УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Павле Васиљевић

Федеративна изолациона шума за ефикасну детекцију аномалија у системима на ивици интернета

МАСТЕР РАД

Нови Сад, 2025.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Дипломски – мастер рад	
Аутор, **АУ**:		Pavle Vasiljević	
Ментор, **МН**:		проф. др Мирослав Поповић	
Наслов рада, **НР**:		Федеративна изолациона шума за ефикасну детекцију аномалија у системима на ивици интернета	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2025	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		6/43/16/3/20/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кључне речи, **ПО**:		Системи на ивици интернета, Федеративно учење, Изолационе Шуме, Детекција Аномалија, Дистрибуирани системи	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		У овом раду представљен је PFLiForest, алгоритам за детекцију аномалија заснован на изолационим шумама у оквиру окружења PTB-FLA. Алгоритам се ослања на формално верификован и структуриран приступ федеративном учењу, чиме се обезбеђује коректност и поузданост процеса обуке уз ефикасно коришћење рачунарских ресурса. Експериментална евалуација показала је висок ниво тачности предвиђања уз мало заузеће меморије и процесорског времена, што га чини погодним за примену у интелигентним системима на ивици интернета.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	проф. др Силвиа Гилезан	
	Члан:	проф. др Илија Башичевић	Потпис ментора
	Члан, ментор:	проф. др Мирослав Поповић	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		Pavle Vasiljević	
Mentor, **MN**:		Miroslav Popović, PhD	
Title, **TI**:		Federated Isolation Forest Algorithm for Efficient Anomaly detection on Edge Systems	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2025	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		6/43/16/3/20/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Edge systems, Federated Learning, Isolation Forests, Anomaly Detection, Distributed Systems	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This thesis presents PFLiForest, an algorithm for anomaly detection based on isolation forests, developed within the PTB-FLA framework. The algorithm relies on a formally verified and structured federated learning approach, ensuring the correctness and reliability of the training process while maintaining efficient use of computational resources. Experimental evaluation demonstrated a high level of predictive accuracy with low memory and execution time requirements, thus making the approach well-suited for deployment in intelligent edge systems.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Silvia Gilezan, PhD	
	Member:	Ilija Bašičević, PhD	Menthor's sign
	Member, Mentor:	Miroslav Popović, PhD	

	УНИВЕРЗИТЕТ У НОВОМ САДУ • **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА ЗАВРШНИ РАД	Датум:

Студијски програм:	Рачунарство и аутоматика		
Студент:	Павле Васиљевић	Број индекса:	Е2 65/2024
Степен и врста студија:	Мастер академске студије		
Област:	Електротехника и рачунарство		
Ментор:	проф. др Мирослав Поповић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:
- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

Наслов завршног рада:

Федеративна изолациона шума за ефикасну детекцију аномалија у системима на ивици интернета

Текст задатка

У оквиру овог мастер рада потребно је урадити следеће:
1. Истражити теоријске основе федеративног учења и алгоритама детекције аномалија.
2. Дати преглед ПТБ-ФЛА и МПТ-ФЛА окружења за развој алгоритама федеративног учења.
3. Приказати тренутно стање у области.
4. Развити алгоритам детекције аномалија у контексту федеративног учења.
5. Пројектовати систем заснован на развијеном алгоритму, укључујући архитектуру и понашање у различитим фазама рада.
6. Имплементирати систем детекције аномалија заснован на предложеном алгоритму.
7. Дати преглед имплементације и уочених ограничења.
8. Дефинисати експерименталну методологију и евалуационе метрике за анализу перформанси алгоритма.
9. Развити апликацију за мерење перформанси алгоритма.
10. Дефинисати ограничења циљне платформе и одредити оптималне параметре за извршавање алгоритма на њој.
11. Спровести експерименталну евалуацију и дискусију резултата евалуације.
12. Дати закључке о перформансама, применљивости приступа у реалним системима на ивици интернета.

Руководилац студијског програма:	Ментор рада:

Примерак за: ◯ - Студента; ◯ - Ментора

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

ИЗЈАВА О НЕПОСТОЈАЊУ СУКОБА ИНТЕРЕСА

Изјављујем да нисам у сукобу интереса у односу ментор – кандидат и да нисам члан породице (супружник или ванбрачни партнер, родитељ или усвојитељ, дете или усвојеник), повезано лице (крвни сродник ментора/кандидата у правој линији, односно у побочној линији закључно са другим степеном сродства, као ни физичко лице које се према другим основама и околностима може оправдано сматрати интересно повезаним са ментором или кандидатом), односно да нисам зависан/на од ментора/кандидата, да не постоје околности које би могле да утичу на моју непристрасност, нити да стичем било какве користи или погодности за себе или друго лице било позитивним или негативним исходом, као и да немам приватни интерес који утиче, може да утиче или изгледа као да утиче на однос ментор-кандидат.

У Новом Саду, дана _____

Ментор

_____

Кандидат

_____

Захвалност

Посебну захвалност бих изразио својој породици, пријатељима и колегама на безрезервној подршци током студија.

Такође, дубоку захвалност упућујем свом ментору, проф. др Мирославу Поповићу, на сарадњи, саветима и подршци, не само приликом израде овог рада, већ још значајније, током целих мастер студија. Његово менторство, у пуном значењу те речи, било је од пресудног значаја за моје научно и професионално сазревање, и помогло ми је да се формирам као један млади истраживач какав сматрам да сваки мастер студент треба да буде.

Садржај

Списак слика

Списак табела

Скраћенице

PTB-FLA - *Python Testbed for Federated Learning Alghoritms*

MPT-FLA - *MicroPython Testbed for Federated Learning Algorithms*

ФЛ - федеративно учење

API - *Application Programing Interface*

TCP - *Transmission Control Protocol*

RIFIFI - *revised isolation forest to identify fraud and the data inner structure*

LIF - *Layered Isolation Forest*

ФПП - функција повратног позива

1. Увод

Федеративно учење нуди решење за ограничења традиционалних централизованих приступа машинског учења, који захтевају масовно прикупљање података, нарушавају приватност корисника и ослањају се на велике рачунарске ресурсе. Значај овог приступа је посебно изражен у системима на ивици интернета, где је неопходно обезбедити ефикасно и континуирана обука модела на ивици интернета. У таквим окружењима, изолационе шуме се издвајају као лагана и робусна техника за детекцију аномалија, која не захтева обележавање података, минимално зависи од подешавања хиперпараметара и самим тим је погодна за примену на уређајима са ограниченим ресурсима.

Досадашња истраживања која су разматрала примену изолационих шума у контексту федеративног учења фокусирала су се углавном на нетривијалан задатак агрегације појединачних модела изолационих шума, али без структурираног приступа, кроз ad-hoc комуникацију између чворова. Такође, ови радови посматрају федеративне изолационе шуме са становишта перформанси модела, без осврта на рачунарске ресурсе неопходне за обуку оваквих модела.

Како би надоместио недостатке истраживања који су му претходили, овај рад, као проширење рада [1], настоји да додатно размотри и формализује већ постојеће приступе, ослањајући се на структуиран приступ дат у оквиру окружења за федеративно учење PTB-FLA, који се заснива формално верификованим генеричким алгоритмима федеративног учења. Поред овога, детаљно су евалуиране перформансе добијених модела и њихова зависност од хиперпараметара, ефикасност процеса обуке и применљивости на системе са ограниченим ресурсима.

Рад је организован у шест поглавља. Друго поглавље даје теоријску позадину неопходну за потпуно разумевање овог рада и преглед истраживања која су му предходила.

У трећем поглављу дат је предлог решења који у себи садржи детаљан опис пројектованог алгоритма, и детаље о архитектури и имплементацији система базираног на њему. Четврто поглавље се бави екперименталном евалуацијом система, и садржи преглед и објашњење коришћених мера, експерименталну методологију, и дискусију резултата. Поглавље пет износи закључак рада, док је у шестом поглављу дат преглед литературе и извора који су коришћени за израду рада.

2. Теоријске основе и сродни радови

У овој секцији представићемо теоријске основе неопходне за разумевање овог рада и споменути и дати кратак преглед сродних радова који се баве сличним истраживањем. Најпре ћемо увести појмове федеративног учења и изолационих шума. Затим ћемо дати преглед окружења коришћених у имплементацији алгоритма федеративног учења. Напокон даћемо и кратак преглед сродних радова који су претходили овом.

2.1 Федеративно учење

Федеративно учење (ФЛ) (*енгл. Federated Learning*) [2] је техника машинског учења у којој више уређаја заједнички обучавају модел, где у процесу обуке клијентски подаци остају локални, тј. приватни. Подаци који се размењују у процесу ФЛ су нове вредности параметара модела добијени у последњој итерацији обуке. Из овакве децентрализованости података у федеративном учењу следе и следећа својства: побољшана приватност, робустност система, отклањање пристрасности модела ка локалним подацима. Федеративно учење у односу на друге технике машинског има и следћа својства: хетерогена расподела података међу чворовима учесницима у процесу учења, која није нужно репрезентативна за цео скуп података; разлике у количини података који су локално доступни за обуку сваком чвору који учествује у ФЛ; ограничена количина комуникације; могућност масовне дистрибуираности, алгоритам ради и када је број учесника у процесу учења већи од просечне количине одбирака на сваком уређају.

Процес ФЛ је у својој природи синхрон, и одвија се у комуникационим рундама. Постоји фиксан скуп од K клијената од којих свако поседује своје локалне податке. На почетку рунде неки подскуп клијената је изабран за учешће у њој. Сваки учесник у учењу

(клијент ФЛ) обучава модел на основу свог локалног скупа података и глобалног стања и по завршетку шаље резултат чвору који има улогу сервера. Сервер након тога прави нови модел на основу нових локалних измена и прослеђује глобално стање клијентима, почињући нову рунду процеса ФЛ.

Препоставка је да ће се процес обуке одвијати само када уређаји имају повољне услове (нпр. неискоришћено процесорско време, напуњена батерија, добра мрежна веза), и да се обука неће понављати превише често. Ова одлика федеративног учења чини га изузетно погодним за имплементацију у контексту уређаја са ограниченим ресурсима који се налазе на ивици интернета. Поред тога задржавањем података локално на уређајима повећава се ниво приватности и сигурности података што је од кључног значаја у многим применама. Овај вид дистибуираног машинског учења такође доприноси и скалабилности система, елиминишући потребу за централизованим чувањем података, и бригу о приватности и анонимизацији истих.

Најчешћи приступ ФЛ у пракси је федеративно усредњавање (*енгл. Federated Averaging, FedAvg*), где сваки клијент локално обучава модел користећи сопствене податке, а затим шаље ажуриране параметре централном серверу који их агрегира.

2.1.1 Класификација федеративног учења

Једна од примарних подела ФЛ [3] односи се на архитектуру комуникације. Централизовано ФЛ продразумева да један централни чвор (сервер) организује читав процес учења, одговоран је за одабир чворова који ће учествовати у учењу, и за скупљање и спајање приспелих модела у нови модел који ће бити коришћен у наредним итерацијама учења. Мана централизованог федаративног учења јесте свакако што сервер може да постане уско грло система, али и потенцијална јединствена тачка отказа целог система. Насупрот томе, у децентрализованом ФЛ чворови међусобно сарађују како би дошли до глобалног модела, без потребе за предефинисаним чвором на ком се одија агрегација модела. Овај приступ елеминише јединствену тачку отказа система, али је понашање система зависно од топологије мреже, и начина на који је дефинисана политика агрегације. Са аспекта расподеле података, федеративно учење се дели на хоризонтално и вертикално. Хоризонтално федеративно учење подразумева да различити клијенти имају различите одбирке података (*енгл. data points*), али са истом структуром одлика (*енгл. feature*). Насупрот томе, вертикално федеративно учење настаје у ситуацијама када различити клијенти имају податке о истим одабирцима, али не и исте одлике у сваком чвору.

Федеративно учење се разликује и по типу учесника у систему. У такозваном међу-уређајном (*енгл. cross-device*) сценарију, учествује велики број хетерогених клијената са

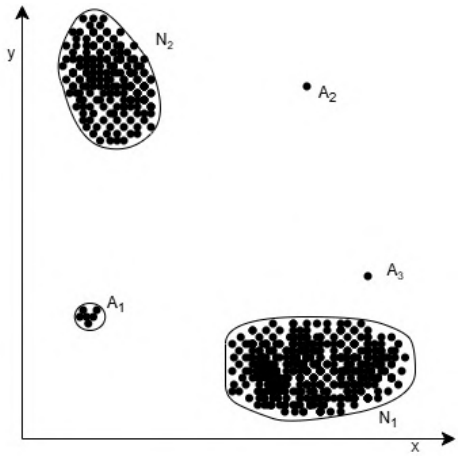
потенцијално нестабилном везом, на пример мобилни уређаји или IoT сензори. Ови уређаји поседују малу количину података, често неуравнотежену, а могућност њиховог учешћа у обуци зависи од услова као што су приступ мрежи, напуњеност батерије и неактивност уређаја. Насупрот томе, у међу-институционалном (*енгл. cross-silo*) федеративном учењу учествује мањи број поузданих институционалних клијената, као што су болнице, банке или универзитети. Ови клијенти располажу већим количинама података и могу учествовати у више рунди учења са већим степеном континуитета и прецизности.

Као посебна категорија издваја се и хијерархијско федеративно [4] учење, које представља архитектонски хибрид централизованог и децентрализованог приступа. Уместо једног централизованог агрегатора, примењује се вишеслојни систем агрегирања. На првом нивоу, клијенти шаљу параметре локалним агрегаторима (нпр. регионалним серверима), који их затим шаљу централном агрегатору на вишем нивоу.

2.2 Детекција аномалија

Детекција аномалија [5] представља проблем проналажења правилности у подацима који немају очекивана својства. Овакви подаци који одударају од остатка скупа података се називају аномалије или изузеци.

Аномалије су подаци који немају својства претходно јасно дефинисана као нормална. Слика 1 приказује аномалије на једноставном дводимензионалном скупу података. На њој су приказане две нормалне регије N_1 и N_2 у којима се налази већина података. Одбирци који су довољно удаљени од регија N_1 и N_2 у регији A_1 и тачкама A_2, A_3 представљају аномалије.



Слика 1: Аномалије у 2-димензионалном простору

Значај детекције аномалија је у чињеници да је на основу њих могуће добити значајне информације неопходне за започињање неког процеса унутар система у разним применама.

Добар пример је детекција неправилности у мрежном саобраћају која може указати на потенцијални напад на инфраструктуру, и која би могла покренути промену политике у администрацији мреже.

Неки од изазова у детекцији аномалија су: тешкоћа дефинисања нормалног понашања, еволуција нормалног понашања током времена, зависност од домена, недостатак означених података и присуство шума.

Груба подела детекције аномалија у домену машинског учења је следећа: надгледана (подаци означени као нормални и изузеци), полу-надгледана (део података означен) и ненадгледана (неозначени подаци) детекција аномалија.

2.3 Изолационе шуме

Изолационе шуме [6] представљају ненадгледану ансамбл методу детекције аномалија која се која кроз изолацију података кроз серију насумичних подела открива аномалије. Ова метода је ненадгледана јер јој нису потребни означени подаци да би детектовала изузетке али је истовремено и ансамбл метода јер истовремено користи више појединачних модела (изолационих стабала, *енгл.*. *Isolation trees*). Оно што чини овај метод другачијим од већине осталих метода детекције аномалија је чињеница да није неопходно моделовање нормалности већ се користи механизам изолације инстанци случајним поделама.

Алгоритам обуке модела се своди на прављење више изолационих стабала. Стабло се гради на следећи начин: случајно се одабере атрибут по коме се деле подаци, из опсега вредности одабраног атрибута скупа података за обуку се узме случајна тачка, такозвана тачка поделе, која представља главни податак у чвору стабла, након тога сви подаци са вредношћу мањом од тачке поделе иду у подскуп леве гране стабла, а сви подаци већи од вредности тачке поделе иду у посдскуп десне гране стабла. Овај процес се понавља у левој и десној грани рекурзивно док се испуни услов завршетка (енгл. termination), који код алгоритма прављења стабла могу бити: (1) достизање максималне висине стабла, (2) да број инстанци у скупу буде један, или (3) да све инстанце у скупу података имају исте вредности. Процес прављења стабла се понавља t пута, где је t број стабала у моделу, изолационе шуме.

Алгоритам откривања аномалија обученим моделом изолационе шуме своди се на пролазак кроз сва стабла, након чега се бележи дужина пута неопходна да би се потенцијална аномалија сместила у стабло. Након тога се рачуна просечна дубина стабла и вредност резултата аномалије (*енгл. Anomaly score*). На крају се ради рангирање према оценама аномалија, где су примери са највишим оценама заправо највероватније аномалије.

У пракси се обично дефинише и праг аномалије, где се примери са вредносима резултата аномалије већим од тог прага аутоматски класификују као аномалије.

Изолационе шуме су предложене као ефикасан алгоритам за детекцију аномалија у великим скуповима података. Временска сложеност обуке је $O(t \cdot \psi \cdot \log \psi)$ где је t број стабала, ψ кардиналност скупа података за обуку, а $\log \psi$ дубина стабла. Временска сложеност откривања аномалије, тј. закључивања (*енгл. inference*) је $O(t \cdot \log \psi)$. t број је $O(t \cdot \log \psi)$. t број стабала, а $\log \psi$ просечна дубина стабла.

Предности овог приступа су свакако његова брзина и ефикасност, његова ненадгледана природа и неослањање на дистрибуцију и доменску природу података. Случајеви када овај метод не даје најбоље резултате су када постоје аномалије маскиране међу нормалним подацима и када имамо веома сложене скупове података са сложеним граничним скуповима.

2.4 PTB-FLA и MPT-FLA

PTB-FLA [7][9] (*енгл. Python Testbed for Federated Learning Alghoritms*) представља окружење за равој алгоритама федеративног учења, односно извршно окружење за ФЛ алгоритме на једном рачунару. Ако се у обзир узме и постојање јасно дефинисаног процеса за развој алгоритама на коректан начин, тј. ФЛ развојна парадигма базирана на PTB-FLA, интеграција унутар Visual Studio Code алата, и остали алати који доприносе лакшем коришћењу, PTB-FLA се може сматрати и пунокрвним радним оквиром за развој ФЛ апликација. Писан је у чистом Python-у, користећи само стандардне Python пакете као што је multiprocessing. Оваква имплементација доноси компактност и поједноствљену инсталацији у односу на сличне алате.

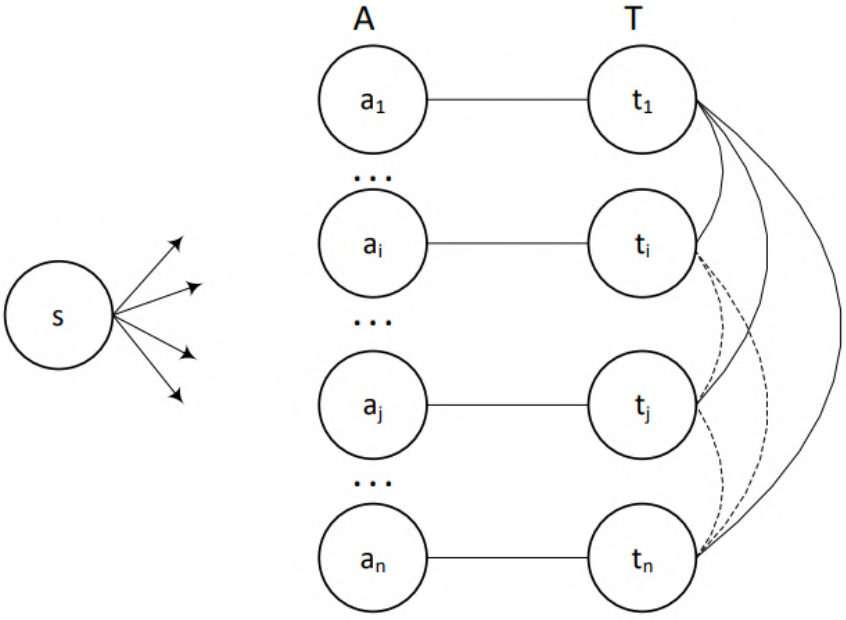
MPT-FLA [8][9] (*енгл. MicroPython Testbed for Federated Learning Algorithms*) изведен је из PTB-FLA, са фокусом на рад у локалној мрежи и на уређајима на ивици интернета (*енгл. edge devices*), за разлику од његовог претходника који је радио само на једном рачунару. MPT-FLA је заснован на апстракцијама које нуди Python-ov asyncio пакет (asyncio корутине, токови и догађаји) и покреће се у како у стандардном Python окружењу тако и у MicroPython окружењу (подскуп Python-а 3 која се извршава директно на микроконтролерима без потребе за оперативним системом).

PTB-FLA од корисника при писању алгоритама захтева две кључне ствари: корисник мора написати једну апликацију која се касније покреће од стране PTB-FLA лансера као скуп независних процеса који на основу своје идентификације (скраћено ID) испољавају одређено понашење (овај захтев проистиче из чињенице да је PTB-FLA базиран на један

програм више података шаблону, *енгл. single program multiple data*); други захтев је да корисник обезбеди функције повратног позива (енг. callback functions) које описују понашање сервера и клијента, и које се у извршавању позивају од стране генеричког алгоритма федеративног учења унутар PTB-FLA. Подржани су како централизовани тако и децентрализовани алгоритми федеративног учења. Уз њих подржане су и примитиве за временски мултиплексирану размену података (*енгл. time division multiplex peer data exchange*). Генерички алгоритми које обезбеђује PTB-FLA су генеричко централизовано ФЛ, генеричко децентрализовано ФЛ и временски мултиплексирана размена података. Особине наведене у овом одељку важе и за MPT-FLA.

2.4.1 Архитектура PTB-FLA

Архитектура PTB-FLA (представљена на слици 2), састоји се од процеса апликације лансера, *S*, дистрибуиране апликације $A = \{a_1 , a_2 , ..., a_n\}$ (*која представља скуп инстаци апликације* a_i) и дистрибуираног оружења $T = \{t_1 , t_2 , ..., t_n\}$ (*скупа инстанци окружења*), где је $i = 1, 2, ..., n$; а n представља број инстанци од A и T.



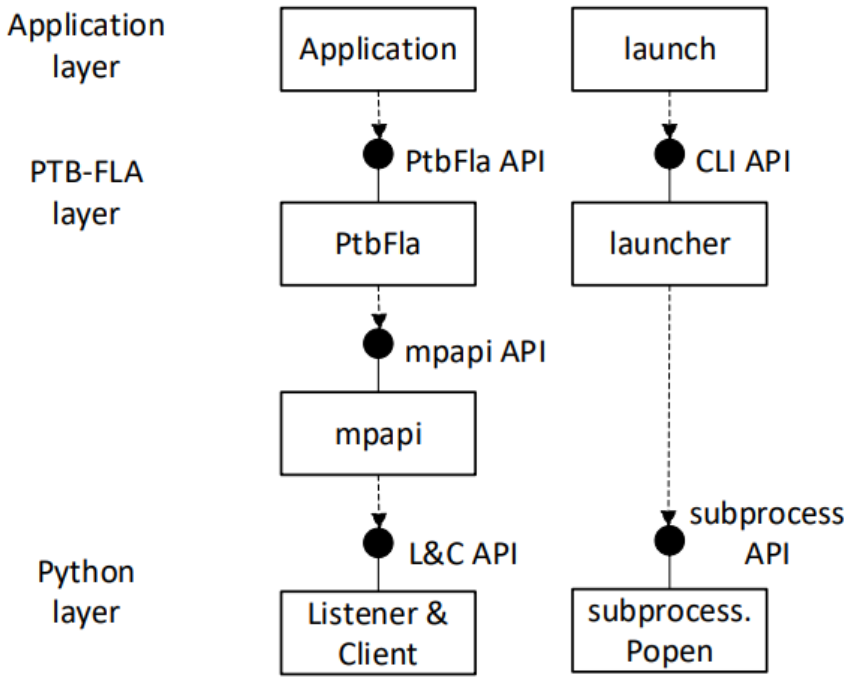
Слика 2: Блок дијаграм архитектуре PTB-FLA

(уз дозволу аутора, преузето из референце [4])

Током извршавања дистрибуирана апликација *A* користи дистрибуирано окружење *T* како би извршавала дистрибуирани алгоритам, који је дефинисан у функцији повратног позива унутар апликације и прослеђен генеричком алгоритму. Извршавање се одвија на следећи начин: Свака инстанца ai припрема податке на основу аргумената командне линије, затим потива генеричке API функције (fl_centralized или fl_decentralized) над инстанцом

окружења t_i Инстанца окружења на основу ID-ја i извршава своју улогу у генеричком алгоритму притом размењујући поруке са другим инстанцама окружења.

PTB-FLA има слојевиту архитектуру и састоји се од (слика 3): дистрибуиране апликације на врху (пише је корисник), PTB-FLA слоја (апстрахује генеричке алгоритме и комуникацију међу чворовима тј. инстанцама) и Python слоја (који обезбеђује класе за рад са multiprocessing процесима као и друге Python примитиве). Преглед архитектуре PTB-FLA је преузет из референце [4].

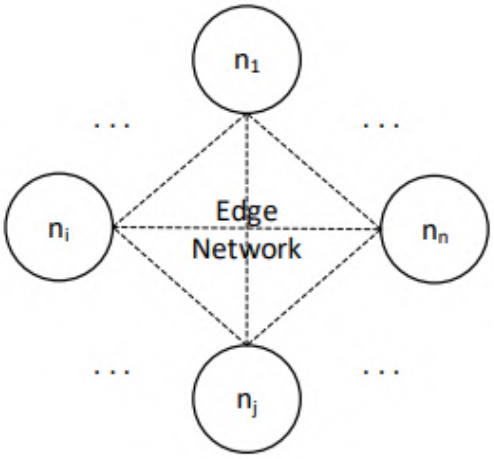


Слика 3:UML дијаграм архитектуре PTB-FLA

(уз дозволу аутора, преузето из референце [4])

2.4.2 Архитектура MPT-FLA

MPT-FLA је систем на ивици интернета (*енгл. edge system*) који се може представити као граф са n чворова (са идентификаторима од 1 до *n* (у имплементацији обележени од 0 до *n*-1)) повезаних луковима, где су чворови процеси који се извршавају на неком уређају (паметни уређаји, рачунари итд.), а лукови су TCP везе међу њима (слика 4). На слици 4 лукови су приказани испрекиданим линијама јер се везе између чворова динамички стварају на захтев за размену порука између учесника у комуникацији, а након завршетка комуникације се раскидају.



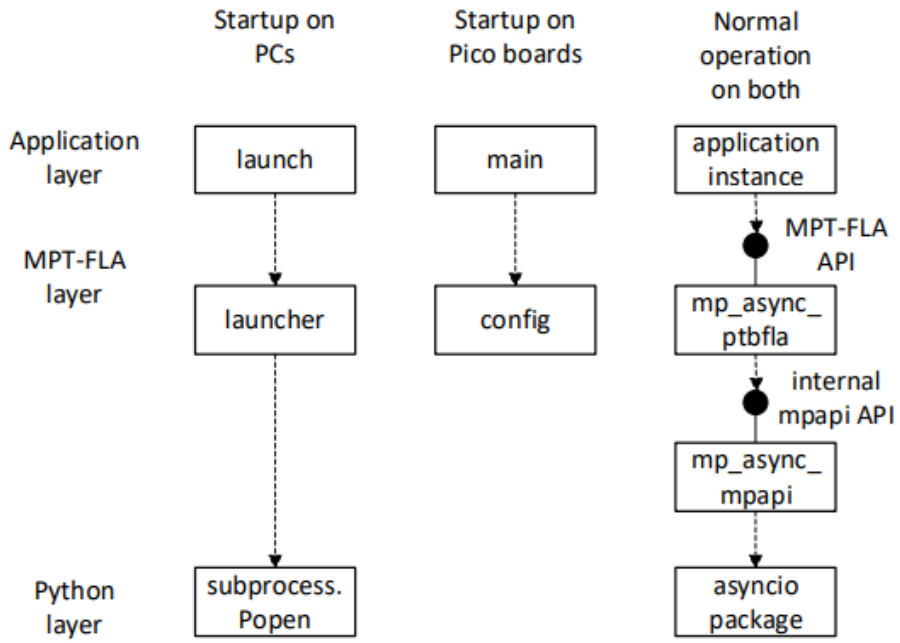
Слика 4: Архитектура MPT-FLAa MPT-FLA

(уз дозволу аутора, преузето из референце [5])

При покретању система, и већином времена, везе између чворова не постоје, и у току извршавања се по потреби динамички успостављају и раскидају након завршетка њихове употребе. Из тога проилзилази да ако два чвора n_i и n_j никада не комуницирају директно између њих никада неће бити успостављена веза. Које везе ће бити успостављене зависи директно од режима у ком систем ради тако да: код генеричког централизованог ФЛ алгоритма граф приказан на слици 4 узима облик звезде, код децентрализованих ФЛ алгоритама облик клике, а код временски мултиплексиране размене података узима облик произвољног графа где су само парови чворова повезани. Сваки чвор ni извршава једну инстанцу апликације a_i који на почетку ствара инстанцу окружења ti . Скуп инстанци $\{a_1, ..., a_n\}$ чини дистрибуирану апликацију *A*, док скуп инстанци MPT-FLA окружења $\{t_1, ..., t_n\}$ чини дистрибуирано окружење *T*. У току извршавања A користи окружење T како би извршавао жељени алготитам користећи генричке алгоритме дате од стране MPT-FLA. Оваква архитектура је идејно еквивалентна архитектури PTB-FLA, упркос другачијој имплементацији и физичкој дистрибуираности чворова.

Архитектура MPT-FLA такође је организована у три слоја: Апликациони слој, MPT-FLA слој и Python слој. Између архитектуре овог окружења и његовог преткодника може се повући јасна паралела са разликом у средњем слоју који замењује стари PTB-FLA слој заснован на Python multiprocessing апстракцијама, новим MPT-FLA слојем заснованим на asyncio пакету. MPT-FLA слој се састоји од модула mp_async_ptbfla и mp_async_mpapi (где префикс mp_async означава респективно MicroPython и asyncio). Осим хоризонталних слојева на слици 5 можемо увидети и вертикалне колоне које предствљају начин покретања MPT-FL апликације како на рачунарима који користе потпуну верзију Python-а 3 (лево) тако

и на уређајима који користе MicroPython (средина), уз начин рада заједнички за обе врсте уређаја (десно).



Слика 5: UML дијаграми архитектуре MPT-FLA

(уз дозволу аутора, преузето из референце [5])

При покретању, инстанца апликације користи MPT-FLA API који јој пружа класа PtbFla, из модула mp_async_ptbfla, ради стварања и уништавања инстанце окружења, покретања чвора, и покретања једног од три генеричка алгоритма. Класа PtbFla користи mpapi API (API за слање порука) који обезбеђује следеће asyncio корутине: server_fun, који се извршава од стране asyncio задатка, који ствара покретање PtbFla корутине; sendMsg шаље једну поруку; rcvMsg прима једну поруку; broadcastMsg шаље поруку свим чворовима; rcvMsgs прима одређени број порука (обично *n*-1). MPT-FLA прати слојевиту архитектуру, апстрахујући међусобну комуникацију између чворова, омогућујући корснику да се усредсреди на развој саме апликације федеративног учења. Од посебног значаја је и могућност покретања MPT-FLA на микроконтролерима који подржавају MicroPython, где се апликације понашају на еквивалентан начин. Преглед архитектуре MPT-FLA је преузет из референце [4].

2.5 Сродни радови

У овом одељку дат је преглед радова који се баве сродним темама. Превасходно изолационим шумама и њиховом применом у контексту федеративног учења.

У раду Јемпоа, Смитса, Лесота и Пиверта [10] предложена је ревидирана изолациона шума за проналажење аномалија (*енгл. revised isolation forest to identify fraud and the data inner structure, RIFIFI*) аутори се баве једним од кључних ограничења класичног алгоритма изолационих шума, немогућности очувања структурних особености података приликом детекције аномалија. У RIFIFI приступу, уместо потпуно случајних подела као у класичној изолационој шуми, уводи се критеријум за избор вредности поделе који избегава поделе у густим регионима тачака, чиме се чува структура података. RIFIFI приступ притом боље чува кластере података и доприноси бољој објашњивости детекције аномалија, што омогућава већу проверљивост излаза модела. Оно што чини овај приступ потенцијално неповољним за примену у уређајима на ивици интернета јесу додатна меморија (константа) и додатно време (линеарно) неопходно за извршавање овог алгоритма, у односу на основни алгоритам изолационих шума. Такође у контексту федеративног учења јавила би се и додатна сложеност у синхронизацији и агрегацији података између чворова.

Приступ приказан у раду [11], назван слојевита изолациона шума (*енгл. Layered Isolation Forest, LIF*), предлаже унапређење класичног алгоритма изолационих шума, тако што дели простор података на више подпростора и прави локалне шуме за сваку регију, унапређујући детекцију локалних аномалија. Даљим комбиновањем локалних и глобалних вредности резултата аномалије постиже се прецизнија детекција. Овим приступом решава се проблем привидно нормалних региона и побољшава се откривање локалних аномалија у сложенијим дистрибуцијама података. Мана овог приступа је свакако повећана сложеност подешавања и зависност од избора параметара, али такође можда и битније за поставку проблема, увећање количине меморијског простора неопходног за рад алгоритма, који је последица чувања више локалних и једне глобалне изолационе шуме. Оно што чини овај приступ занимљивим јесте баш примена локалних шума над подскупом података што јесте на неки начин слично партиционисању података по клијентима у ФЛ.

Горе споменути радови баве се унапређењем својстава изолационих шума централизованим приступом, мада већ други приступ даје назнаке да би федеративни приступ могао бити примењив, ако не и повољан. Оно где федеративни приступ засигурно предњачи у односу на већ споменуте приступе јесте у погледу приватности података и заједничког учења између уређаја.

У докторској дисертацији Јунјан Лија [12] представљено је конкретно решење за откривање аномалија у контексту интернета ствари (*енгл. Internet of Things, IoT*) употребом

изолационих шума. Појединачна изолациона стабла се локално праве на сваком од уређаја, и уместо података или параметара модела шаљу метаподатке о чворовима стабала (идентификатори, димензија по којој се узима тачка поделе, дубина, границе интервала итд.) у облику стека. Пре слања на податке се додаје Лапласов шум, ради очувања приватности. Сервер затим агрегира чворове на основу идентификатора, узимајући тачку поделе из глобалних интервала за сваки чвор, тиме формирајући глобалну изолациону шуму која омогућава откривање аномалија на основу података са свих уређаја без дељења локалних података. Алгоритам је експериментално евалуиран над више скупова података стандардних за детекцију аномалија генерално али и специфично у IoT контексту као што су: „EdgeIIoT“, „IoT23“, „NB15DoS“, „NB15Worms“, „KDD99Smurf“, „Glass dataset“. Аспект у ком ова имплементација свакако предњачи јесте анонимност података и оптимизације коришћења мрежних ресурса, али упркос томе ради са претпоставком о усклађености чворова и има нешто слабије перформансе од оригиналног алгоритма изолационе шуме. Агрегација се у овој имплементацији ослања на претпоставку да су идентификатори чворова усаглашени и да су стабла грађена по сличним критеријумима, што може бити ограничавајући фактор у реалним системима и захтева додатан ниво синхронизације.

У раду Хаолонг Сјанга и др. [13] предложен је алгоритам федеративних изолационих шума, „FLiForest“, заснован на слојевитом грађењу изолационих стабала. Сваки слој изолационог стабла се гради тако што клијенти локално генеришу тачке поделе, шаљу их серверу који их агрегира, и тако добијене глобалне вредности враћа клијентима као тачку поделе која ће бити коришћена за даљу изградњу стабла. Овај алгоритам је фокусиран на континуум облак-магла-ивица (*енгл*. cloud-fog-edge continuum), без конкретизације за реалне системе на ивици интернета. Дата је и опсежна експериментална евалуација алгоритма над стандардним вишедимензионалним скуповима података као што су „Celeba“, „ALOI“, „MNIST“, „FashionMNIST“ , „Skin“ и сл.

Проблем алгоритама изнетих у претходнопоменутим радовима јесте сложеност имплементације и ad-hoc приступ комуникацији између учесника у процесу ФЛ. Овакав приступ може довести до разних грешака, које су тешко поновљиве, неправилне синхронизације између учесника итд. Овакав приступ је малтене неприменљив за имплементацију у уређајима на ивици интернета због додатно отежаног откривања грешака, које долази са одсуством развојног окружења и апстракција које нуди оперативни систем. Још када се у обзир узме дистрибуирана природа ФЛ у опште и нестабилне везе у бежичним мрежама, које јесу доминантан комуникациони медиј у IoT окружењу, количина рада за ad-hoc имплементације оваквих система почиње нагло да расте. Такође оно што оба

алгоритма занемарују јесте и сама прикладност приступа за имплементацију у системима са ограниченим ресурсима, перформансе модела који су обучени на њима, али и потрошња ресурса неопходних за обуку и рад система.

Из горе наведених мана већ постојећих решења по идеји „FLiForest“ алгоритма направљен је нови алгоритам који је назван „PFLiForest“, тј. „PTB-FLA FLiForest“, базиран на генеричким алгоритмима ФЛ које нуди PTB-FLA. Генерички алгоритми су претходно и формално верификовани употребом CSP модела [14], тиме гарантујући коректност комуникације између чворова. Још једна предност примене PTB-FLA јесте постојање јасно дефинисаног начина за имплементацију нових алгоритама машинског учења у виду PTB-FLA четворофазне парадигме за развој ФЛ алгоритама [15]. Уз алгоритам дата је и анализа повољности алгоритма за примену у уређајима на ивици интернета. На тај начин су адресирани уочени недостаци претходних истраживања.

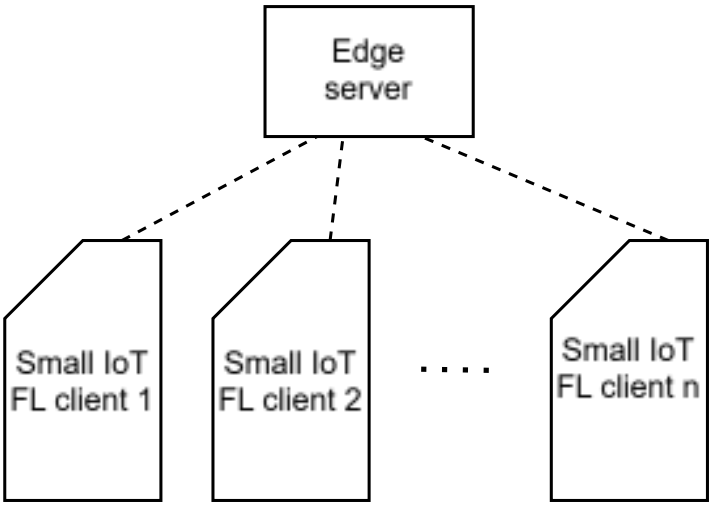
3. Предлог решења

Ово поглавље износи предлог решења федеративне изолационе шуме применом PTB-FLA окружења. У наредним одељцима представићемо архитектуру система, понашање система и дистибуирани алгоритам обуке модела који представља главни део решења, али и начин на који се предвиђају аномалије над улазним одабирцима.

3.1 Архитектура система

Архитектура система представљеног у овом раду, у својој основи представља архитектуру типичну за системе који имплементирају алгоритеме централизованог федеративног учења. Састоји се од *n* клијената, покренутим на уређајима са ограниченим ресурсима који на себи имају сензоре температуре (нпр. Raspberry Pi Pico W), и једног сервера на ивици интернета. Наша имплементација заснована је на PTB-FLA, специфично његовом генеричком централизованом алгоритму федеративног учења (fl_centralized), притом пратећи шаблон један програм више података, што значи да сваки клијент или сервер може да заузме обе улоге, на основу података који су му доступни. У оквиру система сваки чвор извршава инстанцу PTB-FLA, која је задужена за комуникацију међу њима. У погледу топологије мреже, клијенти ФЛ и сервер на ивици формирају топологију звезде. Клијентски и серверски процеси федеративног учења изражени су кроз функције повратног позива (у даљем тексту ће бити представљене кроз алгоритме 3 и 4) које се прослеђују као улазни параметри централизованог генеричког ФЛ алгоритма датом у PTB-FLA. У својој целости систем у потпуности прати слојевиту архитектуру специфичну за коришћено окружење, изнето у [4]. Архитектура система заснованог на PFLiForest алгоритму, намењеног детекцији аномалија уз помоћ федеративног учења, дата је на слици 6. Битно је

напоменути да је овај систем специјализован за предвиђање аномалија у једнодимензионалном простору, на пример предвиђање аномалија температуре у одређеним окружењима.



Слика 6: Архитектура система

3.2 Понашање система

По узору на FLiForest алгоритам, и његов принцип прављења стабала слој по слој, у нашем приступу, традиционални приступ обуке читавих модела у оквиру клијентских функција повратног позива (у даљем тексту ради концизности ФПП) измењен је на следећи начин: (1) клијентска ФПП израчунава вредности тачака подела (*енгл. split points*) у оквиру стабла (2) сереверска ФПП их упросечава. Овакав приступ доноси повећану количину интеракције између клијената и сервера али даје коректну конвергенцију процеса обуке изолационе шуме у контексту федеративног учења уз предности које су стандардне за ФЛ.

У наставку ове секције биће дати псеудокодови алгоритама који чине PFLiForrest и њихова обајшњења.

Табела 1: Глобално дефинисане променљиве које се користе у алгоритмима 1-4

```
01: //тренутна фаза
02: SERVER_PHASE = 0
03: CLIENT_PHASE = 0
04: //вредности фазе унутар процеса праљења стабла
05: INITIAL = 0
06: CLIENT_RESTING = 1
07: END_TREE = 2
```

У табели 1 дат је преглед глобалних променљивих који се користе у свим алгоритмима у оквиру PFLiForrest-а. На линијама 2 и 3 дефинисане су променљиве које служе за праћење

тренутног стања фазе у процесу стварања једног стабла. На линијама 5-7 дефинисане су промењљиве које представљају могуће вредности стања фазе.

Алгоритам 1. Прављење изолационе шуме

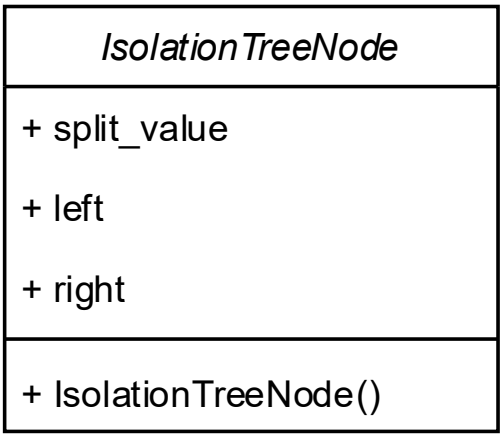
```
01: build_iForest (ptbFla, data, num_trees, max_depth):
02:   forest = []
03:   for _ in range(num_trees)
04:     tree = build_iTree (ptbFla, data, max_depth)
05:     forest. append(tree)
06:   return forest
```

У оквиру алгоритма 1, унутар петље (линија 3) прави се изолациона шума позивом алгоритма 2 (линија 4) како би се направило појединачно стабло које затим бива додато у колекцију стабала тј. шуму (линија 5). Петља се извршава све док се не достигне жељени број стабала (задат кроз вредност аргумента *num_trees*). По завршетку алгоритам 1 повратна вредност алгоритма 1 јесте цео модел изолационе шуме.

Алгоритам 2. Прављење изолационог стабла

```
01: build_iTree (ptbFla, data, max_depth):
02:   // Иницијализација променљивих неопходних за прављење стабла
03:   global CLIENT_PHASE, SERVER_PHASE
04:   root = IsolationTreeNode()
05:   queue = deque([(root, data, 0)])
06:   while True:
07:     if queue:
08:       //узимање вредности из реда
09:       node, pData, depth = queue.popleft ()
10:     else:
11:       node, pData, depth = None, None, None
12:     //комуникација за време федеративног стварања тачке поделе
13:     ret = ptbFla.fl_centralized (serverProcessing,clientProcessing, localData, pData, 1)
14:     //синхронизација фазе између чворова
15:     phase = ptbFla.fl_centralized (serverPhase, clientPhase, ret["phase"], ret["phase"], 1)
16:     //излаз под условом да су сви клијенти направили изолационо стабло
17:     if phase == END_TREE:
18:       //повратна вредност је у потпуности направљено стабло
19:       return root
20:     //у случају да је у питању серверски чвор остатак ове функције се занемарује
21:     if ptbFla.nodeId = ptbFla.flSrvId:
22:       continue
23:     // ако је клијент у фази мировања, не треба додавати нове чворове
24:     if CLIENT_PHASE == CLIENT_RESTING or not node:
25:       continue
26:     // постављање клијента у фазу мировања када је завршио прављење свог стабла
27:     if depth >= max_depth or len(set(pData)) <= 1:
28:       if not queue:
29:         CLIENT_PHASE = CLIENT_RESTING
30:       continue
31:     //прављење левог и десног детета
32:     node.left = IsolationTreeNode()
33:     node.right = IsolationTreeNode()
34:     //додела вредности тачке поделе
35:     node.split_value = ret["data"]
36:     //подела података на леву и десну партицију
37:     left_partition, right_partition = split_data(node.split_value)
38:     //додавање чворова у ред и повећавање тренутне дубине
39:     queue.append((node.left, left_partitions, depth + 1))
40:     queue.append((node.right, right_partitions, depth + 1))
```

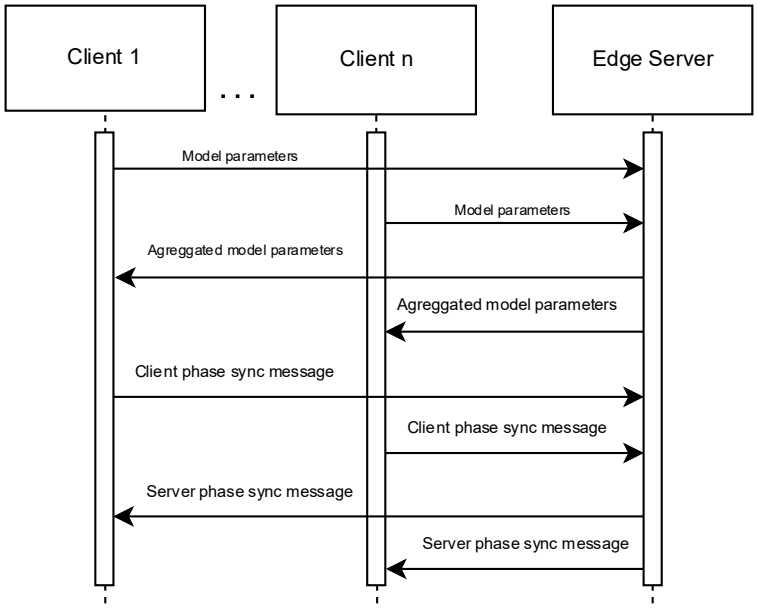
Алгоритам 2 представља процес прављења појединачног изолационог стабла на федеративан начин. Овај алгоритам је имплементиран итеративно користећи ред, како би се пратио раст стабла, уместо ослањања на позивни стек, како би се задржала компатибилност са PTB-FLA окружењем. На самом почетку алгоритма (линија 4) иницијализован је корен изолационог стабла позивом конструктора класе IsolationTreeNode (UML дијаграм ове класе дат је на слици 7), као и ред који се користи за праћење раста стабла (линија 5).



Слика 7: UML дијаграм класе IsolationTreeNode

Након иницијализација неопходних променљивих почиње итеративно извршавање алгоритма (линија 6) где се у свакој итерацији из реда узима чвор (линија 9) над којим се одређује вредност тачке поделе све док се не испразни у потпуности, што се дешава у фази мировања. У линији 13 позива се генерерички алгоритам *fl_centralized* [7] над PTB-FLA објектом. Овом алгоритму као функције повратног позива прослеђују се алгоритми 3 и 4 о којима ће бити више речи у даљем тексту. Одмах по завршетку процеса добијања тачке поделе синхронизује се фаза између чворова (линија 15). Овај механизам је неопходан како неки чворови не би превремено напустили процес прављења стабла и започели прављење следећег. Овакав асинхронизам би довео до неправилног стварања стабала, и био би контрадикторан PTB-FLA функцији *fl_centralized* где нова итерација обуке модела не почиње све док глобални модел није прослеђен свим клијентским чворовима. Као повратна вредност синхронизације фазе очекује се фаза у којој се тренутно налази сервер. На слици 8 дат је преглед једне комуникационе рунде у процесу прављења стабла. За почетак сви клијенти шаљу своје параметре модела (вредности тачке поделе изолационог стабла) серверском чвору. Затим клијентски чворови добијају упросечене вредности параметара модела од сервера, и користе га за прављење стабла. Комуникација клијент-сервер,

сервер-клијент догађа се након тога још један пут, преносећи информације о фази у којој се налазе клијенти и сервер.



Слика 8: MSC дијаграм једне комуникационе рунде

Ако је добијена фаза завршна (END_TREE) тада сви клијенти завршавају алгоритам прављења стабла и кроз повратну вредност враћају стабло (линије 17 и 19). Део алгоритма након линије 21 изоставља се од стране серверског чвора, како он не прави своју изолациону шуму, већ само агрегира параметре модела клијентских чворова. Клијентски чворови у фази мировања (CLIENT_RESTING) такође директно прелазе у слећу итерацију прављења стабла (линије 24 и 25).

Клијенти који су и даље у процесу прављења свог стабла затим проверавају да ли је достигнута максимална дубина стабла или да ли је скуп података постао хомоген, такође се проверава и да ли је ред празан. Ако су већ поменути услови испуњени, клијентски чвор улази у фазу мировања. Фаза мировања се може сматрати локалним завршетком прављења стабла појединачног клијента, док се завршна фаза може сматрати глобалним завршетком алгоритма 1.

У линијама 32 и 33 стварају се објекти левог и десног потомка чвора који је обрађен у овој итерацији, након овога се додељује и вредност тачке поделе том чвору и на основу глобалне тачке поделе те итерације раздељују се подскупови података који ће бити основа добијања тачке поделе левог и десног потомка тренутног чвора (линија 37). На самом крају сваке итерације у оквиру алгоритма 1 на клијентским чворовима у иницијалној фази (INITIAL , коју би смо могли назвати и фазом стварања) на крај реда се додају леви и десни чвор стабла, њихови скупови података, добијени у подели (на линији 37) и вредност дубине на којој ће се наћи тај чвор. Овај процес се понавља све до испуњења глобалног услова

изласка тј. до уласка у фазу завршетка када се као повратна вредност на сваком од чворова враћа у потпуности формирано изолационо стабло.

Алгоритам 3. Клијентска функција повратног позива

```
01: clientProcessing (localData, privateData, msg):
02:   global CLIENT_PHASE
02:   // обавештавање сервера да је клијент завршио прављење стабла
03:   if CLIENT_PHASE == CLIENT_RESTING:
04:      return {"phase": CLIENT_PHASE, "data": 0}
05:   // Израчунавање локалних тачака поделе
06:   splits = client_process_layer(privateData)
07:   // слање података ка серверу
08:   return {"phase": CLIENT_PHASE, "data": splits}
```

Алгоритам 3 представља клијентску функцију повратног позива која има две улоге. Њена примарна улога је да генерише вредност тачке поделе специфичног клијента користећи његове приватне податке (линија 6). Функција *client_process_layer* као повратну вредност враћа вредност тачке поделе, као и величину леве и десне партиције након поделе по локалној вредности. Ови подаци се заједно са тренутном фазом шаљу серверу (линија 8). Секундарна улога ове функције јесте да обавести серверски чвор да је одређени клијент завршио прављење свог стабла слањем своје тренутне фазе серверу (линија 4).

Алгоритам 4. Серверска функција повратног позива

```
01: serverProcessing (localData, msgs):
02:   global SERVER_PHASE
03:   clientSplits = []
03:   // Filtering messages of clients who are resting
04:   for item in msgs:
05:     if item["phase"] != CLIENT_RESTING:
06:       clientSplits.append(item["data"])
07:   //if there are no client splits for aggregation, then all
        clients are done constructing their trees, can move
        to the next tree
08:   if length(clientSplits) == 0:
09:      SERVER_PHASE = END_TREE
10:      return {"phase": SERVER_PHASE, "data": None}
11:    // Aggregate client data
12:    return {"phase": SERVER_PHASE, "data":
              server_aggregate_layer (clientSplits)}
```

Алгоритам 4 представља серверску функцију повратног позива, у којој се клијентске вредности тачке поделе агрегирају упросечавањем њихових вредности. Додатно серверска функција повратног позива има за задатак да прати да ли су сви клијенти завршили прављење својих стабала. Закључивање да ли је прављење стабала готово у свим клијентима се своди на филтрирање порука тако да остану само поруке стабала која су у процесу прављења стабала (линије 3 до 6). Ако након филтрирања не остане ни једна порука шаље се порука за улазак у завршну фазу (линије 8 до 10). Ако ипак постоји још чворова који стварају стабло, онда се параметри модела добијени од клијента агрегирају, тежинским усредњавањем, некон чега се резултат шаље назад клијентима (линија 12). Синхронизација фазе која се дешава у алгоритму 4 (линије 10 и 12) и у алгоритму 2 (линија 15) кључна је

како би сигнал END_TREE, који је услов изласка из алгоритма 2 стигао до свих учесника у процесу прављења стабла. Функција повратног позива serverPhase шаље тренутну фазу сервера, која може бити иницијална или завршна, клијентима користећи PTB-FLA генерички алгоритам fl_centralized. Функција повратног позива clientPhase након тога прима информацију о фази и враћа је кроз повратну вредност, где се она користи као услов завршетка алгоритма 2.

Процес обуке изолационе шуме има потенцијално високу комуникациону сложеност O $(n\ 2^m)$, где је n број стабала у шуми, а m је максимална висина стабла. У најбољем случају, комуникациона сложеност је Ω $(n\ m)$. Ова висока комуникациона сложеност би потенцијално могла бити побољшана чувањем неких резултата локално и њиховим слањем од једном када се нагомилају. Мана овог приступа је што би процес агрегације и услов завршетка били сложенији, и захтевао би додатне метаподатке како би дошло до коректне конвергенције модела.

Резултат извршавања алгоритама 1-4 је у обучен модел изолационе шуме у облику листе изолационих стабала, спреман за процес предвиђања аномалија на сваком од клијената, добијен у процесу заједничког федеративног учења, без дељења приватних података.

3.3 Предвиђање аномалија у систему

Предвиђање аномалија у оквиру изолационих шума заснива се на додавању испитиваних узорака у сва појединачна стабла у оквиру изолационе шуме. За свако додавање бележи се дубина на којој је новододата вредност. На основу просечне дубине израчунава се вредност аномалије (једначина 1). Вредноста аномалије узима вредности између 0 и 1. Вредности ближе 1 указују на већу вредност да је узорак аномалија док вредност ближе 0 указују да је узорак вероватно нормалан.

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \quad (1)$$

$$c(n) = 2(\ln(n-1) + 0.577) - \frac{2(n-1)}{n} \quad (2)$$

У једначини 1, $E(h(x))$ представља просечну дубину у оквиру изолационе шуме, а $c(n)$ (који се израчунава на основу једначине 2) представља коефицијент нормализације и директно је зависан од величине скупа података на ком је модел обучен, који је изражен кроз променљиву n. Број узорака и нормализација неопходни су највише за шуме прављене на основу великог броја података и које имају велику максималну дубину. Променљива x

представља узорак за који се одређује да ли је аномалија. Израчуната вредност аномалије се након овога пореди са одређеним прагом аномалије који одређује да ли је одбирак аномалија. Прагови аномалија нису универзални и често се одређују експериментално, статистичким методама или на основу валидације и метрика перформанси над означеним тестним скупом података. Процес предвиђања аномалија одвија се на сваком од клијената појединачно све до следеће рунде обуке модела PFLiForest.

3.4 Имплементациони детаљи

Систем је у потпуности имплементиран у програмском језику Пајтон, као и модул PTB-FLA који представља основу алгоритма PFLiForest. Модул numpy је коришћен за математчке операције. Модул random коришћен је за добијање вредности тачака подела у процесу прављења стабла. За учитавање и предобраду података коришћен је модул pandas. За добијање вредности у окиру експерименталне евалуације коришћене су функционалности из модула scikit-learn. За чување података експеримента коришћена је SQLite база података, како би смо на једноставан начин могли сачувати резултате на диску, без бриге о конкурентном приступу између више чворова, јер SQLite сам по себи серијализује паралелне уписе у исту табелу. Такође употреба SQL базе подтака нам омогућава ефикасан начин прегледа резултата и извршавање упита над њима, поједностављујући процес анализе података.

3.5 Ограничења система

Алгоритам PFLiForest и његова имплементација представљени у овом раду као главно ограничење имају ограниченост на једнодимензионалне податке, што је и било предпоствљено ограничење у процесу пројектовања алгоритма, како је главни циљ била имплементација ефикасне методе детекције аномалија над температурним подацима, директно усмерена на окружења са ограниченим ресурсима. Уз то није имплементиран механизам за динамичку промену учесника у овиру процеса федеративног учења што је реална могућност у системима на ивици интернета. Још једно ограничење је што сама експериментална евалуација извршавана на једном рачунару у оквиру симулације што само по себи не одражава перформансе у реалном мрежном саобраћају већ представља најбољи сценарио у погледу времена извршења, због минималног кашњења порука.

4. Експериментална евалуација

У овој секцији представићемо експерименталну евалуацију PFLiForest алгоритма, и продискутовати њене резултате.

Улазни скуп података садржи у себи вредности температуре изражене у степенима целзијуса. Скуп података је сакупљен са реалних сензора из система кућне аутоматизације периоду од 6 месеци. У оквиру обуке и тестирања је у датој рунди обуке коришћен само подскуп података, у одрђеној временској околини с обзиром да амбијенталне температуре у августу и децембру знатно варирају.

Главне метрике на основу којих је евалуиран систем су: површина испод ROC (*енгл. receiver operating characteristic curve*) криве и површина испод криве прецизности и одзива (*енгл. precision/recall curve*). Ради концизности у даљем тексту за ове две мере ћемо увести скраћенице AUC-ROC и AUC-PR, и о њима и осталим мерама перформанси система биће више речи у секцији евалуационе метрике.

Главни задатак овог истраживања је да одреди оптималну изолациону шуму, која даје добре резултате, али и даље испуњава ограничења у виду ресурса доступних на конкретној платформи на ивици интернета. Циљна платформа је Raspberry PI Pico W микроконтролер који поседује двојезграни ARM Cortex-M0 процесор који поседује радну меморију од 256 KB и 16 MB меморије за податке.

Прво следи упознавање са евалуационим метрикама коришћеним у анализи перформанси система, након тога преглед поставке експеримента и методологије извођења евалуације.

Затим следи дискусија резултата: прво кроз анализу кључних метрика у односу на искоришћење меморије, након тога кроз зависност времена извршења и искоришћења меморије, утицаја појединачних параметара на перформансе, као и понашања метрика у односу на све параметре вариране у процесу обуке. На крају овог одељка смо упоредили перформансе PFLiForest алгоритма у односу на основни централизовани алгоритам изолационе шуме (iForest)[6].

Све мере приказане у наредним одељцима које се односе на искоришћење системских ресурса односе се на систем у процесу обуке, како је он описан алгоритмом приказаним у овом раду, те је од највећег интереса.

4.1 Евалуационе метрике

У циљу евалуације перформаси алгоритма федеративне изолационе шуме, неопходно је ослонити се на скуп метрика које заједно дају добар увид у ефикасност, тачност, пузданост система. Полазна мера перформанси из које се могу извести друге мере јесте матрица конфузије. Из ње можемо извести метрике као што су осетљивост, стопа лажних узбуна, прецизност и F1 мера, чија је улога да квантификују различите аспекте система. Ипак ове метрике на примеру изолационих шума зависе од избора прага детекције аномалије. Због тога се примарна пажња усмерава на површину испод ROC криве (AUC-ROC) и криве прецизности и осетљивости (AUC-PR), које показују укупну дискриминативну моћ система у читавом опсегу прагова. За праћење ефикасности искоришћења ресурса система, одлучили смо се за просечно време обуке и просечно искоришћење меморије. Управо комбиновањем ових метрика добија се целовита слика: колико је модел ефикасан у откривању аномалија и у којој мери је применљив у практичним системима ФЛ на ивици интернета

4.1.1 Матрица конфузије

Матрица конфузије (*енгл*. *Confusion matrix*) представља основну метрику за евалуацију модела класификације (у шта спада и детекција аномалија). Она је табеларни приказ који пореди предвиђене и стварне класе, омогућавајући детаљан увид у грешке које модел прави. Матрица садржи четири елемента:

- True Positive (TP) – број случајева у којима је модел исправно препознао позитивну класу
- False Positive (FP) – број негативних случајева које је модел погрешно препознао као позитивне
- False Negative (FN) – број позитивних случајева који је модел погрешно препознао као негативне
- True Negative (TN) – број случајева у којима је негативна класа исправно препозната

Интерпретација ових вредности је од суштинског значаја: TP и TN представљају исправна предвиђања, док FP и FN указују на врсте грешакка. У комбинацији, ове четири величине дају увид не само у то колико је модел тачан у целини већ и какву врсту грешака прави што је кључно за практичну примену у реалним системима. На слици 9 дат је приказ матрице конфузије у својој основној форми.

		Actual	
		Positive	Negative
Predicted	Positive	**True Positive**	**False Positive**
	Negative	**False Negative**	**True Negative**

Слика 9: Матрица конфузије

Осетљивост (*енгл. True Positive Rate, TPR, Recall, Sensitivity*) формално се дефинише као:

$$TPR = \frac{TP}{TP+FN} \quad (3)$$

Ова мера представља однос између броја исправно предвиђених позитивних примера и укупног броја тачно предвиђених примера. Осетљивост је показатељ способности модела да препозна све позитивне случајеве у скупу података. Висока вредност ове мере укаује на то да модел ретко пропушта позитивне одабирке, што је кључно у апликацијама где није допустиво игнорисати позитивне догађаје. Међутим, ова метрика не узима у обзир лажно позитивне догађаје, па се обично не корсти самостално.

4.1.2 Стопа лажно позитивних узорака

Стопа лажно позитвних узорака (*енгл. False Positive Rate, FPR*) дефинише се као:

$$FPR = \frac{FP}{FP+TN} \quad (4)$$

Представља однос између броја негативних примера који су погрешно класификовани као позитивни и укупног броја стварно негативних примера. FPR представља меру склоности ситема да ствара „лажне узбуне“. Висока вредност ове мере указује на недовољну дискриминативну моћ.

4.1.3 Прецизност

Прецизност (*енгл. Precision, Positive Predictive Value, PPV)* дефинише се као:

$$PPV = \frac{TP}{TP+FP} \quad (5)$$

Она представља однос између броја исправно предвиђених позитивних примера и укупног броја п позитивно предвиђених примера. Висока вредност значи да модел прави мало грешака у позитивном предвиђању, што је битно у системима где су трошкови лажних узбуна велики. Модел може имати високу прецизност али низак TPR у колико предвиђа мали број случајева као позитивне.

4.1.4 F1 мера

F1 мера представља хармонијску средину између прецизности и осетљивости, дефинисана је као:

$$F1_{score} = 2\frac{TPR \times PPV}{TPR + PPV} \quad (6)$$

F1 мера предствља одличан избор метрике у случајевима када је неопходно истовремено контролисати обе врсте грешака (лажни позитивни и лажни негативни резултати). Висока вредност ове метрике указује на добар баланс између способности модела да идентификује позитивне случајеве и да минимизује број лажних узбуна.

4.1.5 ROC крива и AUC-ROC

ROC (*енгл. Receiver Operating Characteristic*) крива је параметарска крива која показује односе између TPR и FPR за све могуће вредности прага класификатора $\theta \in [0,1]$. Формано се дефинише као:

$$ROC = \{(FPR(\theta), TPR(\theta)) \mid \theta \in [0,1]\} \quad (7)$$

Ова крива представља визуалицију компромиса између осетљивости и стопе лажно позитивних предвиђања.

AUC-ROC (*енгл. Area Under Curve Receiver Operating Characteristic*) предствља квантификацију целе ROC криве, и дефинише се кроз израз:

$$AUC_{ROC} = \int_0^1 TPR\left(FPR^{-1}(x)\right) dx \quad (8)$$

Вредност површине испод ROC криве показује укупну дискриминаторну моћ модела у разликовању нормалних и нерегуларних случајева. Вредности блиске 1 означавају висококвалитетан модел у овом погледу, док вредност 0.5 имплицира да модел нема дискриминаторну моћ и да је еквивалентан случајном погађању.

4.1.6 PR крива и AUC-PR

Крива прецизности и осетљивости (*енгл. Precision-Recall, PR*) је дефинисана као:

$$PR = \{(Recall(\theta)), Precision(\theta) \mid \theta \in [0,1]\} \quad (9)$$

Где θ представља праг класификације. Ова крива се често користи у задацима са неуравнотеженим класама, где је позитивна класа значајно мање заступљена од негативне. У таквим случајевима ROC крива не даје све информације већ је неопходно је додатно користити PR криву. Добар класификатор одржава вредности прецизности и осетљивости при већем делу криве.

AUC-PR (*енгл. Area Under Curve Precision Recall*) предствља квантификацију целе PR криве, и дефинише се кроз израз:

$$AUC_{PR} = \int_0^1 Precision(Recall^{-1}(x))\, dx \quad (10)$$

Ова метрика предствља просечну перформансу модела у одржавању прецизности при различитим нивоима одзива. Високе вредности ове метрике указују да модел устовремено задржава високе вредности прецизности и осетљивости. За разлику од AUC-ROC, ова метрика је стабилнија и информативнија у сценаријима са неуравнотеженим подацима.

4.1.7 Мере заузећа ресурса

Како би смо квантификовали погодност алгоритма за примену у системима на ивици интернета, неопходно је увести мере које описују његово понање са аспекта заузећа ресурса. У овом раду изабране су две кључне метрике: просечна потрошња меморије и просечно време извршавања обуке са пратећим стандардном девијацијама у процентима. Ове мере обухватају како просечно оптерећење система тако и стабилност његовог понашања. Потрошња меморије значајна је због честе мале количине радне меморије доступне у уређајима на ивици интернета, док време извршења обуке указује на применљивост алгоритма у дообучавању модела на ивици. Укључивање стандардних девијација у детаљнијој анализи оногућава процену предвидљиости и поузданости алгоритма што је кључно за процену употребљивости у реалним условима рада.

4.2 Поставка експеримента

Евалуација алгоритма за детекцију аномалија PFLiForest заснована је на серији експеримената у којима су варирани следећи параметри: величина изолационе шуме, максимална дубина стабла, количина података коришћена за обуку. Величина изолационе

шуме представља број стабала који се налазе у њој и у експериментима узима следеће вредности: 10, 25, 50, 75, 100 стабала. Максимална дубина стабла у поставци може бити 4, 6, 8, 10 ,12. Количина података за обуку, на сваком од клијената, варирана је од 50 до 200 тачака у корсцима од 50 тј. 50, 100, 150, 200. Укупна количина података у контексту федеративног учења представља умножак броја клијената и количине података за обуку на једном клијенту.

Циљ екперимената је пронаћи тачку (назовимо је E) која има максималне вредности AUC-ROC и AUC-PR , уз искоришћешћење меморије од приближно 160 kB:

$$E = M(\max(AUC_{ROC}), \max(AUC_{PR})) < 160 \quad (11)$$

Овај критеријум има за циљ да обезбеди најбоље перформансе модела унутар оквира одабране циљне платформе (у нашем случају Raspberry PI Pico W).

Горе наведене вредности параметара одабране су како би омогућиле анализу утицаја сложености модела и количине доступних података на перформансе алгоритма у различитим условима. На тај начин испитујемо компромис између перформанси система и прецизности модела, који је наручито битан у системима на ивици интернета, где доступност меморијских и рачунских ресурса представља значајан фактор за разматрање. На овај начин настојимо да експериментима прикажемо понашање PFLiForest система у реалним оквирима.

Комбинацијом наведених вредности независних параметара формиран је пун факторски пројекат експеримената [16], односно за сваку могућу комбинацију броја стабала, максимане дубине и количине података. На овај начин добијен је свеобухватни скуп конфигурација који омогућава анализу утицаја сваког појединачног параметра као и њихових интерекација. Свака конфигурација из овог скупа извршавана је 40 пута, чиме је онезбеђена валидност презултата и смањен утицај случајних промена услова током извршавања експеримената.

Ради добијања синтетичког тестног скупа података са лабелама, узет је валидациони скуп података, из њега су насумично изабрани одабирци и на њих је додат Гаусов шум. Унутар новог скупа вредности 10% насумичних одабирирака замењено је вредностима изван оригиналног опсега података, притом додајући лабеле на вредности из тестног скупа. Овако генерисан синтетички скуп података броји 10000 одбирака. Тестирање и гренерација тестног скупа података су обављени независно на сваком клијенту појединачно.

Како би смо у току тестирања одредили праг аномалије за истрнирани модел, за вредност прага аномалије узели смо вредност у којој је вредност F1 мере максимална за валидациони скуп података.

Сви резултати појединачних извршавања чувани су у бази података, након чега су агрегирани ради добијања коначних резултата. За сваку експерименталну конфигурацију израчунате су просечне вредности евалуационих метрика, које су након тога сачуване у CSV формату. Коначни резултати имали су следеће колоне:

- max_depth - максимална дубина стабла
- number_of_trees - број стабала у изолационој шуми
- training_data_amount - количина података коришћена за обуку по клијенту
- noise_level - ниво шума додат на скуп за обуку података како би се добио тестни скуп података
- anomaly_ratio - однос броја аномалија и нормалних тачака у тестном скупу података
- avg_auc_roc - просечна вредност AUC-ROC метрике
- avg_auc_pr - просечна вредност AUC-PR метрике
- avg_optThr - просечна вредност оптималног прага детекције аномалија
- avg_training_memory_usage - просечна потрошња меморија током обуке
- avg_training_time - просечна потрошња меморије у током обуке

Уз серију експеримената која испитује перформансе PFLiForest алгоритма, на исти начин евалуиран је и базни случај (iForrest алгоритам, тј. обична изолациона шума), једина разлика у конфигурацији је што је базном случају прослеђен кумулативан скуп података који је био доступан свим клијентима у процесу ФЛ, али распоређен међу њима.

За специфичну тачку која је изабрана као оптимална под задатим условима уређена је додатна серија експеримената која је као резултат има табеле конфузије модела за PFLiForest.

За контекст федеративног учења у оквиру експеримената битно је напоменути да су експерименти рађени са 2 клијента, са не-IID (*енгл. non-IID, non Independent and Identically Distributed*) подацима, што је конзистентно са типом података који се уобичајено појављују у IoT примени. Стратегија агрегације објашњена је у претходним одељцима.

У наредним одељцима представићемо резултате горе описаних експеримената и продискутовати их.

4.3 Резултати и дискусија

У наредним одељцима продискутоваћемо резултате добијене у претходно објашњеним експериментима са више аспеката. Циљ ове дискусије је да се детаљно сагледају перформансе предложеног алгоритма, како у погледу тачности и поузданости, тако и у погледу ефикасности употребе ресурса. Биће дато и поређење перформанси

предложеног алгоритма и основног алгоритма изолационе шуме. На овај начин настојимо да дамо свеобухватан преглед карактеристика система заснованог на предложеном алгоритму, што представља основу за процену његове применљивости у реалним сценаријима.

У већини резултата као тачке од интереса узете су тачке A-D које покривају широк опсег експерименталних конфигурација, које су задате као тројке (максимална дубина стабла, број стабала, количина података коришћена за обуку):

- A (4, 10, 50)
- B (6, 25, 100)
- C (8, 50, 150)
- D (10, 75, 200)

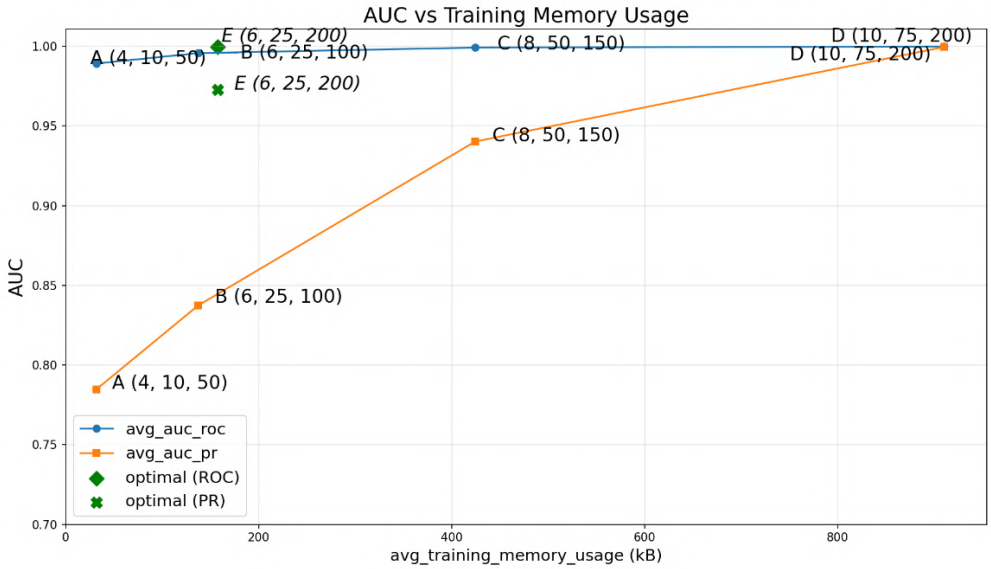
Уз њих дата је и тачка E (6, 25, 200) која је изабрана као оптимална конфигурација, која има највише вредности AUC-ROC и AUC-PR са заузећем меморије које је мање од 160kB, тј. задовољава једначину 11.

4.3.1 AUC-ROC и AUC-PR у односу на искоришћење меморије

Овај одељак представља широк поглед на перформансе система, кроз слику 10 приказан је однос између AUC метрика и потрошње меморије (које су везане за услове оптималности конфигурација, једначина 11). Овај приказ даје увид у опште трендове и разлике између конфигурација.

Уочава се да је метрика AUC-ROC у свим случајевима веома висока (практично изнад 0.99), што указује да модел врло поуздано разликује нормалне узорке и аномалије, независно од хиперпараметара. Ово вероватно проистиче из небалансираности тестног скупа података, у коме се налази само 10% аномалија, стога ради детаљније анализе перформанси морамо се окренути AUC-PR метрици.

За AUC-PR најлошије резултате постиже тачка A, као најскромнија конфигурација, уз постепено побољшаше кроз тачке B и C, док најбоље резултате очекивано постиже најзахтевнија конфигурација D. На графику су означене и вредности AUC-ROC и AUC-PR за тачку E, која је емпиријски изабрана из резултата за скуп свих експерименталних конфигурација, под условом из једначине 11. Она је по хиперпараметрима најближа тачки B али показује значајно боље перформансе, са минималним додатним заузећем меморије. Једина разлика јесте већа количина података доступних за обуку модела изолационе шуме сваком клијенту. Ово наводи на велику зависност перформанси модела од количине података за обуку, што ће и у неком од наредних одељака и бити детаљније разјашњено.

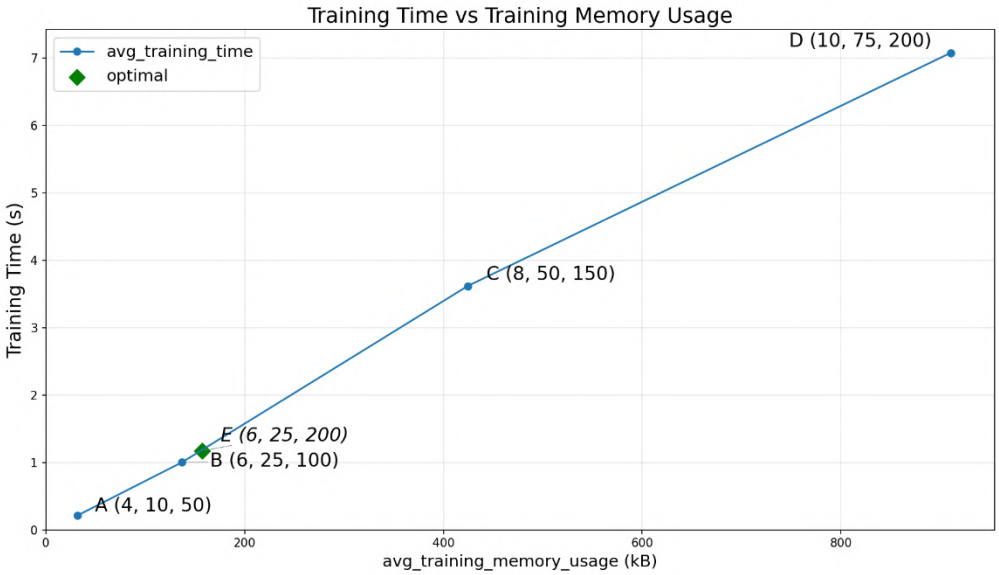


Слика 10: AUC-ROC и AUC-PR као функција искоришћења меморије

(Уз дозволу аутора адаптирано из [1])

4.3.2 Време извршавања и коришћена меморија

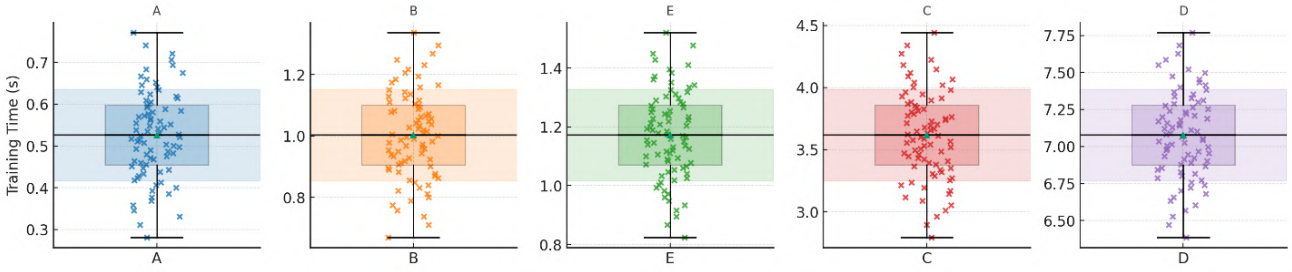
На слици 11 приказано је време извршавања алгоритма у односу на заузеће меморије за пет тачака од интереса (A-E). Резултати приказују скоро па линеаран пораст времена обуке у односу на раст искоришћења меморије. Ови резултати се поклапају са претпоставком да је главни фактор који доприноси повећању неопходне количине меморије модела у ствари величина модела.



Слика 11: Однос времена извршења и заузећа меморије у току обуке модела

(Уз дозволу аутора адаптирано из [1])

Упркос томе што на слици 11 можемо јасно видети корелацију између средњег времена обуке модела и средњег меморијског заузећа, ове мере нам саме по себи не дају целокупну слику о доследности понашања система од понављања од понављања, што је посебно битно испитати због понашања сакупљача некоришћене меморије (*енгл*. Garbage Collector) у овиру програмског језика Python. Из тог разлога урађена је и детаљна анализа вредности ових метрика за одабране тачаке.

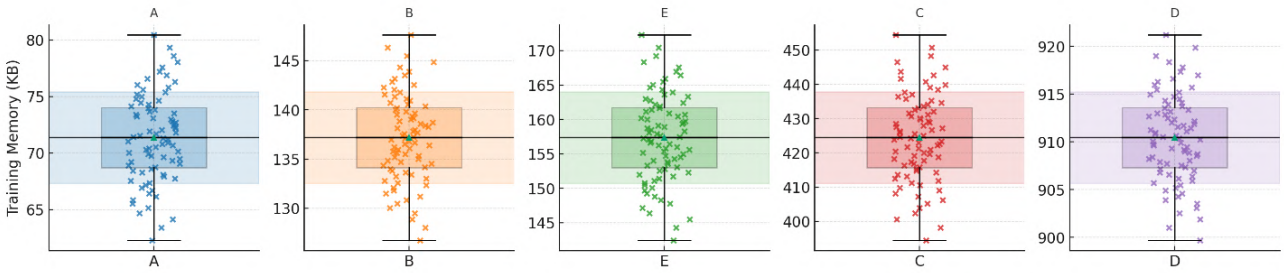


Слика 12: Време обуке модела за тачке од интереса

На слици 12 детаљно су представљена времена обуке за конфигурације од интереса (A, B, E, C, D). Свака конфигурација представљена је представљена кутијастим дијаграмом (*енгл. Box Plot*), где црна хоризонтална линија представља средњу вредност резултата, појединачни резултати су приказани као обојене тачке, обојени правоугаоник приказује распон у ком се налазе средњих 50% резултата, осенчени појас представља средњу вредност ± једну вредност апсолутне стандардне девијације, Док „антене“ представљају распон свих мерења. Већина вредности концентрисана је око средине што указује на малу варијабилност између извршавања. Опсег апсолутних стандардних девијација је релативно узак за свих пет тачака, што додатно потврђује да алгоритам има предвидиво и конзистентно време обуке. У табели 2 приказане су реалне вредности трајања обуке модела и релативне стандардне девијације за тачке од интереса.

Табела 2: Време обуке модела

Конфигурација	A	B	E	C	D
Време обуке (*s*)	0.526	1.003	1.172	3.617	7.077
Релативна стандардна девијација (%)	20.7	14.8	13.2	10.1	4.3



Слика 13: Искоришћење меморије у току обуке модела за тачке од интереса

Слика 13 представља кутијасти дијаграм укупног заузећа меморије за време обуке модела. Информације су представљене на исти начин као и на слици 13. Варијабилност унутар сваке конфигурације је мала што значи да су мерења конзистентна и да потрошња меморије зависи углавном од изабраних параметара, а не од случајних фактора у процесу обуке. Ови резултати оправдавају избор метрика за евалуацију системских перформанси, због стабилности добијених резултата између више покретања. У табели 3 дате су вредности заузећа меморије и релативне стандардне девијације за изабране конфигурације, добијене експерименталном евалуацијом.

Табела 3: Заузеће меморије у току обуке

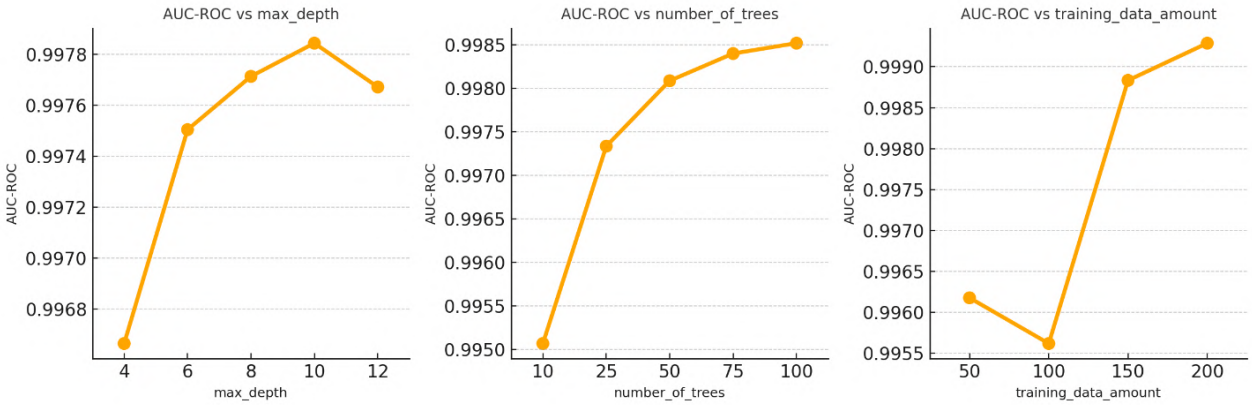
Конфигурација	A	B	E	C	D
Заузеће меморије (kB)	72	138	158	425	911
Релативна стандардна девијација (%)	5.6	4.3	4.1	3.1	0.52

4.3.3 Утицај појединачних параметара на перформансе

У овом одељку дати су графици који показују утицај појединачних параметара на евалуационе метрике. Како би се изоловао утицај појединачних хиперпараметара на перформансе модела, за сваку вредност једног параметра (нпр. максимална дубина), метрика (нпр. AUC-ROC) је упросечена преко свих осталих комбинација преосталих параметара (број стабала, количина података за обуку), чиме се добија просечан ефекат једног параметра. На овај начин се елеминише утицај варијација других параметара и добија

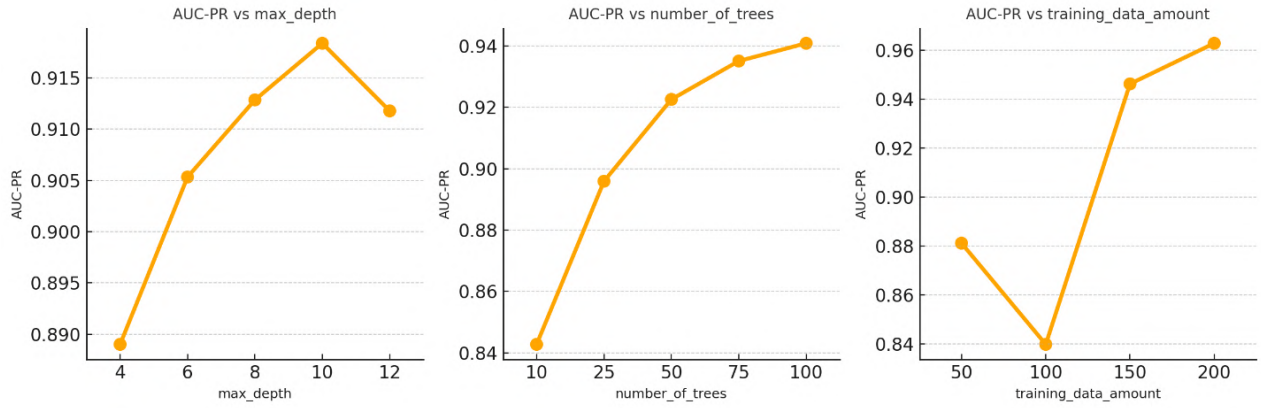
се јаснија слика о појединачном значају сваког параметра. Овај приступ се у литератури препоручује као начин за интерпретацију главних ефеката појединачних фактора. [16]

Слика 14, тј. утицај појединачних параметара на AUC-ROC, показује да повећање максималне дубине стабла утиче на побољшање перформанси до дубине 10 када долази до засићења и благог пада (мада се и даље ради о веома високој вредности до те мере да је пад практично занемарљив). Слично томе, пораст броја стабала позитивно утиче на вредности AUC-ROC, али добитци долазе до засићења након 75 стабала. Количина података за обуку има најјачи утицај, и перфрормансе имају највећи скок повећањем скупа података на 150 узорака. Овакав раст показује да умерено дубока стабла, довољно велики ансамбли и већи скупови података доприносе бољим перформансама. Наравно ове вредности треба узети са задршком, јер су све приказане вредности високе, али је глобални тренд свакако уочљив.



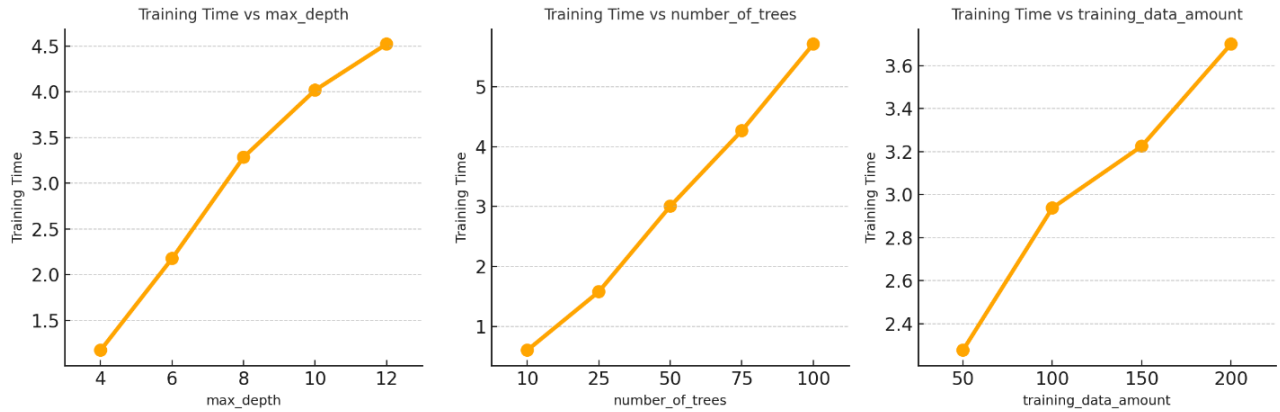
Слика 14: Утицај појединачних параметара на AUC-ROC

На графицима утицаја појединачних параметара на AUC-PR, слика 15, потврђују се већ изнета запажања са слике 14, само на већем распону вредности, дајући већу тежину уоченим трендовима. Међутим, ефекат количине података је посебно изражен: при 100 узорака AUC-PR пада, али се при 150 и 200 узорака перформансе нагло поправљају. Ово укзује да су перформансе веома осетљиве на количину података (самим тим вероватно и разноврсност опсега података) и да захтева критичну масу примера, како би ефикасно препознавао аномалије.



Слика 15: Утицај појединачних параметара на AUC-PR

Време обуке у односу на појединачне параметре приказано је на слици 16, и очекивано расте са сва три параметра али најзначајније са порастом дубине стабла и величине ансамбла. Дубља стабла и повећања броја стабала повећавају сложеност модела изолационе шуме и самим тим директно и број операција неопходан за прављење истог. У овиру изложеног алгоритма ово повећање не одражава се само у погледу броја инструкција, већ и у повећању количине комуникације неопходне за обуку модела.



Слика 16: Утицај појединачних параметара на искоришћење меморије

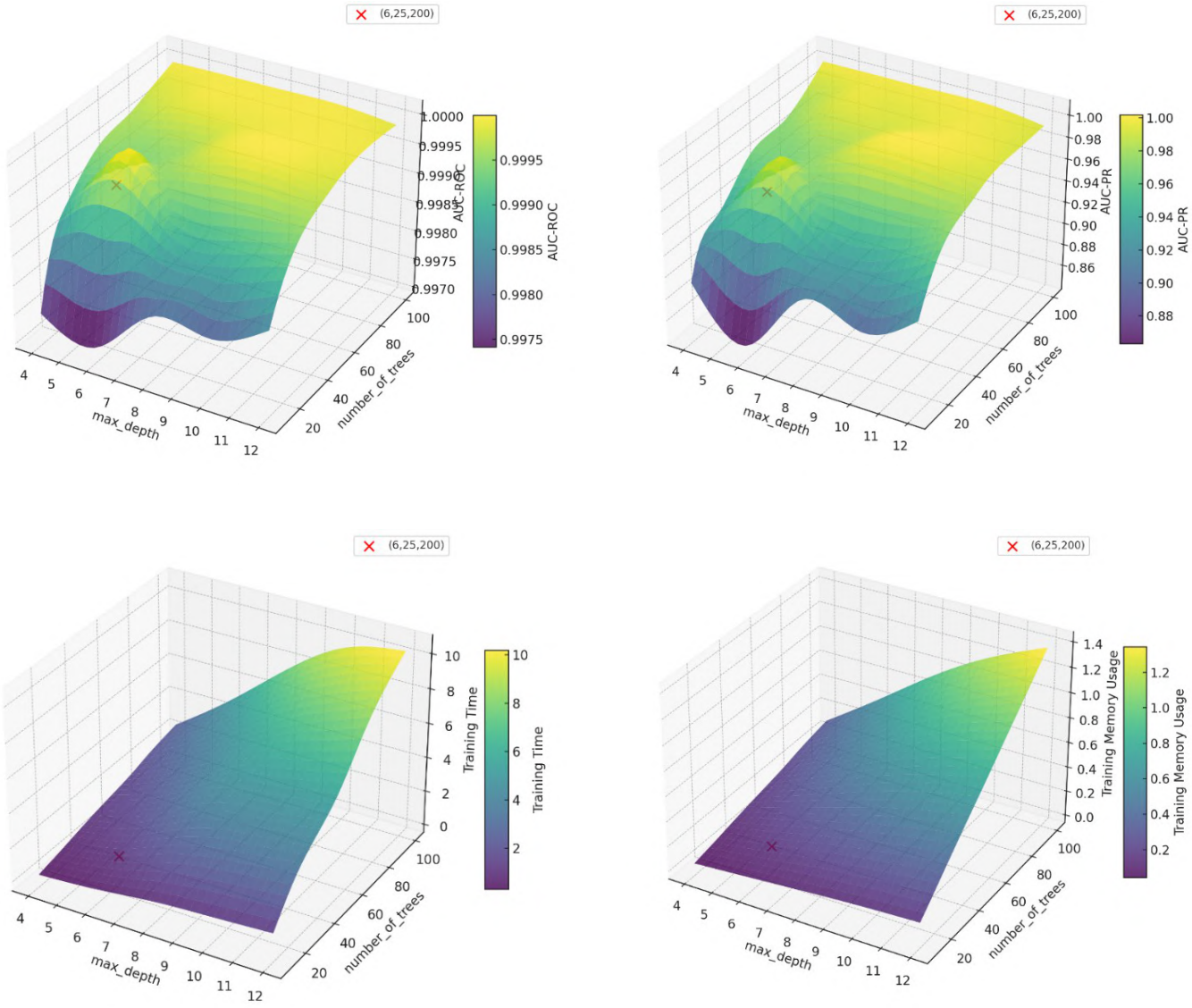
На слици 17, приказана је потрошња меморије (MB) у односу на параметре, где се поново види да повећање сложеност модела (максималне дубине и броја стабала) доминантно утиче на заузеће меморије. С друге стране повећање количине података за обуку доводи до постепенијег раста потрошње меморије.

Слика 17: Утицај појединачних параметара на искоришћење меморије

Графици приказани на сликама 14-17 дају нам целокупну слику о утицају појединачних н хиперпараметара на поанашање система (бар у погледу метрика које смо означили као најбитније). Повећање сложености модела, доприноси бољим резултатима, али улази у засићење након одређене границе, док цена обуке таквих модела у нагло расте са сложеношћу. Са друге стране повећање количине података за обуку (самим тим и диверзитета) нам даје значајна побољшања перформанси, са не тако скупом ценом у виду перформанси. Из тога се да закључити да се оптимална конфигурација под задатим параметрима заправо налази у комбинацији умерене сложености модела и већег броја података за обуку, што се поклапа са нашим емпиријским ибором оптималне тачке E(6,25,200) из скупа експериментално испитаних конфигурација.

4.3.4 Интеракција између параметара

Након анализе утицаја појединачних хиперпараметара на перформансе система, закључено је да повећање количине података неопходних за обуку веома позитивно утиче на перформансе система, без велике цене у виду искоришћења ресурса. Из тога следи питање ако је у експерименталној поставци максималних 200 одбирака за обуку оптимално пре доласка до засићења, како да будемо сигурни да је и сложеност модела оптимална под задатим условима. Како би смо барем стекли интуицију о исправности нашег избора, посматраћемо комбинацију максималне дубине стабла и броја стабала у односу на четири главне метрике.



Слика 18: Главне метрике као функције максималне дубине стабла и броја стабала

Четири графика на слици 18 заједнички илуструју однос између тачности модела и ефикасности у зависности од максималне дубине стабла и броја стабала у шуми. На горња два графика види се да перформансе остају високе у целом простору параметара, са готово оптималним вредностима за умерене дубине (6-8) и средњи број стабала (25-50). Изван овог опсега, додатна комплексност модела доноси само маргиналне добитке. Доња два графика која приказују искоришћење рачунарских ресурса , приказују стабилан и готово линеаран раст у потрошњи ресурса како оба параметра расту, при чему најкомплексније конфигурације имају знатно већу потрошњу меморије и време обуке.

У својој целини, на приказаним графицима јасно се види да након максималне дубине стабла од 8 и броја стабала већег од 50, перформансе улазе у засићење и нису вредне додатне цене обуке модела, за овај сет података. На графицима је приказана и изабрана оптимална тачка E(6,25,200), а на њима се још једном јасно види близина локалном максимуму (који

је маргинално мањи од глобалног максимума), упркос скромном меморијском буџету у односу на већину експерименталних конфигурација.

4.3.5 Поређење перформанси PFLiForest и iForest алгоритама

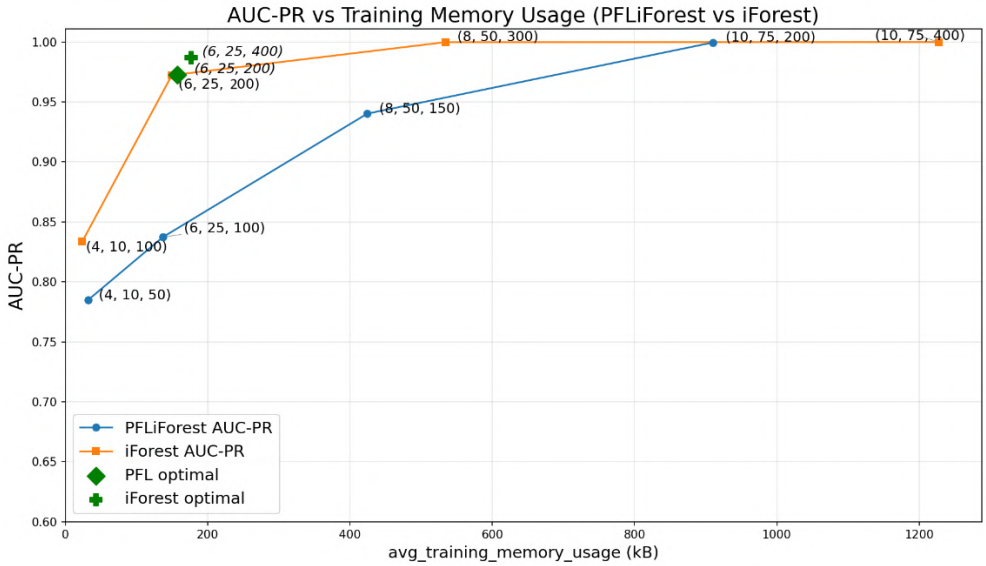
У овом одељку дајемо директно поређење алгоритма изложеног у овом раду (PFLiForest), и основног случаја, тј. обичног алгоритма изолационе шуме (iForest). Док iForest, функционише под предпоставком централизоване доступности података, PFLiForest уводи федеративну парадигму обуке која побољшава приватност података, избегавањем дељења података. У нашем случају посматраћемо следеће сценарије:

- **iForest:** два клијента шаљу своје приватне податке централном серверу који на основу њих прави глобални модел изолационе шуме.
- **PFLiForest:** два клијента задржавају своје приватне податке, учествујући у федеративном процесу прављења глобалног модела, без директног дељења података.

Вредности AUC-ROC неће бити приказане јер оба алгоритма имај вредности ове метрике изнад 96% за све конфигурације параметара, а у већини случајева достижу и 99%. Ово значи да оба модела готово савршено разликују аномалије од нормалних узорака.

На слици 19 приказане су вредности AUC-PR за оба алгоритма у односу на искоришћење меморије. За PFLiForest су приказане тачке A(4, 10, 50), B(6, 25, 100), C(8, 50, 150), D(10, 75, 200), док су за базни алгоритам то тачке A’(4, 10, 100), B’(6, 25, 200), C’(8, 50, 300) и D’(10, 75, 400). Додатна тачка E (6,25,200), која је у претходним одељцима означена као оптимална под условом датом кроз једначину 11, и њен пар E’(6,25,400), су такође приказани на графику.

Као што је и било очекивано, PFLiForest за мање сложене моделе са мање података за обуку има и ниже перформансе од еквивалентних конфигурација обичног алгоритма изолационе шуме. Међутим, ова разлика се смањује и на крају у потпуности нестаје у захтевнијим конфигурацијама.



Слика 19: Поређење перформанси PFLiForest и iForrest алгоритама

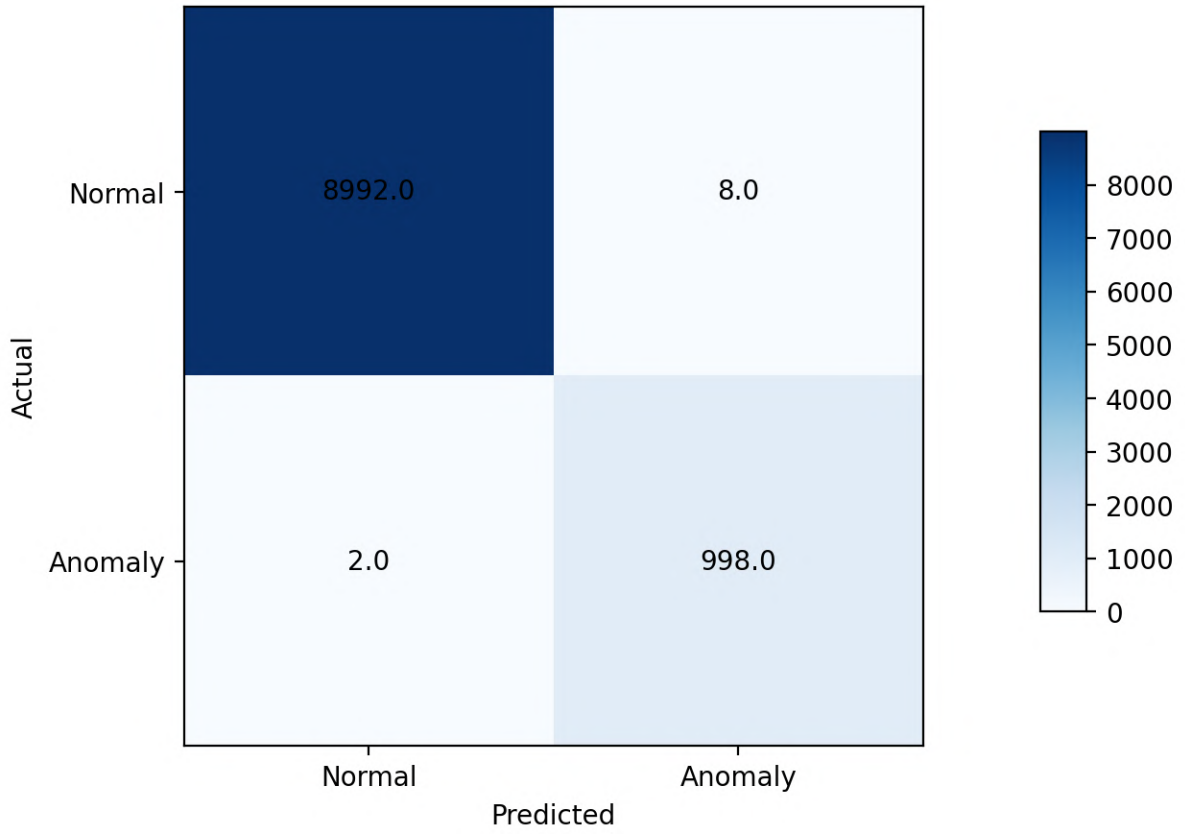
(Уз дозволу аутора адаптирано из [1])

Нижи резултат у левом делу графика очекивани су и директна су последица компромиса направљеног у корист очувања приватности података. Ипак, понашање алгоритма у сложенијим конфигурацијама. Као и постигнути резултати у оптималној тачки, показују веома охрабрујуће одлике. Посебно је значајно што тачка Е и њен пар Е' остварују готово идентичне вредности AUC-PR, што потврђује ефикасност предложеног приступа.

4.3.6 Додатна евалуација у изабраној оптималној тачци

У циљу добијања целокупног увида у перформансе изабране оптималне конфигурације E(6,25,200). конструисали смо матрицу конфузије (слика 20), и израчунали вредности додатних метрика. Она је добијена додатном евалуацијом за вредност прага аномалије од 0.8265 који је изабран као праг за коју смо имали највишу вредност F1 мере (0.994), на синтетичком скупу података (10000 одбирака) направљеном на начин описан у одељку 4.2. Већина нормалних примерака је тачно класификована (8992), док је број лажних позитивних предвиђања минималан (8). Са друге стране готово све аномалије су исправно детектоване (998), уз занемарљиво мали број лажно негативних.

Ови резилтати потврђују да модел остварује високу прецизност (99.17%) и осетљивост (99.8%) што је у складу са високим вредностима AUC-ROC и AUC-PR. Поред тога, стопа лажно позитивних предвиђања је испод 0.1% што додатно указује да је модел стабилан и поуздан за детекцију аномалија.



Слика 20: Матрица конфузије оптималне конфигурације

5. Закључак

У овом раду представљен је PFLiForest, специјализовани алгоритам за детекцију аномалија на основу изолационих шума, развијен у оквиру PTB-FLA окружења. Детаљном анализом резултата у оквиру експерименталне евалуације перформанси и ефикасности система, закључено је да алгоритам представља добар компромис између тачности детекције модела и заузећа рачунарских ресурса. Овакви резултати указују на велики потенцијал овог приступа за примену у интелигентним IoT системима на ивици интернета, фокусираним на приватност података корисника.

Главне предности предложеног приступа су: (i) базираност алгоритма на формално верификованим генеричким алгоритмима, самим тим чинећи PFLiForest коректним по конструкцији (*енгл*. correct by construction); (ii) подршка за додатну обуку и адаптацију на тренутне услове без потребе за ручним означавањем података, што га чини погодним за уређаје на ивици интернета у фази експлоатиације; (iii) могућност размене контекста између клијената, без дељења приватних података, што резултује побољшаном генерализацијом модела и приватношћу података корисника; (iv) постизање релативно високих перформанси модела уз мало заузеће меморије и искоришћење процесорског времена.

С друге стране систем поседује одређена ограничења: (i) алгоритам је евалуиран само за једнодимензионалне скупове података, и неопходно је модификовати га за вишедимензионалне податке; (ii) релативно високи трошкови комуникације и потенцијално велика количина мрежног саобраћаја у фази тренирања модела због високог нивоа сарадње између клијената и сервера. (iii) нешто слабије перформансе у односу на базни алгоритам, које је неопходно надоместити повећаном сложености модела.

Овај рад представља основу за даља истраживања у области примене федеративног учења и алгоритама федеративних изолационих шума у реалним системима, али и за теоријска истраживања усмерена на унапређења перформанси и скалабилности досадашњих алгоритама федеративног учења. У погледу истраживања применљивости система федеративног учења, најприроднији наставак овог рада представља евалуација перформанси система у разним стандардним применама детекције аномалија над вишедимензионалним скуповима података, праћена интеграцијом предложеног приступа у реалне интелигентне системе на ивици интернета. Са теоријског становишта, истраживања би могла бити усмерена ка смањењу комуникационе сложености система, анализа предности и недостатака различитих метода агрегације модела или робустизације система у случају отказа учесника у процесу федеративног учења.

6. Литература

[1] P. Vasiljevic, M. Matic and M. Popovic, "Federated Isolation Forest for Efficient Anomaly Detection on Edge IoT Systems," 2025 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia, 2025, pp. 30-35, doi: 10.1109/ZINC65316.2025.11103552.

[2] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in Proc. of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), JMLR: W&CP volume 54, pp. 1-10, 2017.

[3] P. Kairouz, H. B. McMahan, B Avent, A. Bellet et al., "Advances and Open Problems in Federated Learning", 2019, doi: https://doi.org/10.48550/arxiv.1912.04977.

[4] O. Rana et al., "Hierarchical and Decentralised Federated Learning," arXiv.org, Apr. 28, 2023. https://arxiv.org/abs/2304.14982

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, Jul. 2009, doi: https://doi.org/10.1145/1541880.1541882.

[6] F. T. Liu, K. M. Ting and Z. -H. Zhou, "Isolation Forest," 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008, pp. 413-422, doi: 10.1109/ICDM.2008.17.

[7] M. Popovic, M. Popovic, I. Kastelan, M. Djukic, and S. Ghilezan, A Simple Python Testbed for Federated Learning Algorithms, 2023, in: Proceedings of the 2023 Zooming Innovation in Consumer Technologies Conference, 2023, pp. 148-153, https://doi.org/10.1109/ZINC58345.2023.10173859.

[8] M. Popovic, M. Popovic, I. Kastelan, M. Djukic, and I. Basicevic, “MicroPython Testbed for Federated Learning Algorithms” , 2024, doi: https://doi.org/10.48550/arxiv.2405.09423.

[9] M. Popovic, Github репозиторијум PTB-FLA. Доступно на адреси: https://github.com/miroslav-popovic/ptbfla, (приступљено 28. Јула 2025.)

[10] Yepmo, V., Smits, G., Lesot, M.-J., & Pivert, O., "Leveraging an Isolation Forest to Anomaly Detection and Data Clustering," Data & Knowledge Engineering, vol. 151, no. C, pp. 102302, Август 2025. [Online]. Available: https://doi.org/10.1016/j.datak.2024.102302.

[11] Liu, T., Zhou, Z., & Yang, L., "Layered isolation forest: A multi-level subspace algorithm for improving isolation forest," Neurocomputing, vol. 581, no. C, pp. 127525, May 2024. [Online]. Available: https://doi.org/10.1016/j.neucom.2024.127525.

[12] Li, J., "Federated anomaly detection with Isolation Forest in the IoT network," Ph.D. thesis, Macquarie Univ., 2024. [Online]. Available: https://doi.org/10.25949/25286269.v1Li, J., "Federated anomaly detection with Isolation Forest in the IoT network," Ph.D. thesis, Macquarie Univ., 2024. [Online]. Available: https://doi.org/10.25949/25286269.v1.

[13] Xiang, H., Zhang, X., Xu, X., Beheshti, A., Qi, L., Hong, Y., & Dou, W. (2024). Federated learning-based anomaly detection with isolation forest in the IoT-edge continuum. ACM Transactions on Multimedia Computing, Communications, and Applications, Just Accepted. https://doi.org/10.1145/3702995.

[14] I. Prokić, S. Ghilezan, S. Kašterović, M. Popovic, M. Popovic, I. Kaštelan, Correct orchestration of Federated Learning generic algorithms: formalisation and verification in CSP, in: J. Kofron, T. Margaria, C. Seceleanu (Eds.), Engineering of ComputerBased Systems, Lecture Notes in Computer Science, Vol. 14390, Springer, Cham, 2024, pp. 274–288, https://doi.org/10.1007/978-3-031-49252 5_25.

[15] M. Popovic, M. Popovic, I. Kastelan, M. Djukic, and I. Basicevic, A Federated Learning Algorithms Development Paradigm, in: J. Kofron, T. Margaria, C. Seceleanu (Eds.), Engineering of Computer-Based Systems, Lecture Notes in Computer Science, Vol. 14390, Springer, Cham, 2024, pp. 26–41, https://doi.org/10.1007/978-3-031-49252 5_4.

[16] D. C. Montgomery, *Design and Analysis of Experiments*, 9th ed. Hoboken, NJ, USA: Wiley, 2017.