejteni, így egyedüliként hozzáférve a valós eredményekhez.

Formálisan ez a következőképpen érható le. A homomorfizmus (h) teljesül két gyűrű (X,Y) között, ha

$$h(x + y) = h(x) + h(y),$$

$$h(x * y) = h(x) * h(y),$$

ahol *x* az *X*, *y* pedig az *Y* gyűrű egy tetszőleges eleme.

Legyen továbbá k és d egy kódoló és egy dekódoló homomorfizmus, vagyis $d(k(x)) = x$. Illetve legyen egyen *f* egy függvény, ami kizárólag összeadás és szorzás műveletekből áll. Ekkor az adatfeldolgozás folyamata a következőképpen alakul:

1) Az adat tulajdonosa kódolja az adatokat, és az így kapott $k(x)$ eredményt elküldi a számítást végző félnek.
2) Az analízist végző szereplő kiértékeli az *f* függvényt a kapott értékeken, majd az eredményt visszaküldi a felhasználónak. Mivel *k* homomorf művelet, így az eredményre igaz, hogy $f(k(x)) = k(f(x))$.
3) Végül a kliens dekódolja a kapott információt és így megkapja a kívánt számítási eredményt: $d(f(k(x))) = d(k(f(x))) = f(x)$.

Végeredményül a felhasználó megkapja a számára szükséges információt anélkül, hogy adatait felfedte volna egy külső szervezet előtt.

A továbbiakban áttekintjük a homomorf titkosítással megoldható legfontosabb adatvédelmi feladatokat. Ezen az úton az első legfontosabb kérdés, hogyan tudja két szereplő egyeztetni, hogy vannak-e közös entitások az adathalmazaikban.

5.1. Privát adatok védelme: Private Set Intersection (PSI)

A privát közös metszet keresés (*Private Set Intersection*) egy biztonságos, több félből álló számítási kriptográfiai technika, amely lehetővé teszi, hogy két halmazzal rendelkező fél összehasonlítsa e halmazok titkosított verzióit a közös metszet kiszámítása érdekében. Ebben a forgatókönyvben mindkét fél kizárólag annyi információt oszt meg a másik féllel, amennyi a közös elemek beazonosításához feltétlenül szükséges. Vagyis senki sem tudhatja, hogy a másik adathalmazába milyen további adatok vannak.

Elképzelhető, hogy egy felhasználó profitálni szeretne egy adott szolgáltatásból és ehhez rendelkezésre áll a megfelelő adat, viszont nem szeretné, hogy az illetéktelenek kezébe kerüljön. Például szeretné megtudni, hogy ismerősei közül kik használnak egy adott telefonos applikációt, amit az adott személy is használ. Vagyis van egy nyilvános modell, amit privát adatokon szeretnénk futtatni úgy, hogy annak készítője másra ne használhassa a szóban forgó adatokat. Ez a probléma előfordulhat olyan esetbe, ahol a felhasználó szeretné megvizsgálni, hogy a rendelkezésére álló adatok közül melyek alkotnak metszetet egy nagyobb adathalmazzal. Például egy applikáció felhasználója kíváncsi rá, hogy az ismerősei közül kik használják ugyanazon applikációt, hogy azon keresztül kapcsolatba léphessen velük. Ugyanakkor nem szeretné kiadni ismerőseinek adatait az applikáció tulajdonosának.

Ebben az esetben a felhasználó megosztja az előzetesen titkosított adatokat a szolgáltatóval, aki a szükséges számításokat követően visszaküldi a továbbra is kódolt eredményt. Az ügyfél ezt követően dekódolja azt a rendelkezésére álló privát kulccsal és így hozzájut a kívánt eredményhez. Ez a struktúra akkor tud működni, ha az adatok tárolása és a szükséges algoritmusok futtatása egy rendszerben található, vagy az adattárház (például egy felhő szolgáltató) és az elemzést végző fél kölcsönösen megbíznak egymásban.

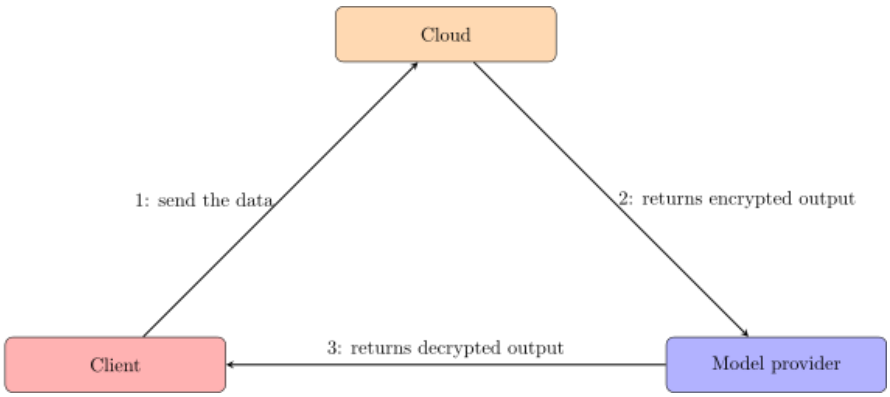
A közös elemek beazonosításának problémája mindenhol megjelenik, ahol két (vagy több) fél adatainak együttes felhasználása szükséges. Erre láttunk példát a federált gépi tanulásnál is. A homomorf titkosítás teszi

lehetővé, hogy a titkosított adathalmazok megosztásakor az elemek közötti hasonlóság mérhető legyen, tehát meg lehessen találni a közös elemeket. Ehhez mindenképpen távolságtartó transzformációra van szükség, aminek a homomorf titkosítás eleget tesz [18].

5.2. Modell védelme

Gépi tanulási modellek használatakor nem csak az adatok, de a modell paramétereinek védelmére is szükség van. Egyrészt azért, mert (például nyilvánosan hozzáférhető adatok esetén) maga a modell az, ami értéket képvisel egy szervezet számára, illetve azért is, mert a tanított modell súlyaiból nagy pontossággal visszanyerhető az eredeti adathalmaz.

Tekintsük most azt az esetet, amikor nyilvánosan hozzáférhető adatokon szeretnénk privát modellt tanítani, ahogy az a 17 látható. Például egy szervezet publikusan hozzáférhető piaci adatok elemzésével és az eredmények eladásával foglalkozik. Ekkor az adatszolgáltató (Client) megosztja kódolatlan adatait egy felhő szolgáltatáson (Cloud) keresztül. Az elemzést végző fél (Model provider) ugyancsak megosztja modelljét a felhőben, viszont előzetesen kódolja annak paramétereit. A felhő szolgáltató elvégzi a szükséges számításokat és az eredményeknek megfelelően frissíti a modellparamétereket. Az eredményt visszajuttatja a modellező félnek, aki dekódolja azt, majd az immár valós eredményeket továbbítja a felhasználónak. A gyakorlatban ez például egy hibrid felhővel valósítható meg, aminek a nyilvános része biztosítja a számítási kapacitást, a privát oldala pedig az érzékeny adatok tárolásáért felel [19].

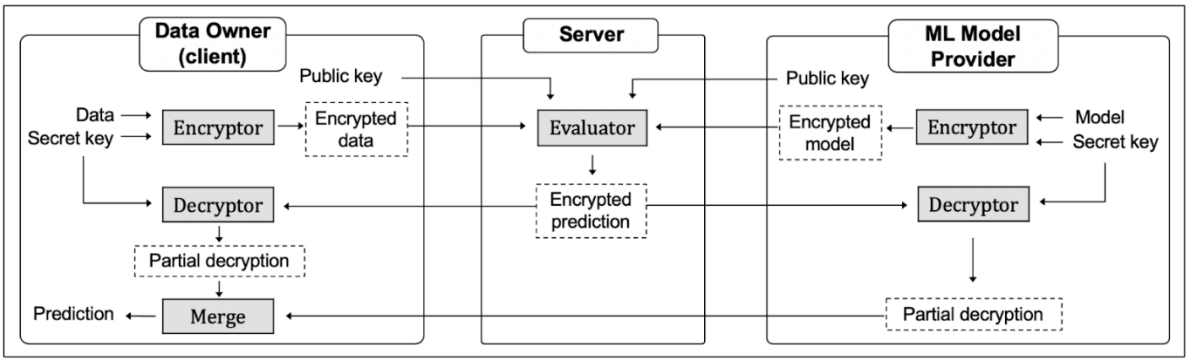


17. ábra -- Gépi tanulási modell védelme homomorf titkosítási eljárással [31].

5.3. Bizalmas adatok kezelése három szereplővel

Az előzőeknél eggyel bonyolultabb struktúra szükséges akkor, ha mind az adatok, mind a modellparaméterek védelme szükséges. A klasszikus homomorf titkosítási rendszerben a rejtjelezett szövegeken csak akkor végezhetők el az elvárt számítások, ha azok ugyanazon nyilvános kulccsal lettek előzetesen kódolva. Ez nyilvánvalóan nem lehetséges, hiszen az adat- és a modellszolgáltató külön kulcsokat használnak annak érdekében, hogy egymástól biztonságban tudják az információikat.

Ebben az esetben megoldás lehet a többkulcsos homomorf titkosítás csomagolt rejtjelekkel, aminek sematikus rajzát a 18 mutatja. Itt a felhasználó (Data Owner) és a modell tulajdonosa (ML Model Provider) is külön-külön titkosítja az adatait, majd azokat elküldik egy központi szervernek (Server), aki elvégzi a modell kiértékelését. Az eredményeket a szerver visszaküldi a többi szereplőnek. Ezt követően a modell tulajdonosa dekódolja az kapott információt a saját privát kulcsának segítségével, majd továbbítja azt a felhasználónak. Végül a kliens is dekódolja a kapott eredményt a saját titkos kulcsának felhasználásával és így hozzáfér az immár valós adatokhoz.



18. ábra – Több szereplős homomorf titkosítási struktúra sematikus rajza [32].

5.4. Homomorf titkosítás a gyakorlatban

A homomorf titkosítás egy jól kidolgozott elméleti rendszer az adatok védelmére, a gyakorlatban azonban egyelőre nem sikerült maradéktalanul megvalósítani. A teljes homomorfizmus (fully homomorf encryption) elérése túl sok számítási kapacitást igényel, ezért helyette közelítő eljárásokat (somewhat homomorf encryption) szokás használni. Legelterjedtebbek a második és a harmadik generációs teljes homomorf titkosítási eljárások. Előbbi főként egész és valós számok feldolgozására, utóbbi pedig logikai problémák kezelésére alkalmas.

További problémát jelent, hogy a valóságban véges számú művelet végezhető csak el anélkül, hogy az adatokat terhelő zaj akkorára nőjön, hogy az már a visszafejthetőséget gátolja. Ráadásul a műveletek fajtája is erősen limitált, hiszen csak összeadás és szorzás esetén teljesül a homomorfizmus feltétele. Mindezek ellenére a homomorf titkosítás egy intenzíven kutatott és gyorsan fejlődő terület, többek közt már a Microsoft és az IBM is rendelkezik saját, nyílt forráskódú implementációval [20], [21], illetve létezik független python könyvtár is [22].

Megfelelő algoritmust választva a homomorfikusan titkosított adatok regressziós [23], osztályozási [24], [25] és *deep learning* [26] problémák megoldására is használható.

6. Összefoglaló

Elmondható, hogy a biztonságos és a személyes adatokat védő mesterséges intelligencia megoldásokra széles körű kereslet mutatkozik. Éppen ezért, az utóbbi néhány évben intenzíven kutatott területté vált. A kérdéskört különböző irányokból megközelítve eltérő megoldásokat találhatunk attól függően, hogy milyen támadási formát igyekszünk kivédeni, illetve, hogy a nyers adatok, avagy a gépi tanulási algoritmus védelme a hangsúlyosabb.

Összességében elmondható, hogy a legtöbb módszer még kiforratlan, ennek ellenére, szinte mindegyikre láthatunk példát a gyakorlatban. A kulcs mindenképp az, hogy ezen eljárások egymást kiegészítik, nem egymás helyettesítői [27]. Attól kezdve, hogy bármilyen adatok továbbításra kerülnek, elengedhetetlen azok titkosítása. Ugyanakkor önmagában a titkosítás nem feltétlenül védi meg az adatokat, illetve sok esetben kifejezetten törekedni kell a felhasználói adatok minimális mozgatására, ami további adatvédelmi eljárásokat tesz szükségessé. Az itt bemutatott módszerek mind a felhasználói, mind a modellezői oldal védelmére kiterjedő megoldásokat kínálnak, amik akár együttesen is bevethetők.

7. Hivatkozásjegyzék

[1] G. A. Kaissis, M. R. Makowski, D. Rückert, és R. F. Braren, „Secure, privacy-preserving and federated machine learning in medical imaging”, *Nat. Mach. Intell.*, köt. 2, sz. 6, o. 305–311, 2020, doi: 10.1038/s42256-020-0186-1.

[2] UnitedStates, „How the Census Bureau Protects Your Data”, 2020. https://2020census.gov/en/data-protection.html.

[3] Apple, „Learning with Privacy at Scale”, 2017. https://machinelearning.apple.com/research/learning-with-privacy-at-scale.

[4] D. Alabi, A. McMillan, J. Sarathy, A. Smith, és S. Vadhan, „Differentially Private Simple Linear Regression”, o. 1–20, 2020, [Online]. Elérhető: http://arxiv.org/abs/2007.05157.

[5] W. Fan, J. He, M. Guo, P. Li, Z. Han, és R. Wang, „Privacy preserving classification on local differential privacy in data centers”, *J. Parallel Distrib. Comput.*, köt. 135, o. 70–82, 2020, doi: 10.1016/j.jpdc.2019.09.009.

[6] S. Fletcher és M. Z. Islam, „Decision tree classification with differential privacy: A survey”, *ACM Comput. Surv.*, köt. 52, sz. 4, o. 1–35, 2019, doi: 10.1145/3337064.

[7] Y. Zhang, N. Liu, és S. Wang, „A differential privacy protecting K-means clustering algorithm based on contour coefficients”, *PLoS One*, köt. 13, sz. 11, o. 1–15, 2018, doi: 10.1371/journal.pone.0206832.

[8] C. Xia, J. Hua, W. Tong, és S. Zhong, „Distributed K-Means clustering guaranteeing local differential privacy”, *Comput. Secur.*, köt. 90, 2020, doi: 10.1016/j.cose.2019.101699.

[9] L. Ni, C. Li, X. Wang, H. Jiang, és J. Yu, „DP-MCDBSCAN: Differential privacy preserving multi-core DBSCAN clustering for network user data”, *IEEE Access*, köt. 6, o. 21053–21063, 2018, doi: 10.1109/ACCESS.2018.2824798.

[10] S. Miyabe *és mtsai.*, „DIFFERENTIALLY PRIVATE DISTRIBUTED PRINCIPAL COMPONENT ANALYSIS Hafiz Imtiaz and Anand D . Sarwate Department of Electrical and Computer Engineering , Rutgers University”, *IEEE Int. Conf. Acoust. Speech, Signal Process. Proc.*, köt. 1, sz. 1, o. 2201–2205, 2018.

[11] M. Gong, K. Pan, Y. Xie, A. K. Qin, és Z. Tang, „Preserving differential privacy in deep neural networks with relevance-based adaptive noise imposition”, *Neural Networks*, köt. 125, o. 131–141, 2020, doi: 10.1016/j.neunet.2020.02.001.

[12] N. Holohan, „Diffprivlib Documentation”, 2020.

[13] J. Oss és A. Kopp, „Privacy Preserving Machine Learning with Differential Privacy¶”, 2020. https://github.com/opendp/smartnoise-samples/blob/master/whitepaper-demos/4-ml-dp-classifier.ipynb.

[14] B. McMahan és D. Ramage, „Federated Learning: Collaborative Machine Learning without Centralized Training Data”, *Google AI Blog*, 2017. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[15] A. Bhowmick, J. Duchi, J. Freudiger, és G. Kapoor, „Federated Learning 白皮书”, o. 1–50, 2018.

[16] Q. Yang, „Federated Learning - https://federated.withgoogle.com/”, 2019, [Online]. Elérhető: https://federated.withgoogle.com/.

[17] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, és Y. Gao, „A survey on federated learning”, *Knowledge-Based Syst.*, köt. 216, o. 106775, 2021, doi: 10.1016/j.knosys.2021.106775.

[18] H. Chen, K. Laine, és P. Rindal, „Fast private set intersection from homomorphic encryption”, *Proc. ACM Conf. Comput. Commun. Secur.*, o. 1243–1255, 2017, doi: 10.1145/3133956.3134061.

[19] U. Nations, „UN Handbook on Computation Techniques”, 2019, [Online]. Elérhető: http://publications.officialstatistics.org/handbooks/privacy-preserving-techniques-handbook/UN Handbook for Privacy-Preserving Techniques.pdf.

[20] Microsoft, „Microsoft SEAL”, 2018. https://www.microsoft.com/en-us/research/project/microsoft-seal/#!release-news.

[21] IMB, „Homomorphic Encryption Services”, 2020. https://www.ibm.com/security/services/homomorphic-

encryption.
[22] C. Science és C. Science, „pyFHE - A Python Library for Fully Homomorphic Encryption”, sz. 2019, 2020.
[23] G. Qiu, X. Gui, és Y. Zhao, „Privacy-Preserving Linear Regression on Distributed Data by Homomorphic Encryption and Data Masking”, *IEEE Access*, köt. 8, o. 107601–107613, 2020, doi: 10.1109/ACCESS.2020.3000764.
[24] S. Arita és S. Nakasato, „Fully Homomorphic Encryption for Classification in Machine Learning”, *2017 IEEE Int. Conf. Smart Comput. SMARTCOMP 2017*, o. 2–5, 2017, doi: 10.1109/SMARTCOMP.2017.7947011.
[25] J. H. Cheon, D. Kim, Y. Kim, és Y. Song, „Ensemble method for privacy-preserving logistic regression based on homomorphic encryption”, *IEEE Access*, köt. 6, o. 46938–46948, 2018, doi: 10.1109/ACCESS.2018.2866697.
[26] M. Minelli, „Fully Homomorphic Encryption for Machine Learning. (Chiffrement TotalementHomomorphe pour l’Apprentissage Automatique)”, 2018, [Online]. Elérhető: https://tel.archives-ouvertes.fr/tel-01918263.
[27] H. Fang és Q. Qian, „Privacy preserving machine learning with homomorphic encryption and federated learning”, *Futur. Internet*, köt. 13, sz. 4, o. 1–20, 2021, doi: 10.3390/fi13040094.
[28] D. Desfontaines, „Ted is writing things On privacy, research, and privacy research.”, 2018. https://desfontain.es/privacy/differential-privacy-awesomeness.html.
[29] W. Commons, „File:Laplace pdf mod.svg --- Wikimedia Commons{,} the free media repository”, 2020. https://commons.wikimedia.org/w/index.php?title=File:Laplace_pdf_mod.svg&oldid=453660607.
[30] Q. Yang, Y. Liu, Tianjian Chen, és Yongxin Tong, „Federated Machine Learning: Concept and Applications”, *arXiv*, köt. 10, sz. 2, o. 1–19, 2019.
[31] D. Huynh, „Homomorphic Encryption intro”, 2020. https://towardsdatascience.com/homomorphic-encryption-intro-part-1-overview-and-use-cases-a601adcff06c.
[32] H. Chen, M. Kim, W. Dai, és Y. Song, „Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference”, *Proc. ACM Conf. Comput. Commun. Secur.*, o. 395–412, 2019, doi: 10.1145/3319535.3363207.