VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Baigiamasis bakalauro darbas

Privačios informacijos išsaugojimas taikant dirbtinio intelekto technologijas

(Privacy-preserving artificial intelligence)

Atliko: 4 kurso 3 grupės studentas
Paulius Milmantas (parašas)
Darbo vadovas:
Dr. Linas Petkevičius (parašas)
Recenzentas:
Dr. Tomas Plankis (parašas)

Vilnius
2021

Turinys

Santrauka

Darbe yra tiriami ir lyginami privatumą saugantys dirbtinio intelekto algoritmai pagal jų saugumą, našumą ir panaudojamumą, bei pateikiamos rekomendacijos. Algoritmų lyginimui darbe pateikta metrika, kuri leidžia palyginti skirtingus mašininio mokymosi modelius pagal tai, kiek jie atskleidžia pradinius duomenis. Tyrimų tikslams buvo sukurti ir lyginti "PyTorch" neuroninis tinklas, gradientinio nuolydžio metodas ir homomorfinis šifravimas su gradientinio nuolydžio metodu. Tyrime pastebėta, kad mašininio mokymosi modeliams pasiekus 70% tikslumą, "PyTorch" neuroninio tinklo modelis yra saugesnis už gradientinio nuolydžio metodą ir homomorfinį šifravimą. Modelių tikslumui artėjant 100%, "PyTorch" neuroninio tinklo modelis labiau prisiriša prie pradinių duomenį, o gradientinio nuolydžio metodas ir homomorfinio šifravimo algoritmas didėjant modelio tikslumui, mažiau prisiriša prie pradinių duomenų ir tampa saugesnis.

Raktiniai žodžiai: Mašininis mokymasis, duomenų saugumas, homomorfinis šifravimas

Summary

The work is evaluating privacy-preserving artificial intelligence algorithms by their security, performance, usability and provides recommendations. For evaluating algorithms, a metric was proposed, which allows to compare different machine learning models by how much they reveal training data. For experiments, “PyTorch” neural network, gradient descent algorithm and homomorphic encryption models were created. The conclusion was that when machine learning models reach 70% accuracy, “PyTorch” neural network model is more secure than gradient descent method and homomorphic encryption with gradient descent. When model accuracy is close to 100%, “PyTorch” neural network starts to reveal more information, gradient descent method and homomorphic encryption with gradient descent becomes more secure and reveals less data.

Keywords: Machine learning, data privacy, homomorphic encryption

Įvadas

Mašininis mokymas yra dirbtinio intelekto sritis, kuri pasitelkia statistinius algoritmus, kad apibrėžtų duomenų generavimo mechanizmą, ar egzistuojančius sąryšius, priklausomybes. Modelis dažnai turi didelį kiekį nežinomų parametrų, kuriuos reikia įvertinti iš duomenų, todėl modelio apmokymui dažniausiai reikia turėti daug duomenų. Kai kurie uždaviniai reikalauja duomenų, kurie nėra laisvai prieinami ir yra privatūs. Mašininio mokymo tyrimų srityje yra kilusi problema dėl jų saugojimo [BJJ+18]. Vienas iš faktorių, kuris lėmė šį susidomėjimą yra 2016 metais Europos Sąjungoje priimtas duomenų apsaugos reglamentas (GDPR). Pagal jį, fizinių asmenų duomenys turi būti saugomi naudojantis tam tikromis taisyklėmis ir negali būti atskleisti trečiosioms šalims, be asmens sutikimo [Par16].

Šią problemą išspręsti siekia įvairūs tyrimai ir naujai pasiūlyti metodai privatumą saugančio dirbtinio intelekto srityje. Šią problemą galima išskaidyti į kelias atskiras sritis:

- Analizuojamų duomenų privatumas [Tha20]. Algoritmas apmoko modelį atpažinti duomenis. Turint sukurtą modelį, neturi būti galima atgaminti duomenų, pagal kuriuos jis buvo mokomas, bei negali būti identifikuoti asmenys. Taip nukentėtų žmonių privatumas ir būtų pažeistas Europos duomenų apsaugos reglamentas. Šio pažeidimo pavyzdys gali būti ir paprastas teksto atkūrimo modelis. Duodama sakinio pradžia, modelis nuspėja jo pabaigą. Jeigu suvedus tam tikras detales modelis užbaigia sakinį naudodamas asmeninius duomenis, kurie atskleidžia žmonių tapatybę, šis modelis nėra saugus [CTW+20].

- Duomenų įvesties privatumas. Trečios šalys neturi matyti įvedamų duomenų. Tai gali būti tinklo saugumo spragos, duomenų surinkimo aplikacijų spragos ir t.t…

- Modelio išvesties privatumas. Modelio išvesties neturi matyti asmenys, kuriems šie duomenys nepriklauso. Šis punktas yra sąlyginis, priklauso nuo modelio svarbos. Jeigu tai yra svarbūs asmeniniai duomenys, negalima rizikuoti. Tačiau jeigu tai yra viešai prieinami duomenys, šis punktas negalioja.

- Modelio apsauga. Sukurtas modelis negali būti niekieno pasisavintas. Šis punktas yra skirtas apsaugoti programos kūrėją.

Darbo tikslas - ištirti ir palyginti privatumą saugančius dirbtinio intelekto algoritmus pagal jų saugumą, našumą ir panaudojamumą, bei pateikti rekomendacijas.

Darbo uždaviniai:

- Išanalizuoti esamus algoritmus pagal jų saugumą ir panaudojamumą.

- Identifikuoti kriterijus, kurių pagalba galima įvertinti privatumo išsaugojimą, bei palyginti algoritmus tarpusavyje.

- Ištirti kurie algoritmai yra realizuoti ir realizuoti dalį algoritmų, kurie nėra atvirai prieinami.

- Palyginti algoritmus pagal našumą ir pateikti rekomendacijas.

1 Asmens duomenų privatumas mašininio mokymo kontekste

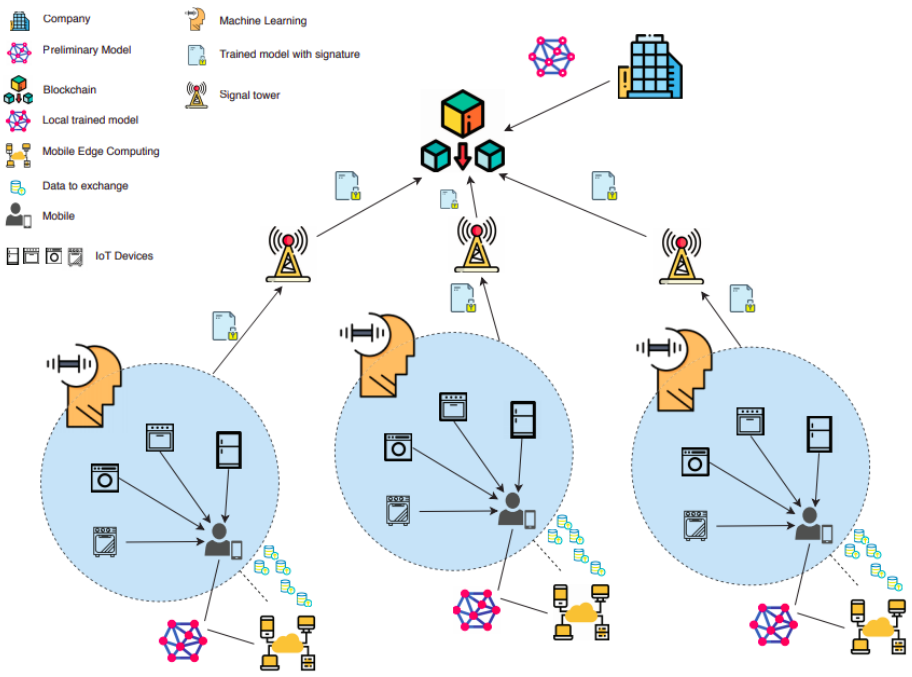
Vienas iš būdų apsaugoti duomenis yra decentralizuoti modelį ir naudoti paskirstyto mokymo algoritmus. Paprastai mašininiam mokymui yra naudojamas centralizuotas serveris. Yra surenkami duomenys į vieną vietą ir modelį moko vienas kompiuteris. Jeigu norima pridėti daugiau duomenų arba papildomą klasifikatorių, modelį reikia apmokyti iš naujo. Taip pat, kas kuria šį modelį, turi visą prieigą prie duomenų, šie modeliai nėra saugūs [BEG+19].

Šiai bėdai išspręsti, vienas iš metodų yra federuotas mašininis mokymas ("Federated machine learning"). Šis metodas padeda išspręsti kai kurias bėdas, vykdant modelio mokymą decentralizuotai [Bha19]. Prie bendro tinklo gali prisijungti daug įrenginių. Visi jie turi savo unikalų duomenų rinkinį. Kiekvienas įrenginys pasirenka geriausią statistinį metodą modelio mokymui ir pagal tą metodą sukuria modelį. Taip yra sukuriama daug skirtingų modelių, realizuotų su skirtingais duomenų rinkiniais. Visi šie rinkiniai vėliau yra surenkami į vieną vietą ir toliau naudojami duomenų analizei. Taip surinkus atskirų įrenginių sukurtus modelius, neturime prieigos prie pradinių duomenų ir daugiau žmonių gali prisidėti prie modelio kūrimo, nematant pilno duomenų rinkinio. Tačiau naudojant šį metodą ne visos problemos yra išsprendžiamos. Duomenų saugumas nėra garantuojamas [Bha19]. Jeigu duomenys nėra užšifruojami, jie gali būti pavogti. Tai gali būti padaryta, jeigu yra naudojamas nesaugus interneto ryšys, arba jeigu yra bandoma analizuoti atskirų įrenginių atsiųstus modelius. Šis metodas išsprendžia tik kelias problemas. Likusios yra: komunikavimo kaštai, įrenginių heterogeniškumas, statistinis heterogeniškumas ir saugumo problemos [LSTS19].

Visos šios problemos yra nagrinėjamos ir joms spręsti kuriami nauji metodai. Komunikacijos kaštams mažinti, yra kuriami algoritmai, kaip turi būti perduodamas modelis tinkle, kad būtų maža tinklo apkrova. Įrenginių heterogeniškumui spręsti, yra parenkami tam tikri įrenginių rodikliai ir pagal tai nustatoma, kokia bus įrenginio komunikacija: ar jis naudos asinchroninį komunikavimą, kaip dažnai tai vyks ir kokia yra įrenginio klaidos tikimybė [LSTS19]. Statistikos heterogeniškumui pataisyti kuriami yra metodai, kaip meta duomenų-mokymas ir keletos užduočių mokymas ("Multi-task learning").

Straipsnyje apie privatumą išsaugančius blokų-grandinių tinklus ("Privacy-Preserving Blockchain-Based Federated Learning for IoT Devices") yra pateikiamas naujas paskirstyto mokymosi algoritmas, kuris yra paremtas blokų grandinių technologijomis [ZZJ+20]. Šis modelis yra grafiškai pateiktas pav.1. Šią architektūrą pavyzdyje sudaro 3 komponentai: gamintojai, klientai ir blokų grandinių tinklas. Šiame pavyzdyje gamintojas pareiškia užklausą, kad reikia atlikti apklausą. Klientai, kurie sutinka su užklausa, išsiunčia savo sukurtą modelį. Blokų grandinių tinklas elgiasi kaip centralizuotas serveris ir surenka visus klientų modelius. Tuomet, pasirinktas kompiuteris, kuris atlieka visą darbą ("miner") atlieka galutinį modelių sujungimą.

Minėti decentralizuoti būdai apjungia skirtingus sukurtus modelius į vieną ir visi įrenginiai kurie kuria modelius, naudoja savo unikalius duomenis. Saugus skirtingų pusių skaičiavimo metodas ("Secure multiparty computation") yra kriptografinis protokolas, kuris leidžia įrenginiams, su unikaliais duomenimis, skaičiuoti funkcijos reikšmę, nematant kitų įrenginių reikšmių [Inp]. Su šiuo



1 pav. Paskirstytas mokymas [ZZJ+20]

metodu, yra sukuriamas vienas modelis tarp įvairių įrenginių. Pagrindinis įrenginys, kuris ruošia skaičiavimo užklausą, gali suskaidyti pradinius duomenis, užšifruoti juos ir taip paskirstyti tarp kitų įrenginių. Taip sukurti įrenginio rezultatai negali būti atšifruoti ir pasisavinti, o minėto federuoto mokymo metodo sukurti rezultatai, jeigu nėra gerai apsaugoti, gali būti pasisavinti ir atgauti pradiniai duomenys su kuriais modelis buvo sukurtas.

Kitas būdas apsaugoti modelį yra naudoti homomorfinį šifravimą. Pagal šį metodas, modelis yra mokomas naudojant užšifruotais duomenimis. Jie mokymo metu, nėra atšifruojami. Šis metodas leidžia keliems įrenginiams vienu metu atlikti skaičiavimus su duomenimis, kurių jie nemato. Yra daug metodo varijacijų. Kai kurios yra pažeidžiamos ir gali atskleisti visos infrastruktūros duomenis [Sen13]. Dažniausiai homomorfinės sistemos yra labiau pažeidžiamos nei nehomomorfinės.

2016 metais buvo pasiūlytas naujas metodas paspartinti pilnai homomorfines šifravimo sistemas [CKKS16]. Homomorfinės sistemos, kurios naudoja šį metodą yra laikomos ketvirtos kartos. Šis metodas leidžia aproksimuoti užšifruoto teksto sudėtį, daugybą ir pakeisti atšifruoto teksto proporcijas. Proporcijų pakeitimo procedūra suskaido užšifruotą tekstą į dalis, to pasekoje yra apvalinamas neužšifruotas tekstas. Šio metodo pagrindinė idėja yra pridėti triukšmo prie pagrindinės žinutės. Šis triukšmas yra pridedamas prie neužšifruoto teksto dėl saugumo priežasčių ir jis yra laikomas kaip aproksimavimo paklaida. Tuo pasekoje, metodas veikia su tam tikra paklaida

Bendro įspūdžio agregavimo metodas yra naujausias iš visų minėtų. Jis leidžia įrenginiui su savo duomenimis sukurti modelį naudojant bet kokius metodus ir taip prisidėti prie bendro progreso [Inp]. Visi modeliai yra surenkami ir leidžiami duomenys per šiuos modelius. Kiekvienas modelis skiria savo balsą, jie surenkami ir padaromas bendras sprendimas. Šis metodas leidžia

lengvai plėsti modelį. Taip pat yra saugomas ir duomenų privatumas. Jeigu keli modeliai, kurie nesidalina duomenimis, teigia vienodai, tai reiškia, kad negalima atkurti duomenų ir jie yra saugūs. Skaičiavimas vykdomas pasirenkant optimalią strategiją ir imant mažiausią galimą žingsnių skaičių.

2 Duomenų pažeidžiamumo metrikos

2.1 Matematinė uždavinio formuluotė

Norint apsaugoti duomenis, sukurtam duomenų aptikimo modeliui reikia atlikti analizę, kaip tikėtina, kad modelio pradiniai duomenys bus atkurti. Tai išanalizuoti yra daug būdų, vienas iš jų yra pateiktas straipsnyje [CLE+19]. Šis straipsnis ieško metrikos reikšmės, kuri parodo, kaip pradinius duomenis atsimena modelis.

Tarkime turime duomenų rinkinį s[r]. Šiam rinkiniui pirma reikia apskaičiuoti rangą (1).

$$rangas(s[r]) = |r` \in R : Px(s[r`]) \leq Px(s[r])| \tag{1}$$

Čia funkcija Px – logaritminis entropijos matas, kuris nusako, ar modeliui paduoti duomenys yra tipiniai, ir ar modelis buvo matęs panašius duomenis. Didesnė Px reikšmė reiškia, kad modelis panašių duomenų nebuvo matęs, o mažesnė reikšmė reiškia, kad panašius duomenis modelis yra matęs. Logaritminės entropijos formulė pateikta lygtyje (2). $s[.]$ – yra pradinis duomenų rinkinys. Skaičius skliaustuose parodo, kuriuos duomenis reikia paimti iš rinkinio.

$$Px_\theta(x_1, ..., x_n) = -\log_2 Pr(x_1, ..., x_{i-1}|f_\theta) \tag{2}$$

Čia formulė Pr – nežinomas skirstinys. Jeigu skaičiuotume funkciją $Pr(x_i|x_1, ..., x_n)$, gautume tikimybę, kad x_i atsiranda po reikšmių, nuo x_1 iki x_{i-1}

Šis rangas nurodo, kurioje vietoje yra šis inicijuotas duomenų rinkinys sąraše, tarp visų galimų testinių rinkinių kombinacijų. Pavyzdžiui, norima sukurti natūralios kalbos atpažinimo modelį. Jo mokymui, yra paduodamas atsitiktinai sugeneruotas sakinys “Atsitiktinis skaičius 125” ir apskaičiuojama jo entropija. Tada galima išrašyti visus galimus sakinius, kuriuos yra leidžiama siųsti, ir juos surūšiuoti pagal entropijos laipsnį. Tada reikia apskaičiuoti, kaip dažnai pasitaiko panašūs sakiniai. Taip yra gaunamas rangas. Kadangi šis pavyzdinis sakinys yra sudėtingas, su atsitiktiniais skaičiais ir jo entropija yra aukšta, tikriausiai aukštesnė nei dauguma mokymo duomenų, jo rangas bus artimas vienetui. Darant prielaidą, kad šio sakinio entropija yra pati aukščiausia iš galimų testavimo duomenų, galima teigti, jog šio sakinio rangas yra lygus vienetui.

Rangas tiesiogiai nepasako kokia yra tikimybė, kad bus sugeneruotas toks testavimo rinkinys. Jo skaičiavimas paima daug resursų, nes reikia sugeneruoti visas galimas kombinacijas [CLE+19].

Kitas nagrinėjamas kintamasis yra “atvirumo” metrika. Tai aproksimacija, kuri nusako, kaip tikėtina, jog testavimo duomenys bus atgaminti iš modelio. Ši metrika pasako ką naujo sužinome apie pradinius duomenis, kai per modelį paleidžiame atsitiktinai sugeneruotus duomenis. Todėl, reikia skaičiuoti prognozuojamą entropiją, pagal pateiktas modelio prognozes. Prognozės entropija, tai spėjimų skaičius $E(X)$, kuris reikalingas atspėti diskretų, atsitiktinai sugalvotą parametrą X.

Jeigu kintamasis r yra pasirenkamas atsitiktinai $r \in R$, tai reiškia, kad reikia generuoti atsitiktinius skaičius tol, kol bus gauta r reikšmė. Iš to seka, kad spėjimų turėtų būti lygus (3) lygčiai.

$$E(s[r])_\theta = \frac{1}{2}|R| \tag{3}$$

Turint suskaičiuotus galimus rinkinių rangus, bei entropijas, galima naudoti patobulinta skaičiavimo taktiką. Visi galimi duomenys yra surušiuojami pagal entropiją arba sudėtingumą. Sąrašo pradžioje yra elementas su mažiausia entropija. Jo atspėjimo tikimybė yra viena iš didžiausių. To pasekoje, gauname formulę (4).

$$E(s[r]|f_\theta) = rank_\theta(s[r]) \tag{4}$$

Ši formulė padidina skaičiavimų spartumą ir tai galima apskaičiuoti padalinus vieną formulę iš kitos. Tai pateiktas (5) lygtyje.

$$\frac{E(s[r])}{E[s[r]|f_\theta]} = \frac{\frac{1}{2}|R|}{rank_\theta(s[r])} \tag{5}$$

Rangai tiksliai neapibrėžia kokia yra tikimybė, jog elementas bus atspėtas. Jis tik lygina visas galimas kombinacijas ir skaičiuoja jų entropijas. To pasekoje, visi šie skaičiavimai yra tik apytikslis spėjimas. Kadangi tai nėra tikslu ir norima sužinoti tik bendrą vaizdą apie algoritmą, galima naudoti logaritmus. Tai atlikta lygtyje (6).

$$log_2(\frac{E(s[r])}{E(s[r]|f_\theta)}) = log_2(\frac{\frac{1}{2}|R|}{rank_\theta(s[r])}) \tag{6}$$

Šias lygtis supaprastinus, gaunama lygtis (7).

$$log_2|R| - log_2(rank_\theta(s[r])) - 1 \tag{7}$$

Dėl paprastumo yra pridedamas vienetas ir taip gaunama atvirumo metrika, pateikta lygtyje (8).

$$atvirumas(s[r])_\theta = \log_2|R| - \log_2 rangas_\theta(s[r]) \tag{8}$$

2.2 Skaičiavimas praktikoje

Pažeidžiamumo metrika “Atvirumas” (angliškai “Exposure”) yra naudojama tikrinant mašininio mokymo algoritmų duomenų įsiminimą. Atlikus šiuos skaičiavimus, programuotojai gali spręsti dėl tolimesnių veiksmų: tęsti kūrimą ar dirbti ties duomenų anonimizavimu.

Mašininio mokymo algoritmo tikrinimui su atvirumo metrika, pirma reikia pasiruošti duomenų rinkinį s[r]. Sudarius šį rinkinį, reikia sukurti mašininio mokymo modelį. Turint gautą modelį ir duomenis, kurie buvo naudojami modelio gavimui, galima skaičiuoti atvirumo metriką. Siekiant daugiau sužinoti apie modelį, galima savo duomenis įterpti daugiau nei vieną kartą. Pavyzdžiui, jeigu vienus duomenis įterpsime kelis kartus, o kitus duomenis šimtą kartų, galima analizuoti atvi-

rumo metriką su šiais duomenimis. Taip galima gauti abstrakčią koreliaciją tarp duomenų kiekio ir modelio atsiminimo. Kad šie eksperimentai būtų tikslūs, būtina kiekvieno modelio kūrimo metu naudoti tuos pačius parametrus: optimizavimo funkcijas, hiperparametrus ir kitus duomenis. Taip su kiekvienu pavyzdiniu duomenų rinkiniu, turi būti atlikta atskira analizė. Jeigu galima, turi būti didinamas vienodų duomenų kiekis ir tikrinama atvirumo ir žinomų duomenų kiekio koreliacija.

Atvirumui sužinoti, reikia skaičiuoti rangą. Tikslaus jo radimo procedūros nėra. Todėl reikia naudotis analitiniais metodais. Vienas iš būdų, tikimybiškai vertinti aproksimuotą skirstinį. Tarkime, turime duomenis, kurių entropija yra didesnė, nei s[r] duomenų. Reikia rasti kiek yra tokių duomenų, kurių entropija yra mažesnė. Taip pat, tarkime, kad s[r] entropija yra $\rho(.)$ pasiskirstymo distribucijos. Atvirumą galima aproksimuoti skaičiuojant šio skirstinio plotą grafike, iki testavimo duomenų, ir dauginant iš logaritmo. Taip gauname formulę (9).

$$atvirumas(s[r])_\theta = -log_2 \int_0^{Px_\theta(s[r])} \rho(x)dx \tag{9}$$

2.3 Iliustracinis pavyzdys

Metodo panaudojamumą pademonstruoti, pasinaudokime mašininiu modeliu, pagrįstu cukriniu diabetu sergančių moterų duomenimis. Modelis priima 5 parametrus: nėštumų skaičių, gliukozės kiekį kraujyje, kraujo spaudimą, BMI ir amžių. Visų šių parametrų ribos yra žinomos, todėl turint atsitiktinių duomenų rinkinį, galima apskaičiuoti rangus. Todėl bus naudojama lygtis (10), turinti rangus.

$$atvirumas(s[r])_\theta = \log_2 |R| - \log_2 rangas_\theta(s[r]) \tag{10}$$

Tarkime, kad šių duomenų galimos ribos yra tokios: nėštumai: [0-30], gliukozės kiekis kraujyje: [0-180], kraujo spaudimas: [0-250], BMI: [1-70], amžius: [1-110]. Suskaičiuoti kiek iš viso yra galimų variantų, galima sudauginus visas šias ribas, tai padaryta lygtyje (11).

$$R = 30 \cdot 180 \cdot 250 \cdot 69 \cdot 109 = 10153350000 \tag{11}$$

Tarkime, turime dviejų žmonių testinių duomenų rinkinį. Pirmo žmogaus duomenys: [2, 60, 98, 25, 25], antro: [0, 90, 128, 27, 18]. Skaičiuojame rangus (12) ir (13). Vėliau pagal apskaičiuotus rangus, ieškoma atvirumo metrika (14) ir (15).

$$s[1] = 2 \cdot 180 \cdot 250 \cdot 69 \cdot 109 + 60 \cdot 250 \cdot 69 \cdot 109 + 98 \cdot 69 \cdot 109 + 25 \cdot 109 + 25 = 790444808 \tag{12}$$

$$s[2] = 90 \cdot 250 \cdot 69 \cdot 109 + 128 \cdot 69 \cdot 109 + 27 \cdot 109 + 18 = 170188149 \tag{13}$$

$$atvirumas(s[1]) = log_2 10153350000 - log_2 790444808 = 3.68314726 \tag{14}$$

$$atvirumas(s[2]) = log_2 10153350000 - log_2 170188149 = 5.89868142 \tag{15}$$

Pagal rezultatus, galima teigti, kad yra labiau tikėtina, kad bus atskleisti antro žmogaus duomenys. Turint daugiau parametrų, būtų dar mažesnė atvirumo metrikos reikšmė, nes būtų dar mažiau tikėtina, kad bus atkartoti duomenys. Turint kelius modelius, galima lyginti modelių atvirumo metrikas ir spręsti apie modelių saugumą.

3 Modelių palyginimo metodai

3.1 Lyginimas pagal duomenų nuokrypį

Siekiant palyginti, kaip modeliai gerai saugo privačius duomenis, apibrėžiame metriką DMDK (didžiausias modelio duomenų nuokrypis). Kuo metrika yra mažesnė, tuo tiksliau galima nuspėti, kokie duomenys buvo naudojami modelio mokymui. Kuo metrika didesnė, tuo yra sunkiau nuspėti, kokie duomenys buvo naudojami mokymui. Ši metrika leidžia lyginti skirtingus modelius su skirtingais duomenimis. Metrika buvo pasiūlyta specialiai šiam darbui, siekiant palyginti skirtingus modelius.

DMDK yra išvestas lygtyje (16). Prieš DMDK skaičiavimą reikia paimti visus modelio mokymui skirtus duomenis ir kiekvienai duomenų eilutei apskaičiuoti modelio išvestį. Skaičiavimus reikia atlikti tik su tomis eilutėmis, su kuriomis modelis išvedė teisingą atsakymą. Teisingas atsakymas yra laikomas tas, kuris sutampa su atsakymu pradiniuose duomenyse. Pavyzdžiui, su duotais duomenimis modelis grąžina tam tikrą atsakymą, jeigu pradiniuose duomenyse yra nurodyta, kad su šiais parametrais atsakymas yra toks pats, tai reiškia, kad modelis grąžino teisingą atsakymą. Tyrimui reikia imti tik tas duomenų eilutes, su kuriomis buvo grąžinamas teisingas atsakymas, nes šio tyrimo tikslas yra patikrinti, kaip modelis yra prisirišęs prie pradinių duomenų. Prisirišimą testuoja DMDK metrika. Eilutės su neteisingais atsakymais negali būti įtraukiamos į tyrimą, nes prie jų modelis ir taip nėra prisirišęs. Turint tik tas eilutes, su kuriomis modelis išvedė teisingą atsakymą, galima į nelygybę įstatyti kintamuosius. Lygtyje yra naudojami tokie kintamieji: m - duomenų eilučių skaičius, h - parametrų skaičius (stulpeliai), ϵ - ieškomas didžiausias galimas kintamasis, su kuriuo modelis nepakeičia išvesties rezultatų, $D_{eilut.:n,stulp.:k}$ - duomenys n eilutėje ir k stulpelyje, R - realiųjų skaičių aibė.

$$DMDK = \sum_{n=0}^{m}(\sum_{k=0}^{h}(max((|\epsilon| + D_{eilut.:n,stulp.:k}) : \epsilon \in R))/h)/m \tag{16}$$

3.2 Skaičiavimo pavyzdys

Tarkime egzistuoja modelis, kuris turi 2 parametrus: gliukozės kiekis kraujyje ir KMI. Pagal šiuos du parametrus modelis išveda rezultatą: ar žmogus serga cukriniu diabetu ar ne. Paimamos visos mokymo duomenų eilutės, su kuriomis modelis išveda teisingą rezultatą. Tarkime, jos yra aprašytos 1 lentelėje.

Gliukozė	KMI	Išvestis
148	33.6	1
85	26.6	0
183	23.3	1

1 lentelė. Pavyzdiniai duomenys

Taikant jau sukurtą modelį, įvertiname ar keičiasi išvestis didinant kiekvienos eilutės kiekvieno

Gliukozė	KMI
3	2
6	1
4	0.1

2 lentelė. Pavyzdinės epsilon reikšmės

stulpelio reikšmes, pridedant ϵ. Kiekvienai reikšmei yra atskirai didinamas ϵ, kol gaunamas tos eilutės tam tikro stulpelio maksimalus ϵ. Skaičiavimams parodyti, tarkime, kad gautos maksimalios ϵ reikšmės yra pateiktos lentelėje 2.

DMDK skaičiavimas pateiktas lygtyje (17).

$$DMDK = \frac{3+2}{2} + \frac{6+1}{2} + \frac{4+0.1}{2} = 8.05 \tag{17}$$

Apskaičiavus kitų modelių DMDK reikšmes, galima lyginti, kuris modelis efektyviau saugo duomenis, naudotus mokymosi metu. Jeigu kito modelio apskaičiuota DMDK reikšmė būtų didesnė už 8.05, reikštų, kad tas modelis labiau saugo duomenų privatumą.

3.3 Algoritmo verifikavimas

Siekiant pademonstruoti, kad algoritmas teisingai įvertina skirtingus modelius ir galima juos palyginti tarpusavyje, paimkime kelis skirtingus modelių pavyzdžius.

Tarkime, kad pirmas modelis turi viena parametrą - KMI. Pagal šį parametrą, modelis prognozuoja, ar žmogus serga cukriniu diabetu ar ne. Modelio tikslumas yra 54%, jis visą laiką prognozuoja, kad žmogus serga cukriniu diabetu. Skaičiuojant DMDK, reikia pasirinkti tik tuos duomenis, su kuriais buvo išvesta teisinga prognozė. Modelis visą laiką prognozuoja, kad žmogus serga cukriniu diabetu, todėl, nėra tokios reikšmės, kurią pridėjus prie duomenų, pasikeis modelio prognozė. Todėl DMDK reikšmė yra ∞ ir tai reiškia, kad modelis negali būti saugesnis. Kai modelio prognozė visą laiką yra tapati, neįmano atgaminti pradinių duomenų, su kuriais modelis buvo mokomas.

Antras pavyzdinis modelis prognozuoja ar žmogus serga cukriniu diabetu, pagal 2 parametrus: KMI ir gimdymų skaičiumi. Modelio tikslumas yra 72%, modelio DMDK reikšmė yra 0,00134. Sprendžiant pagal DMDK, modelis yra nesaugus ir yra labai priklausomas nuo pradinių duomenų, su kuriais modelis buvo mokomas. Modelio nesaugumui įrodyti, paimkime kelias savo sugalvotas duomenų eilutes ir pabandykime atgaminti pradinius duomenis. Tarkime, kad mūsų sugalvotos duomenų eilutės yra lentelėje 3.

KMI	Gimdymų skaičius
25	3
23	2

3 lentelė. Sugalvotos duomenys reikšmės modelio tikrinimui

Kiekvienai sugalvotai duomenų eilutei, analizuojame kiekvieną stulpelį. Prie stulpelio reikš-

mės pridedame ϵ ir didiname ϵ reikšmę tol, kol pasikeičia modelio rezultatas. Analizuojame kiekvieną stulpelį ir pasirenkame stulpelį su mažiausia ϵ reikšme. Pakeičiame analizuojamos duomenų eilutės stulpelio reikšmę, su kuria gauname mažiausią ϵ reikšmę. Išanalizuokime pirmą duomenų eilutę lentelėje 3. Po pirmos iteracijos, gauname pakeistą duomenų eilutę: KMI - 26, gimdymų skaičius - 3. Atliekame iteracijas tol, kol gaunamų naujų reikšmių skirtumas tampa labai mažas. Atlikus visų duomenų analizę, gauname duomenis, kurie buvo gauti iš sugalvotų duomenų eilučių. Jeigu modelio DMDK yra mažas, šie duomenys turi būti artimi pradiniams duomenims, kurie buvo naudojami modelio mokymui. Atlikus analizę buvo gauti duomenys lentelėje 4. Duomenys yra labai panašūs pradiniams modelio duomenims. Tai reiškia, kad DMDK matavimo algoritmas pasiteisino.

KMI	Gimdymų skaičius
26	1
22	2

4 lentelė. Sugalvotos duomenys reikšmės modelio tikrinimui

Siekiant įsitikinti, kad algoritmas yra teisingas ir neduoda rezultatų, kurie tiesiogiai priklauso nuo duomenų, atlikti du eksperimentai.

1. Skaitant duomenis, jų visų reikšmės yra padalintos iš 10. Sukūrus modelį, apskaičiuota nauja DMDK reikšmė. Ji yra labai artima DMDK reikšmei, kai duomenys nebuvo dalijami iš 10. Tai reiškia, kad algoritmas nėra tiesiogiai priklausomas nuo duomenų, kuriais modelis yra mokomas.

2. Skaitant duomenis, prie jų reikšmių yra pridedamas atsitiktinai sugeneruotas triukšmas - skaičius nuo -1 iki 1. Naujo modelio DMDK reikšmė yra panaši į modelio, prieš duomenų keitimą. Tendencijos išlieka panašios.

4 Homomorfinis šifravimas

4.1 Metodas

Homomorfinis šifravimas, yra šifravimo algoritmų klasė, kuri yra grindžiama principu, leidžiančiu atlikti skaičiavimus su užšifruotais duomenimis, jų neatšifruojant. Homomorfizmas matematikoje yra dviejų vienodo tipo objektų struktūra, kurioje vienas objektas yra kito atspindys. Šie algoritmai yra šiek tiek panašūs į viešo šifravimo algoritmus, kurie yra paplitę tinklalapių apsaugoje. Yra naudojamas viešas ir privatus raktas. Su viešu raktu duomenys yra užšifruojami, o su privačiu, atšifruojami. Momomorfinis šifravimas skiriasi nuo viešo šifravimo tuo, kad naudoja algebrainę sistemą, kuri leidžia atlikti skaičiavimus su užšifruotais duomenimis [CWo13]. Algebrainė sistema yra duomenų rinkinys kartu su savo matematinėmis operacijomis.

Yra trijų tipų homomorfiniai šifrai: dalinai homomorfiniai (PHE), šiek tiek homomorfiniai (SHE) ir pilnai homomorfiniai šifrai (FHE). Dalinai homomorfinės sistemos leidžia atlikti skaičiavimus visiems duomenims, tik su pasirinkta operacija. Tai gali būti arba sudėtis arba daugyba. Operacijos duomenims gali būti atliktos neribotą kartą. Šiek tiek homomorfinės sistemos (SHE) naudoja vieną operaciją iki tam tikro sudėtingumo. Šios operacijos gali būti atlikto ribotą kartų skaičių. Pilnai homomorfinės sistemos leidžia naudoti sudėtį ir daugybą vienu metu, neribotą kartų.

Homomorfinių sistemų saugumas yra paremtas žiedinio mokymo su klaidomis problema ("Ring learning with errors"). Ši problema prašo atgaminti slaptą raktą s, pagal duotas aproksimuotas tiesinias lygtis, su slaptu raktu s [Reg10]. Ši problema nebūtų sunki, jeigu nebūtų aproksimuotų lygčių. Jas išspręsti būtų galima ir su Gauso metodu, tačiau dėl galimos paklaidos, sprendimas yra sunkesnis. Vienas iš efektyviausių sprendimų yra pateiktas A. Blum, A. Kalai, ir H. Wasserman darbe [BKW03]. Pateiktas algoritmui reikia $2^{O(n)}$ duomenų ir laiko. Algoritmas padeda surasti rinkinį tiesinių lygčių S, kurio dydis n, tarp visų galimų lygčių $2^{O(n)}$. Susumavus šias lygtis turi gautis pirma slapto rakto s koordinatė.

4.2 Paillier kriptografija

Pallier kriptosistema yra dalinai homomorfinis šifras, kuris leidžia atlikti homomorfinės sudėties operaciją [Wil15]. Nors palaikoma yra sudėties operacija, norint atlikti veiksmus su užšifruotais duomenimis, reikia naudoti daugybą, nes duomenų užšifruoti duomenys yra pakelti laipsniu. Pallier algoritmas yra pagrįstas problema, kuri skaičiuoja reikšmės likutį [Wil15]. Skaičiavimas pateiktas lygtyje (18).

$$z \equiv y^n \cdot (mod n^2) \tag{18}$$

Turint sudėtinę reikšmę n ir sveiką skaičių z, yra sunku nuspręsti, ar z yra n-oji liekana modulio n^2 ar ne [Wil15].

Prieš duomenų šifravimą, reikia rasti viešą ir privatų raktą.

1. Pasirinkti du didelius pirminius skaičius p ir q, kad bendras didžiausias daliklis tarp $p \cdot q, (p-1), (q-1)$ būtų lygus 1.
2. Apskaičiuoti $n = p \cdot q$
3. Rasti didžiausią bendrą kartotinį skaičių λ, tarp $(p-1), (q-1)$
4. Pasirinkti atsitiktinį sveiką skaičių g.
5. Apskaičiuoti $u = (\frac{(g^{\lambda} mod(n^2))-1}{n})^{-1} mod(n)$
6. Viešas raktas (n, g), privatus raktas (λ, u).

Turint viešą raktą, turinio šifravimui reikia sugalvoti atsitiktinį sveiką skaičių r [Wil15]. Tada galima užšifruoti turinį M pagal formulę (19).

$$c = g^m \cdot r^n mod(n^2) \tag{19}$$

Turinio c iššifravimui naudojama formulė (20).

$$M = (\frac{c^{\lambda mod(n^2)} - 1}{n} \cdot u) mod(n) \tag{20}$$

Minėti skaičiavimai yra vykdomi su teigiamais sveikais skaičiais. Turi būti galima atlikti veiksmus ir su neigiamais skaičiais, nes neuroninių tinklų svoriai ir šališkumai gali būti neigiamos reikšmės. Originalus Paillier algoritmas neapima neigiamų reikšmių, todėl [Reg10] straipsnyje pateiktas skaičiavimo būdas, kuris leidžia atlikti veiksmus ir su neigiamais skaičiais.

- Atšifruojant duomenis, turi būti validi sąlyga (21).

$$2 \cdot sup(|M|) + 1 < N \tag{21}$$

  Čia $sup(.)$ - grąžina didžiausią reikšmę tekste, |M| - absoliuti teksto reikšmė, N - vienas iš viešų raktų.

- Norint užšifruoti neigiamą reikšmę $-m$, reikia laikytis lygybės (22)

$$E(-m) = E(m)^{-1} = E(m)^{-1 mod(N)} \tag{22}$$

  Čia m yra teigiama reikšmė, N - vienas iš viešų raktų.

5 Federuotas mašininis mokymas

5.1 Metodas

Duomenys, skirti mašininio mokymo modelio kūrimui, priklauso N savininkų. Konvenciniai modeliai agreguoja visų savininkų duomenis ir su jais moko modelį. Asmuo kuriantis modelį mato visų savininkų duomenis. Šią problemą sprendžia federuotas mašininis mokymas. Kiekvienas savininkas moko modelį atskirai ir visi duomenys nėra agreguojami. Taip vienas iš duomenų savininkų nemato duomenų, kurie jam nepriklauso [YLCT19].

5.2 Galimos saugumo spragos

Viena iš federuoto mašininio mokymosi modelių spragų yra duomenų nuodijimo ataka [SCD+20]. Kenkėjas naudoja gradientinio nuolydžio algoritmą ir analizuoja modelio grąžinamą optimalų sprendinį. Taip kenkėjas gali gauti modelio gradientų reikšmes ir atskleisti modelio pradinius duomenis. Arba gali būti pakenkiama modelio tikslumui.

5.3 Įrenginių heterogeniškumas

Prie federuoto mokymo tinklo gali būti prijungti labai įvairūs prietaisai. Jų galingumas ir surenkamų duomenų kiekis gali smarkiai skirtis. Gali tinkle atsirasti vienas įrenginys turintis daug išteklių, bet mažai duomenų, o kitas įrenginys su mažais ištekliais ir daug duomenų. Duomenys tarp įrenginių gali būti panašūs. Tarkime, jeigu yra renkami duomenys iš lėktuvų apie skristus maršrutus, neapsimoka analizuoti duomenų iš skirtingų lėktuvų, kurie skrido tuo pačiu maršrutu. Praktikoje, didžiajai daliai įrenginių yra nurodoma apskaičiuoti tam tikrą skaičių epochų. Įrenginiai kurie nespėjo to padaryti laiku yra praleidžiami. Praktikoje dažniausiai nėra atsižvelgiama į duomenų panašumą ir kiek įrenginys jų turi [WLH+21]. Šiai problemai spręsti, straipsnyje [LSZ+20] pasiūlytas duomenų paskirstymo algoritmas.

Straipsnyje [WLH+21] pasiūlyta, kad federuoto mokymo tinkle esantis serveris, modelio kūrimui, neturėtų remtis atsitiktinai pasirinktais įrenginiais. Turi būti atsižvelgiama į jų parametrus. Pasiūlyta naudoti komunikaciją tarp pačių įrenginių. Taip jie tarpusavyje galėtų palyginti ar labai skiriasi jų duomenys ir išskirti tam tikras skirtingas charakteristikas. Pagal šias charakteristikas, pagrindinis serveris turėtų pasirinkti tik tam tikrus įrenginius. Jeigu duomenų charakteristikos tarp įrenginių yra labai panašios, įrenginiai turėtų apsikeisti duomenų skirtumais, pasidalinti bendrais duomenimis ir pagal resursus atitinkamai pasidalinti skaičiavimo kiekiu.

Pateiktas metodas yra skirtas sistemoms, kuriose yra apriboti resursai ir negalima analizuoti visų įrenginių. Serveris gali efektyviau sukurti geresnį bendrą modelį. Vykdant įrenginių analizę, būtų galima bandyti atskirti įrenginius, pagal duomenų skirtumų charakteristikas, kurie yra kenkėjiški ir bando pakenkti tinklui. Tinklo efektyvumas padidėja, tačiau atsiranda saugumo spraga, kai įrenginiai tarpusavyje pradeda lyginti duomenis ir jais apsikeisti. Šios spragos sprendimas yra nau-

doti homomorfinį šifravimą kiekviename įrenginyje. Jeigu tinkle atsirastų kenkėjiškas įrenginys ir jis sugebėtų surasti su juo komunikuojančių įrenginių homomorfinio šifro slaptą raktą, jis negalėtų perimti viso tinklo duomenų, tik dalį jų.

2017 metais buvo pasiūlytas labiau praktikoje panaudojamas ir saugesnis "FedAvg" algoritmas [LSZ$^+$20]. Tai yra iteratyvus algoritmas, kuris padeda išspręsti įrenginių heterogeniškumo problemą. Kiekvienos iteracijos metu, algoritmas atlieka E stochastinio gradientinio nuolydžio metodo epochų su K įrenginiais. Įrenginiai komunikuoja su centriniu serveriu, kuriame jų įverčiai yra suvidurkinami. Naujesnis pasiūlytas algoritmas yra "FedProx". "FedProx" yra panaūs į "FedAvg", nes pagrindiniai principai išlieka tokie patys: atliekami lokalūs modelių atnaujinimai, pasirenkama tik nedidelė dalis įrenginių ir centrinis serveris suvidurkina rezultatus. Pagrindinis "FedProx" skirtumas yra kiekvienam įrenginiui skiriamas epochų skaičius. Šis skaičiui kiekvienam įrenginiui yra dinamiškai apskaičiuojamas pagal įrenginio turimų duomenų kiekį ir turimų išteklių dydį. "FedProx" metode yra įrenginių išteklių heterogeniškumas, nes lėti įrenginiai tik dalinai apskaičiuoja savo duomenis ir juos pateikia pagrindiniam serveriui. Toks metodas yra efektyvesnis už "FedAvg", nes "FedAvg" turi didesnį duomenų heterogeniškumą. Silpni įrenginiai su svarbiais duomenimis yra praleidžiami ir centrinis serveris neatsižvelgia į šiuos duomenis visai. Duomenų heterogeniškumui sumažinti, "FexProx" metodas naudoja aproksimuotus kintamuosius. Pridedant šiuos kintamuosius, gali būti sumažinama lokalių modelio atnaujinimų įtaka ir sumažinamas bendras heterogeniškumas. Vietoj funkcijos F optimizavimo, metodas optimizuoja tikslo funkciją h, kuri pateikta lygtyje (23).

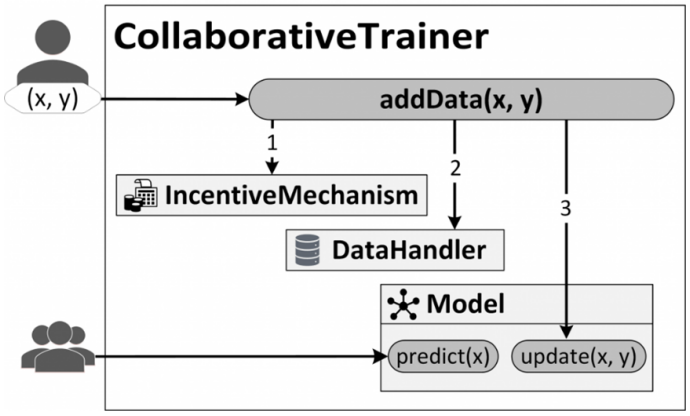
$$minh_k(w_1; w_2) = F_k(w_1) + \frac{\mu}{2}||w_1 - w_2||^2 \tag{23}$$

Čia k - įrenginių skaičius, F_k - tikslo funkcija, $\mu \in [0; 1]$. Kuo μ yra mažesnis, tuo tikslesnis centrinio serverio tikslumas gaunasi.

6 Blokų-grandinių technologija

Dažna mašininio mokymosi modelių problema yra riboti resursai, centralizuotumas, duomenų rinkimas ir tikimybė, kad laikui bėgant, duomenys taps pasenę ir nebenaudingi. Šią problemą sprendžią Microsoft įmonė, pasitelkdama blokų-grandinių technologiją [Har19]. Minėta technologija leidžia vartotojams praplėsti duomenų rinkinį ir tobulinti modelį.

Vieši modeliai yra laikomi išmaniusiose kontraktuose. Jie užtikrina, kad kodas esantis blokų-grandinių tinkle atitinka reikalavimus [Har19]. Išmanieji kontraktai yra protokolai, kurie užtikrina susitarimo tarp vartotojo ir tiekėjo teisingumą. Jie yra automatiškai blokų-grandinių tinkle automatiškai vykdomi ir leidžia kiekvieną transakciją atsekti, bei padaro ją grįžtama [Fra20]. Šiuos kontraktus vertina daug vartotojų ir juos yra sunku modifikuoti. Esant išmaniesiems kontraktas, tinkle galima atsekti, kurie pateikti nauji duomenys yra geri, ir galima vartotoją apdovanoti.



2 pav. Blokų-grandinių technologijų pritaikymas [Har19]

Duomenų pridėjimas prie tinklo pavaizduotas pav. 2. Procesas susideda iš trijų žingsnių [Har19]:

- Iniciatyvumo kūrimo mechanizmas. Jis skatina vartotojus teikti tik teisingus duomenis. Šiame žingsnyje pateikti duomenys yra validuojami, kaip transakcija. Iniciatyvumui gerinti yra daug būtų, vienas iš jų gali būti depozito sistema. Vartotojas pateikdamas duomenis tinklui, sumoka depozitą. Jeigu duomenys yra klaidinantys, depozitas nėra grąžinamas. Jeigu teisingi, depozitas grąžinamas ir, skatinimo tikslams, pridedama tam tikra suma. Duomenų teisingumas yra vertinamas pagal tai, kaip keičiasi modelio tikslumas su duotais duomenimis. Atitinkamai pagal modelio tikslumo pagerėjimą galima proporcingai vartotojui grąžinti daugiau pinigų.

- Duomenys yra siunčiami į blokų-grandinių tinklą ir ten pridedami. Pridėjus duomenis, visi vartotojai juos mato.

- Mašininio mokymosi modelis yra atnaujinamas pagal duomenis tinkle.

Esamas suprojektuotas karkasas geriausiai veikia tik nedideliems modeliams [Har19]. Kol kas laukiama daugiau pasiūlymų, kaip būtų galima patobulinti blokų-grandinių technologijas, kad

modelio mokymo procesas būtų efektyvesnis. Šiuo momentu pasiūlytas metodas neatsižvelgia į įrenginių ir duomenų heterogeniškumą, nėra efektyvaus mechanizmo nusakančio kaip tiksliai reikėtų apmokyti modelį su naujais duomenimis. Esantis sprendimas moko modelį su visais tinkle esančiais duomenimis.

6.1 Privatumo išsaugojimas

Pasiūlytas įmonės Microsoft sprendimas nėra saugus, nes kiekvieno vartotojo pateikti duomenys yra viešai matomi. Tačiau užtenka naudoti 4 skyriuje minimus homomorfinio šifravimo algoritmus ir problema yra išsprendžiama. Modelio saugumas tokiu atveju priklausytų tik nuo pasirinkto šifravimo algoritmo saugumo.

Lyginant su federuotu mokymosi algoritmu, pateiktas blokų-grandinių sprendimas išsprendžia kenkėjiškų įrenginių problemą. Pasiūlytame sprendime, prieš priimant vartotojo duomenis, jie yra įvertinami. Tačiau duomenų įvertinimui, į blokų-grandinių tinklą yra pridedama 10% duomenų, validumo tikrinimo tikslams.

Kadangi yra naudojami išmanieji kontraktai, šiuos duomenis galima bandyti iškarto pašalinti iš tinklo. Jeigu kenkėjiškas vartotojas bandytų apkrauti tinklą blogais duomenimis, jam tai daug kainuotų, nes kiekvieno tikrinimo atveju, jis neatgautų sumokėto depozito. heterogeniškumo problema sprendime nėra sprendžiama ir duomenys yra lygiavertiškai imami iš visų įrenginių ir kiekvienas įrenginys gali vienodai prisidėti prie modelio kūrimo. Kadangi heterogeniškumas nėra tikrinimas, tinkle gali atsirasti daug duomenų, kurie turi labai mažą įtaką modelio tikslumui. Tai kainuoja daug skaičiavimo resursų modelio kūrimui.

Sprendžiant, kurį metodą pasirinkti, ar federuoto mokymosi, ar blokų-grandinių, reikia atsižvelgti į būsimo tinklo planuojamą dydį. Jeigu planuojama turėti daug įrenginių tinkle, blokų grandinių metodas yra geresnis, nes jis apsaugo nuo kenkėjiškų įrenginių. Jeigu tokių įrenginių būtų daug, iš surenkamo depozito būtų galima atitinkamai apdovanoti teigiamai tinklą paveikiančius naudotojus. Tačiau kadangi nėra atsižvelgiama į įrenginių ir duomenų galimą heterogeniškumą, prarandamas didelė dalis tinklo apmokymo efektyvumo. Jeigu vartotojų bus mažai blokų-grandinių tinkle, atsiranda rizika, jog tinklas nebus toks saugus ir ne visada jis bus prieinamas visiems vartotojams. Jeigu planuojama turėti nedaug vartotojų tinkle, federuotas mokymasis yra geresnis variantas. Kadangi yra vienas centralizuotas serveris, nėra problemų dėl tinklo pasiekiamumo, esant mažam skaičiui vartotojų. Analizuojant įrenginių ir duomenų heterogeniškumą, tinklas yra efektyvesnis apmokant modelį.

7 Saugus skirtingų pusių skaičiavimas

7.1 Apibrėžimas

Saugus skirtingų pusių skaičiavimo algoritmas („Secure Multiparty Computation“, trumpinys MPC) leidžia keliems įrenginiams skaičiuoti bendros funkcijos rezultatą, nesidalinant duomenimis tarpusavyje [Lin20]. Pagrindinis skirtumas, kuo skiriasi MPC nuo kitų paskirstyto skaičiavimo algoritmų, yra didelis dėmesys saugumui. Kuriant MPC protokolą skaičiavimams atlikti, yra du reikalavimai: privatumas ir teisingumas. Privatumo reikalavimas reikalauja, kad įrenginiai, kurie vykdys skaičiavimus, sužinotų tik apskaičiuotą rezultatą ir nieko daugiau. Vienintelė papildomai sužinota informacija gali būti gauta tik išvedus ją iš gauto rezultato. Teisingumo reikalavimas reikalauja, kad kiekviena šalis turi gauti teisingai apskaičiuotą reikšmę. Jeigu vienas įrenginys apskaičiuos rezultatą klaidingai, visų įrenginių bendras skaičiavimo rezultatas bus irgi neteisingas. Paskirstyto skaičiavimo algoritmai irgi duotų neteisingą reikšmę, tačiau visų įrenginių bendras rezultatas nenukryptų daug.

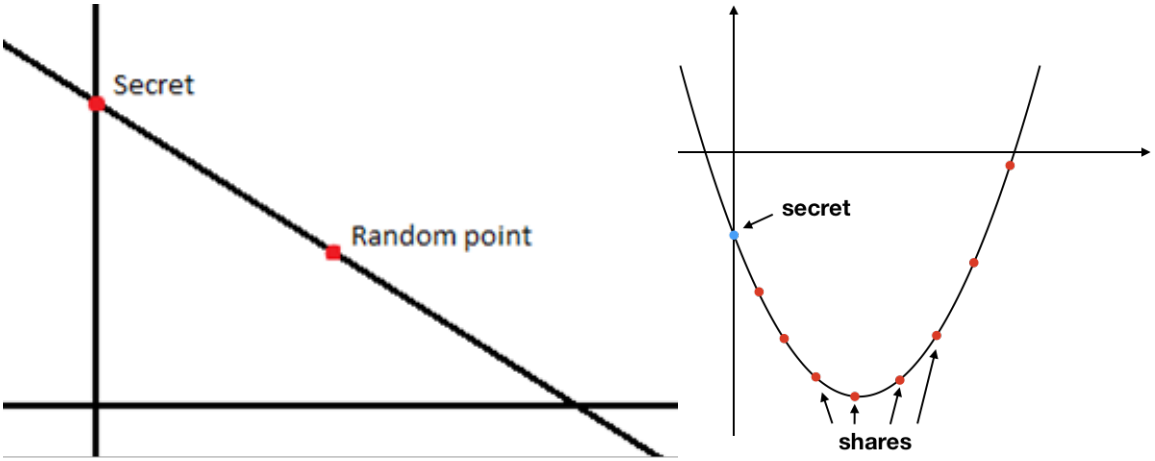
Per laiką atsirado MPC protokolo patobulinimų, norint apsaugoti algoritmą nuo kenkėjiškų įrenginių, ir dėl to sukurti dar trys reikalavimai: įvesties duomenų nepriklausomumas, garantuotas reikšmės grąžinimas ir lygumas. Įvesties duomenų nepriklausomumo reikalavimas reikalauja, kad kiekvienas įrenginys nieko nežinotų apie kitų įrenginių naudojamus duomenis ir atitinkamai nekeistų savo duomenų. Garantuoto reikšmės grąžinimo reikalavimas reikalauja, kad kenkėjiški įrenginiai negalėtų sutrikdyti kitų įrenginių skaičiavimų. Pavyzdžiui, jeigu skaičiavimams naudojamas interneto ryšys, negali būti sudaryta galimybė leisti kenkėjiškiems įrenginiams paveikti interneto ryšio. Lygumo reikalavimas reikalauja, kad jeigu įrenginys turi blogus duomenis ar gautas rezultatas yra sugadintas, šis įrenginys turi gauti savo apskaičiuotą reikšmę tik tokiu atveju, jeigu visų kitų įrenginių rezultatai nėra sugadinti.

7.2 Skaičiavimo algoritmai

MPC algoritmui paaiškinti, pirma reikia išsianalizuot atvejį, kai yra du įrenginiai su savo slaptais raktais ir viena paslaptis. Jeigu norime atgaminti paslaptį, reikia turėti abiejų įrenginių slaptus raktus ir taip galime atgaminti paslaptį. Pavyzdžiui, jeigu paslaptis yra skaičius 654321, tada reikia sugeneruoti atsitiktinį skaičių, kuris yra mažesnis už 654321. Tarkime tai yra 123456. Šis naujas atsitiktinis skaičiuos yra pirmo įrenginio slaptas raktas. Norint sukurti antro įrenginio raktą, galime iš slapto rakto atimti pirmo įrenginio raktą ir gauname 530865. Norint turėti daugiau įrenginių, kyla problema, kad siekiant atgaminti paslaptį, reikia visų įrenginių sutikimo. Jeigu vienas įrenginys bus prarastas, dings paslaptis. Tai nėra priimtinas sprendimas, todėl buvo sugalvotas algoritmas, kuris leidžia atgaminti paslaptį, turint tik dalį rakto dalių [Raf18]. Šis algoritmas vadinasi Shamir slapto rakto dalijimosi schema („Shamir‘s secret key sharing schema“), kitaip žinomas, kaip rakto skaldymo algoritmas. Slaptas raktas yra padalijamas yra kelias dalis. Kiekviena dalis pavieniui yra nepanaudojama. Slapto rakto atstatymui iš padalintų dalių, nebūtina turėti visas dalis. Pavyzdžiui,

jeigu slaptas raktas yra padalijamas į penkias dalis, gali užtekti turėti tik tris dalis ir iš jų galima jau atstatyti pradinį slaptą raktą.

Shamir pasiūlė skaičiuoti slapto rakto dalis grafiškai 2D erdvėje [Raf18]. Paslaptis gali būti laikoma Y koordinate pradinio taško, kurio X koordinatė yra 0. Tada pasirenkame atsitiktinį tašką grafike ir brėžiame liniją tarp slapto taško ir atsitiktinai sugeneruoto taško. Tai grafiškai pavaidzuota pav. 3a Ši linija yra slapta ir ją žinant, galima atgaminti pradinę paslaptį. Kiekviena rakto dalis yra atsitiktinis taškas slaptoje linijoje. Turint dvi dalis, galima atgaminti slaptą raktą.



(a) Slapto rakto dalijimas tiesėje (b) Slapto rakto dalijimas parabolėje [Cho19]

3 pav. Slapto rakto dalijimas

Norint leisti, kad paslaptį būtų galima atskleisti su dviem raktais, reikia sugeneruoti du atsitiktinius taškus ir nubrėžti slaptą parabolę [Raf18]. Turint tris sugeneruotus raktus, galima atkurti pradinę parabolės lygtį ir atkurti paslaptį. Rakto dalijimas grafiškai pavaizduotas pav. 3b Norint, kad paslaptis būtų atgaminta iš keturių raktų, reikia sugeneruoti kubinę kreivę ir taip toliau.

8 Bendro įspūdžio privatus agregavimas

Bendro įspūdžio privatus agregavimas, angliškai "Private Aggregation of Teacher Ensembles" (PATE), leidžia skirtingiems įrenginiams prisidėti prie bendro mašininio mokymosi modelių tinklo. Tinklą sudaro daug skirtingų mašininio mokymosi modelių, kurie nepriklausomai priima sprendimus. Kiekvieno modelio sprendimą tinklas susumuoja, pritaiko Gauso filtrą ir galutinį rezultatą pateikia vartotojui [PG18]. Kiekvienas įrenginys turi galimybę pasirinkti savo mašininio mokymosi algoritmą ir realizuoti savo būdu, nepriklausomai nuo kitų įrenginių ar tinklo. Ši galimybė padaro tinklą efektyvesniu, nes įrenginiai gali pasirinkti algoritmus, kurie yra efektyvesni už tinko kūrėjo parinktą algoritmą.

Prieš kiekvienam įrenginiui pradedant kurti modelį, yra paskirstomi pagrindinio serverio privatūs duomenys dalimis [PG18]. Kiekviena duomenų dalis neturi tokių įrašų, kurie kartojasi kitose dalyse. Kiekvienoje paskirstytoje dalyje kiekvienas įrašas yra unikalus. Jeigu su duotais duomenimis, tinklas išvedė teigiamą atsakymą, atsakyme negali būti atskleista jokia informacija apie pradinius privačius duomenis. Tai yra todėl, nes jeigu du modeliai priima bendrą sprendimą apie duomenis ir tik vienas modelis galėjo būti apmokytas su tokiais duomenimis, antro modelio išvestis yra nepriklausoma nuo jo pradinių duomenų.

Vienintelis būdas gauti informacijos apie modelių pradinius duomenis yra analizuoti bendrą tinklo išvestį ir tikrinti kiek kokių modelių ką išvedė. Dėl šios priežasties, PATE tinklas naudoja Gauso filtrą, prieš pateikiant galutinę išvestį [PG18]. Gauso filtras paprastai yra naudojamas vaizdų suliejimui, detalių panaikinimui [Pat10]. Modelių "balsų" skaičiavimui, naudojamas Gauso 1D filtras. Gauso filtro skaičiavimas pateiktas lygtyje (24).

$$G(x) = \frac{1}{\sqrt{2 \cdot \pi \cdot \sigma^2}} e^{-\frac{x^2}{2 \cdot \sigma^2}} \tag{24}$$

Čia σ yra standartinis skirstinio nuokrypis.

Dabar PATE tinkle atsiranda dvi problemos [PG18]. Pirma problema atsiranda, kai vykdoma daug užklausų ir jos yra panašios. Tokiu atveju, galima atgaminti informaciją apie pradinius duomenis analizuojant tinklo skirtingas išvestis su panašiais duomenimis. Dėl šios problemos reikėtų riboti galimą užklausų skaičių kiekvienam tinklo naudotojui. Skaičiui pasiekus ribą, reikia gauti naujus duomenis ir sukurti naujus modelius su šiais duomenimis. Antra, tinklas turi daug skirtingų modelių ir dėl to negalima paviešinti tinklo, kaip vieno modelio. Taip yra daroma, norint apsisaugoti nuo pradinių parametrų atgaminimo. Dėl šių problemų, PATE tinklas turi papildomą žingsnį: studento modelio kūrimas. Studento modelis yra apmokomas, perkeliant visų sukurtų modelių tinkle parametrus į vieną. Tai yra atliekama naudojant viešus nesurūšiuotus duomenis ir juos leidžiant pro tinklą. Susumavus visų tinklo modelių "balsus", jie yra filtruojami su Gauso filtru. Galutinis rezultatas yra naudojamas mokant studento modelį. Šis studento modelis yra galutinis PATE tinklo rezultatas.

PATE galima patobulinti keliais būdais:

- Serveris, kuris paskirsto duomenis į dalis, turėtų naudoti homomorfinį šifravimą, prieš atiduodant duomenis kitiems įrenginiams. Tai netik paslėptų duomenis nuo modelių kūrėjų ir pašalintų duomenų nutekėjimo riziką, bet ir padarytu pačius modelius ir visą tinklą saugesnį.

- Kuriant studento modelį, turi būti atliekama kiek galima mažiau užklausų į bendrą modelių tinklą [Fra20]. Pavyzdžiui, užklausos turi būti atliekamos tik iki tol, kol pasiekiamas studento modelio tam tikras tikslumas. Taip pat užklausos turi būti atliekamos su nepanašiais duomenimis. Kuo daugiau yra atliekama užklausų į bendrą modelių tinklą ir kuo jos yra panašesnės, tuo labiau yra tikėtina, kad bus galima atskleisti pradinius modelių duomenis.

- Norint turėti optimalų privatumo išsaugojimą, tinkle esantys modeliai turėtų grąžinti panašius rezultatus [Fra20]. Jeigu keli modeliai išsiskiria rezultatais nuo kitų modelių tinkle, atsiranda tikimybė, kad bus atskleisti pradiniai modelių duomenys. Tai gali nutikti kaip bus siunčiama daug panašių užklausų ir gausis skirtingi rezultatai. Panašus eksperimentas, kuris veiktų ir su PATE tinklu, siekiant atskleisti pradinius duomenis, yra atliktas 3.3 skyriuje. Pavyzdyje yra naudojami kiti algoritmai, tačiau PATE tinklo pažeidžiamumo esmė yra tapati.

Lyginant PATE algoritmą su federuotu mokymu, PATE yra saugesnis, nes tinkle modelių yra nors keli ir jeigu duoti duomenys yra panašūs į vieno modelio pradinius duomenis, esant bendram modelių sutarimui, sunku atgaminti pradinius duomenis, iš tinklo išvesties. Pradinių duomenų atgaminimą taip pat apsunkina ir Gauso filtras. Optimaliu atveju, privatumas tinkle yra geriausiai išsaugomas, kai kiekvienas modelis grąžina panašius rezultatus su duotais viešais duomenimis, tada negalima atsekti kuris modelis tinkle yra už tai atsakingas. PATE yra taip pat ir efektyvesnis už federuotą mokymą, nes modelius kuria įrenginiai, kurie gali pasirinkti tam tikrai duomenų sričiai optimalius algoritmus, o federuotas mokymas apibrėžia tik vienodų algoritmų naudojimą. PATE taip pat išsprendžia įrenginių ir duomenų heterogeniškumo problemas. Vienintelis federuoto mokymosi pranašumas yra, kad nereikia dalintis pradiniais duomenimis. Jeigu jie yra svarbūs, o įrenginiai nėra patikimi, atsiranda rizika, kad įrenginiai gali atskleisti duomenis. Tačiau ši problema gali būti išspręsta su 4 skyriuje minimais šifravimo algoritmais, kurie leistų paslėpti duomenis nuo pačių įrenginių.

9 Tensorflow karkasas

“Tensorflow” karkasas - Google technologijų įmonės atviro kodo, mašininio mokymosi karkasas. Šiame karkase yra realizuota daug algoritmų ir mašininio mokymosi modelių. 2019 metais kovo mėnesį “Tensorflow” karkaso puslapyje buvo pristatytas straipsnis apie diferencinį privatumą. Prie “Tensorflow” karkaso buvo pridėta Tensorflow Privacy biblioteka, kuri realizuoja mašininio mokymosi modelius, naudojančius diferencinį privatumą [RE19]. Iki to laiko, “Tensorflow” karkasas neatsižvelgdavo į duomenų išsaugojimą. 4 skyriuje aprašytus homomorfinio šifravimo algoritmus nebuvo galima taikyti karkase. Vienintelis būdas išsaugoti privatumą buvo naudoti skirtingų mašininių modelių tinklą, pavyzdžiui 5 skyriuje aprašytą federuotą mašininį mokymąsi arba 8 skyriuje aprašytą bendro įspūdžio agregavimo algoritmą.

Atsitiktinai parinktos reikšmės $M : D \rightarrow R$ tenkina diferencinio privatumo sąlygą, kai du duomenų rinkiniai $X ir X' \in D$, išvestys $\Upsilon \in R$ ir tenkinama sąlyga lygybėje 25 [MAE+19]. Čia $Pr[X]$ - tikimybė, kad įvyks įvykis X. Ši sąlyga užtikrina, kad parinkti du duomenų rinkiniai X ir X', kurie skiriasi tik vienu įrašu, duos labai panašius rezultatus, kurių neis atskirti. Šioje lygtyje ϵ yra vadinamas privatumo biudžetu - kuo mažesnis yra ϵ, tuo gaunamas didesnis duomenų saugumas.

$$Pr[M(X) \in \Upsilon] \leq e^{\epsilon} Pr[M(X') \in \Upsilon] + \delta \tag{25}$$

Pagrindinė pasiūlyto algoritmo idėja - vykdant stochastinio gradiento nuolydžio algoritmą, modifikuoti gradientų reikšmes. Tai yra atliekama su dviem algoritmo pakeitimais:

- Kiekvieno gradiento jautrumas pokyčiams turi būti apribotas. Reikia nurodyti, kiek kiekviena duomenų eilutė gali paveikti kiekvieną gradientą. Šiam tikslui yra taikoma cenzūravimo procedūra “Clipping”, kuri nusako maksimalų galimą gradientų pokytį.
- Siekiant išsaugoti privatumą, pridedamas triukšmas prie reikšmės, gautos iš cenzūravimo metodo. Triukšmas yra gaunamas su Gauso filtru. Taikant filtrą nustatomos ir apribojamos reikšmės kiekvienam gradientui atsitiktinai, taip išsaugomas privatumas

“Tensorflow Privacy” leidžia saugumo prasme optimizuoti modelį, nekuriant jo iš naujo ir nekeičiant pradinių modelio parametrų. Tai yra verslui geras sprendimas, nes kuriant sprendimą yra daugiausiai skiriama dėmesio į modelio tikslumą ir mažai investuojama laiko galvojant apie modelio saugumą. Šį funkcionalumą atlieka du paketai esantys “Tensorflow Privacy” karkase. Vienas iš jų yra “Bolt-on” paketas, kuris modifikuoja sukurtą modelį prieš jo naudojimą gamybinėje aplinkoje. Jis prie kiekvieno modelio svorio prideda triukšmo.

10 Tyrimas

10.1 Modelių palyginimas

Tyrimo tikslams, buvo sukurti modeliai ir, skaičiuojant DMDK metriką, jie lyginami tarpusavyje. Tyrimui paimti diabetu sergančių žmonių duomenys iš "Sklear" bibliotekos viešai prieinami duomenys. Gauti rezultatai pateikti lentelėje 5.

Modelis	Nuostolių f-jos reikšmė (MSE)	Tikslumas	DMDK
PyTorch neuroninis tinklas	103.9837813	81.1%	87.419
Gradientinio nuolydžio algoritmas	2272.06	63.78%	0.051792
Pallier homomorfiniu šifravimu grįstas modelis	2185.37	63.52%	0.1416

5 lentelė. Sugalvotos duomenys reikšmės modelio tikrinimui

Pagal gautus rezultatus, pateiktus lentelėje 5, gautos išvados:

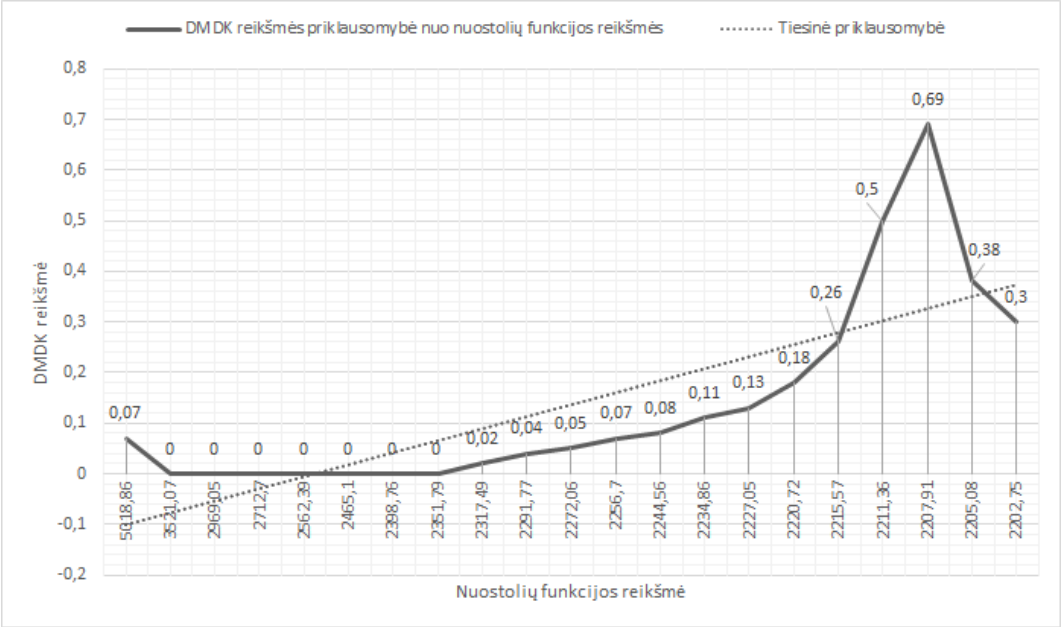
- PyTorch neuroninis tinklas greičiau artėja prie nuostolių funkcijos minimumo, nei kiti modeliai. Esant 70% modelių tikslumui, PyTorch modelis yra saugiausias. Mažiausiai saugus yra paprastu gradientiniu nuolydžiu grįstas modelis.
- Palliers kriptografija grįstas modelis yra saugesnis už paprastą gradientinio nuolydžio metodą.
- Didėjant PyTorch modelio tikslumui, jis pradeda labiau prisirišti prie duomenų ir vidutinė galima maksimalaus nuokrypio reikšmė smarkiai krenta – modelis tampa vis mažiau saugus. Jeigu uždavinio tikslas yra sukurti labai tikslų modelį, vertėtų apgalvoti ar Pallier sistema grįstas modelis nėra geriau.
- Su visais modeliais, išskyrus PyTorch neuroninį tinklą, didėjant modelio tikslumui, didėja vidutinė maksimalaus nuokrypio reikšmė – modelis tampa saugesnis, o su PyTorch neuroniniu tinklu, mažėja - modelis tampa mažiau saugus.

Tas pats tyrimas buvo atliktas su MNIST rašytinių skaitmenų nuotraukomis. Tik PyTorch neuroninis tinklas su vienu paslėptu sluoksniu ir VGG16 konvergavo minimumo link. Pasiekus 74% ir 82% tikslumus, PyTorch neuroninio tinklo DMDK reikšmės išlieka vienoda - 0.04999, MSE nuostolių funkcijos reikšmės atitinkamai 266.66730 ir 203.6602. VGG16 tinklui pasiekus 72% tikslumą, MSE nuostolių funkcijos reikšmė - 2.59, DMDK reikšmė - 0.17. Reiškia VGG16 modelis yra saugesnis už paprastą vieno paslėpto sluoksnio PyTorch neuroninį tinklą.

Kiekvienam modeliui atliktas atskiras nuostolių funkcijos (MSE), DMDK rodiklių ir modelio tikslumo, DMDK priklausomybės tyrimas.

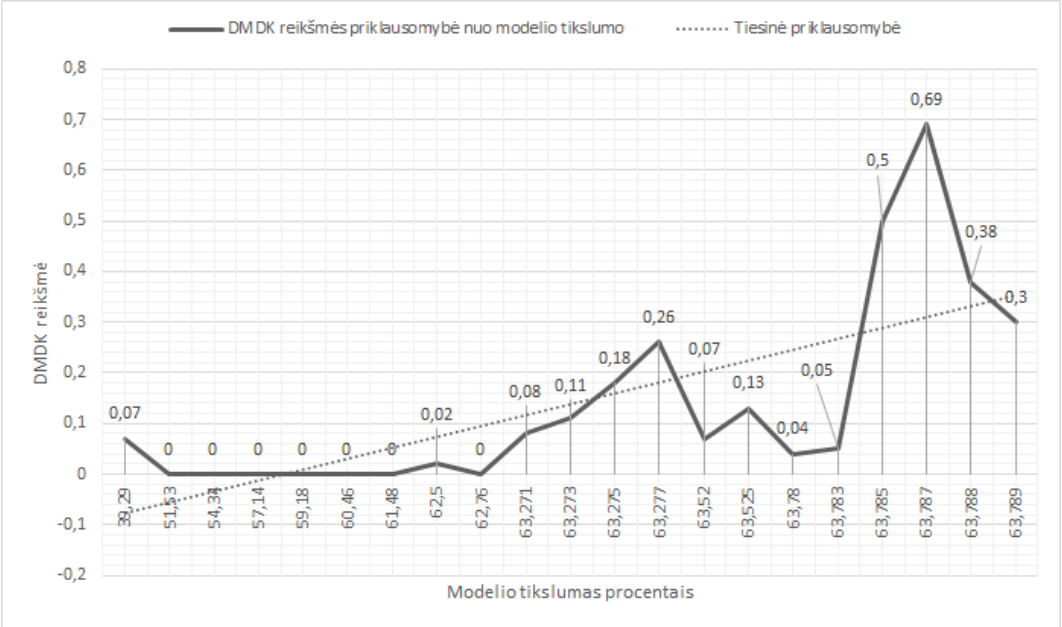
10.1.1 Pallier homomorfiniu šifravimu grįsto modelio detalus tyrimas

DMDK rodiklio reikšmių priklausomybė nuo MSE nuostolių funkcijos reikšmių pateikta pav. 5. Čia X ašis - MSE nuotolių funkcijos reikšmės, Y ašis - DMDK reikšmės.



4 pav. Palliers DMDK rodiklio reikšmių priklausomybė nuo MSE nuostolių funkcijos reikšmių

DMDK rodiklio reikšmių priklausomybė nuo modelio tikslumo pateikta pav. 5. Čia X ašis - modelio tikslumas procentais, Y ašis - DMDK reikšmės.

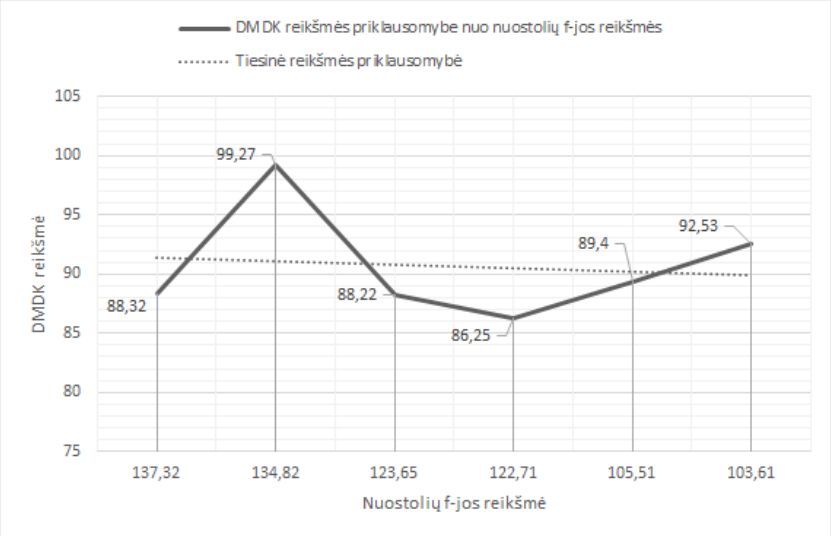


5 pav. Palliers DMDK rodiklio reikšmių priklausomybė nuo MSE nuostolių funkcijos reikšmių

Iš atlikto tyrimo matosi koreliacija tarp modelio nuostolių funkcijos reikšmių ir DMDK reikšmių. Kuo nuostolių funkcijos reikšmė yra mažesnė, tuo DMDK yra didesnis ir modelis yra laikomas saugesnis. Vietos, kuriose DMDK yra lygus 0, modelis yra per daug prisirišęs prie parametrų. DMDK tiesioginės priklausomybės nuo modelio tikslumo, neturi.

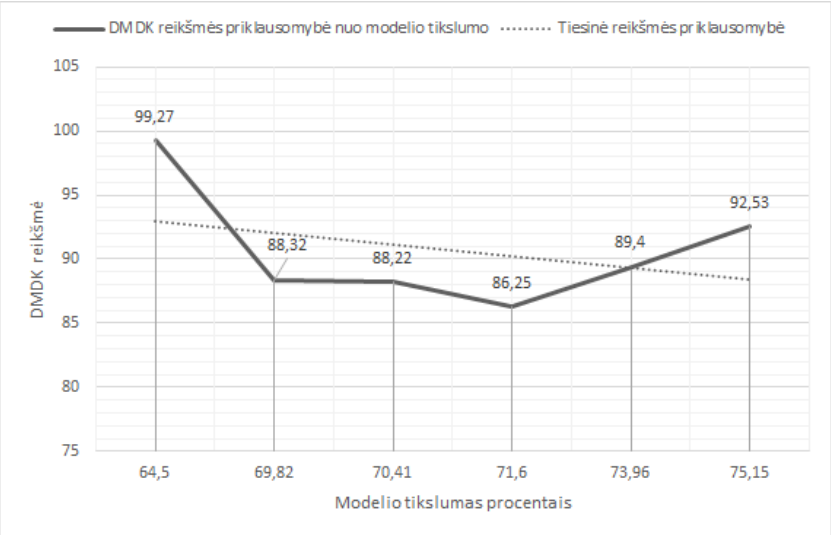
10.1.2 PyTorch neuroninio tinklo detalus tyrimas

DMDK rodiklio reikšmių priklausomybė nuo MSE nuostolių funkcijos reikšmių pateikta pav. 7. Čia X ašis - MSE nuotolių funkcijos reikšmės, Y ašis - DMDK reikšmės.



6 pav. PyTorch neuroninio tinklo DMDK rodiklio reikšmių priklausomybė nuo MSE nuostolių funkcijos reikšmių

DMDK rodiklio reikšmių priklausomybė nuo modelio tikslumo pateikta pav. 7. Čia X ašis - modelio tikslumas procentais, Y ašis - DMDK reikšmės.



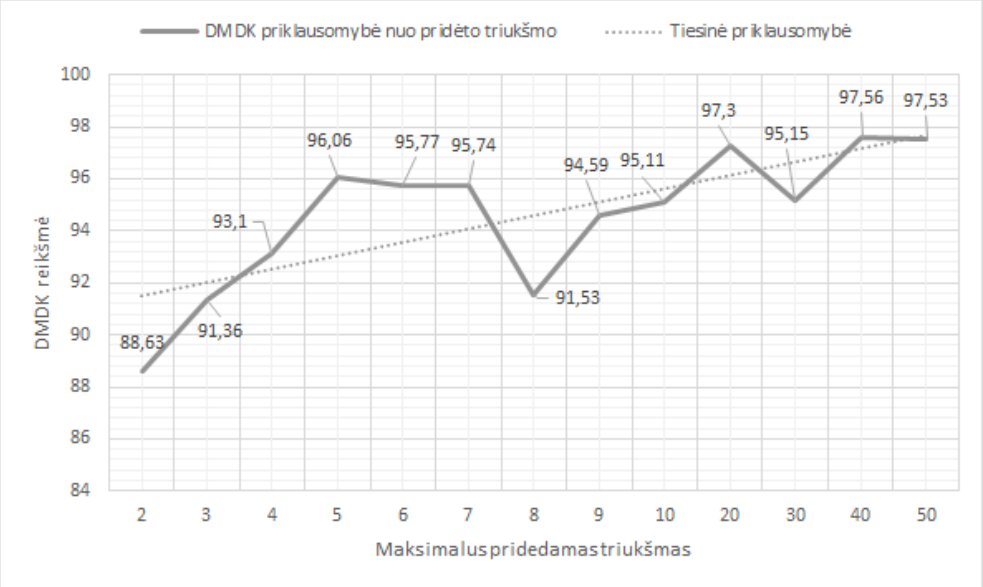
7 pav. PyTorch neuroninio tinklo DMDK rodiklio reikšmių priklausomybė nuo MSE nuostolių funkcijos reikšmių

Iš atlikto tyrimo nesimato jokios stiprios koreliacijos tarp DMDK rodiklio ir MSE nuostolių funkcijos reikšmių ar modelio tikslumo.

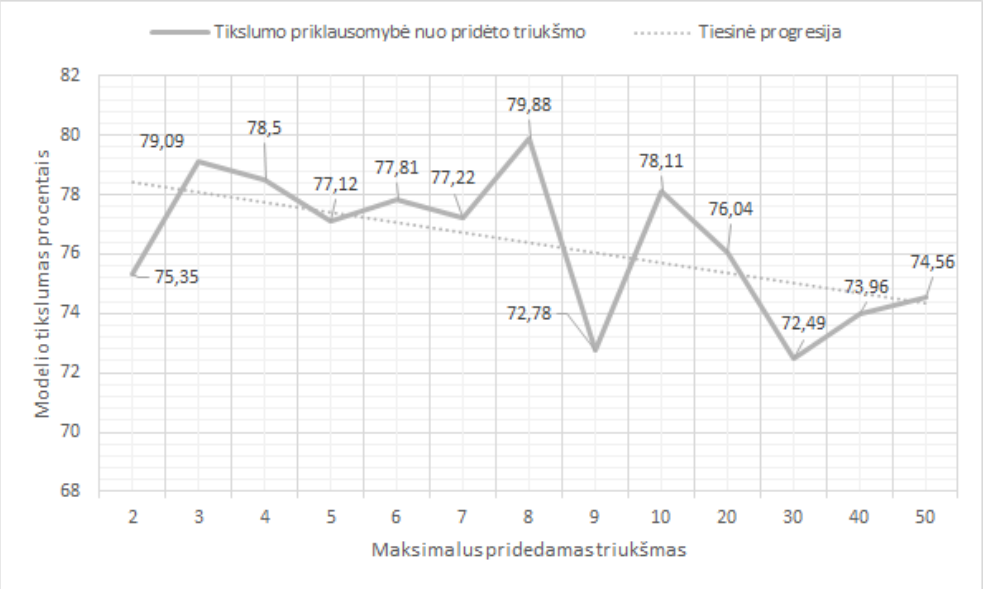
10.2 Duomenų triukšmo įtakos tyrimas

Atliktas tyrimas su PyTorch 3-jų sluoksnių neuroniniu tinklu ir duomenų triukšmu. Modelis kiekvieno bandymo metu atlieka 250 iteracijų. Kiekvieną kartą kai yra nuskaitomi duomenys, pridedamas atsitiktinis triukšmas, kuris neviršija nurodytos reikšmės. Tyrime buvo keičiamas maksimalus generuojamas duomenų triukšmas ir stebima, kokią įtaką tai turi sukurtam modeliui.

Tyrime pastebėta, kad tarp triukšmo ir modelio duomenų saugumo rodiklio DMDK yra koreliacija, kuri matoma pav. 8. Didinant atsitiktinį duomenų triukšmą, gerėja modelio duomenų saugumas. Pastebėta taip pat ir koreliacija tarp modelio tikslumo ir pridėto triukšmo prie duomenų. Pav. 9 parodyta, kad didinant modelio duomenų triukšmą, mažėja modelio tikslumas.



8 pav. DMDK priklausomybė nuo pridėto triukšmo



9 pav. Modelio tikslumo priklausomybė nuo pridėto triukšmo

10.3 Duomenų kiekio ir DMDK tyrimas

Buvo atliktas bandymas generuoti duomenis ir tikrinti, kaip keičiasi DMDK reikšmė kintant duomenų kiekiui. Tyrimas buvo atliekant generuojant duomenis panašius į esančius Sklearn žmonių sergančiu cukriniu diabetu, duomenų rinkinyje. Tiriant nebuvo rasta koreliacija tarp sugeneruoto duomenų kiekio ir DMDK reikšmės. Tai reiškia, kad didinant pradinių duomenų kiekį, modelio saugumas nekinta.

10.4 Tyrimo išvados

- Lyginant PyTorch neuroninį tinklą, paprastą gradientinio nuolydžio algoritmą ir Pallier homomorfiniu šifravimu grįstą modelį, jeigu visi minėti modeliai vykdant tyrimą konverguoja minimumo link, pats saugiausias modelis yra PyTorch neuroninis modelis. Mažiausiai duomenis saugantis yra paprastas gradientinis metodas.
- Naudojant Pallier homomorfiniu šifravimu grįstą modelį, egzistuoja koreliacija tarp modelio nuostolių funkcijos reikšmės ir DMDK rodiklio. Vidutiniškai, kuo nuostolių funkcijos reikšmė yra mažesnė, tuo DMDK rodiklis yra didesnis ir modelis yra saugesnis.
- Pallier homomorfiniu šifravimu grįstas modelis, naudojantis gradientinio nuolydžio algoritmą, diverguoja su MNIST rašytinių skaičių nuotraukomis.
- Paprastas PyTorch neuroninis tinklas vienintelis iš minėtų modelių konverguoja minimumo link, su MNIST rašytinių skaičių nuotraukomis. Šiuo atveju, DMDK rodiklis vidutiniškai nukrenta iki 0.049 ir modelis tampa mažiau saugus, nei Pallier algoritmu grįstas modelis.
- Didinant duomenų triukšmą, didėja DMDK reikšmė ir modelis tampa saugesnis, tačiau sumažėja modelio tikslumas.
- Pradinių duomenų kiekis neturi įtakos modelio duomenų saugumui.

11 Išvados

Darbe buvo išanalizuoti algoritmai, kurie yra skirti duomenų privatumo išsaugojimui mašininiame mokymesi. Darbe buvo realizuotas vienas iš homomorfinio šifravimo algoritmų - Paillier sistema. Atlikti saugumo tyrimai parodė, kad naudojant gradientinį nuolydį ir Paillier sistemą, pradiniai modelio duomenys yra saugesni, nei tik naudojant gradientinį nuolydį be jokio šifravimo. Nustatyta, kad modelio tikslumui artėjant 100%, greitai kyla ir DMDK reikšmė. Tai reiškia, kad modelio pradiniai duomenys tampa saugesni. Planuojant, kad modelio tikslumas bus arti 100%, modelio duomenų saugumui pagerinti reikia naudoti homomorfinį šifravimą.

Buvo atliktas duomenų saugumo tyrimas su PyTorch 3-jų sluoksnių neuroniniu modeliu. Analizuojant Sklearn bibliotekoje esančius cukriniu diabetu sergančių žmonių duomenis, paaiškėjo, kad PyTorch neuroninis modelis yra 20 kartų saugesnis nei Paillier kriptogravifos taikymas su gradientiniu nuolydžio algoritmu. Pastebėta tendencija, kad modelio tikslumui artėjant 100%, mažėja DMDK reikšmė ir modelis tampa mažiau saugus. Jis labiau prisiriša prie pradinių duomenų ir linksta link jų atskleidimo. Planuojant, kad modelio tikslumas nebus didelis ir siek tik 60% tikslumą, reiktų apsvarstyti homomorfinio šifravimo saugumą su tokiu uždaviniu.

Minėtas PyTorch tyrimas su cukriniu diabetu sergančiais duomenimis parodė, kad esant 15 parametrų, modelis yra saugesnis už homomorfinį šifravimą. Kitas atliktas tyrimas buvo su MNIST duomenų rinkiniu. Šiame rinkynyje yra ranka rašyti skaitmenys. Esant dideliam modelio paramentrų skaičiui, PyTorch prarado savo pradinių duomenų saugumą, jis tapo mažesnis už Paillier kriptografijos saugumą. Esant didesniam parametrų skaičiui, PyTorch 3-jų sluoksnių modelis yra labiau linkęs “prisirišti” prie pradinių duomenų. Prieš kuriant modelį ir pasirenkant algoritmą reikia apsvarstyti kokio tikslumo modelio galima tikėtis gauti ir kiek parametrų modelis priims, nes nuo to priklauso ar naudoti neuronininus tinklus ar homomorfinį šifravimą.

Kitame tyrime pastebėta koreliacija tarp modelio saugumo metrikos DMDK ir pridedamo triukšmo. Kuo pridedamas triukšmas yra didesnis, tuo DMDK metrika yra didesnė ir modelio pradiniai duomenys yra saugesnis, tačiau didinant pridedamą triukšmą, mažėja modelio tikslumas. Planuojant, kad bus naudojami neuroniniai tinklai be homomorfinio šifravimo ir tikimasi didesnio nei 80% modelio tikslumo, vertėtų pridėti triukšmą prie pradinių modelio duomenų.

Literatūros sąrašas

[BEG+19] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design, 2019.

[Bha19] Dr. Santanu Bhattacharya. The new dawn of ai: Federated learning, Jan 2019.

[BJJ+18] Ho Bae, Jaehee Jang, Dahuin Jung, Hyemi Jang, Heonseok Ha, and Sungroh Yoon. Security and privacy issues in deep learning, 2018.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003.

[Cho19] Seongjae Choi. Secret sharing, May 2019.

[CKKS16] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers, 2016.

[CLE+19] Nicholas Carlini, Chang Liu, Ulfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks, 2019.

[CTW+20] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models, 2020.

[CWo13] CWoo. Algebraic system, Mar 2013.

[Fra20] Jake Frankenfield. Smart contracts: What you need to know, Dec 2020.

[Har19] Justin D. Harris. Leveraging blockchain to make machine learning models more accessible, Jul 2019.

[Inp] Inpher. What is secure multiparty computation? - smpc/mpc explained.

[Lin20] Yehuda Lindell. Secure multiparty computation (mpc), 2020.

[LSTS19] Tian Li, Anit Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions, 08 2019.

[LSZ+20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

[MAE+19] H. Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. A general approach to adding differential privacy to iterative training procedures, 2019.

[Par16] Europos Parlamentas. Europos parlamento ir tarybos reglamentas, 04 2016.

[Pat10] Patrice. Patrices lectures: Compsci 373 semester 1, city campus, 05 2010.

[PG18] Nicolas Papernot and Ian Goodfellow. Privacy and machine learning: two unexpected allies?, Apr 2018.

[Raf18] Eric Rafaloff. Shamir's secret sharing scheme · eric rafaloff, Oct 2018.

[RE19] Carey Radebaugh and Ulfar Erlingsson. Introducing tensorflow privacy: Learning with differential privacy for training data, Mar 2019.

[Reg10] Oded Regev. The learning with errors problem. 01 2010.

[SCD+20] Gan Sun, Yang Cong, Jiahua Dong, Qiang Wang, and Ji Liu. Data poisoning attacks on federated machine learning, 2020.

[Sen13] Jaydip Sen. Homomorphic encryption: Theory & applications, 07 2013.

[Tha20] Patricia Thaine. Perfectly privacy-preserving ai, 01 2020.

[Wil15] Ko Ryan K.L. Will, Mark A. Paillier cryptosystem. *Paillier Cryptosystem - an overview*, 2015.

[WLH+21] Su Wang, Mengyuan Lee, Seyyedali Hosseinalipour, Roberto Morabito, Mung Chiang, and Christopher G. Brinton. Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation, 2021.

[YLCT19] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications, 2019.

[ZZJ+20] Yang Zhao, Jun Zhao, Linshan Jiang, Rui Tan, Dusit Niyato, Zengxiang Li, Lingjuan Lyu, and Yingbo Liu. Privacy-preserving blockchain-based federated learning for iot devices, 2020.