TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Elmo Egers 211506IABM

Veebirobotite tuvastamine hiire dünaamika abil

Magistritöö

Juhendaja: Avar Pentel
MSc

Tallinn 2024

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Elmo Egers

02.01.2024

Annotatsioon

Magistritöö eesmärk on uurida hiire dünaamika kasutamise võimalusi veebirobotite tuvastamise kontekstis ja välja tuua sellega kaasnevad piirangud.

Veebirobotite tehniline võimekus on viimase aastakümne jooksul oluliselt tõusnud. Seepärast on robotite tuvastamiseks ette nähtud traditsioonilised robotilõksud muutunud oluliselt ebaefektiivsemaks. Probleemi lahendamiseks on üha enam tähelepanu pööratud käitumise dünaamika kasutamisele, mille alla kuulub ka hiire dünaamika.

Töö esmane eesmärk on uurida hiire andmete kogumise ja töötlemise võimalusi. Samuti annab autor ülevaate veebirobotite liikidest, nende tuvastamiseks kasutatavatest meetoditest ja nende puudustest.

Töö teine eesmärk on autori poolt kogutud andmete põhjal läbi viia eksperimendid, mille käigus teostatakse andmete eeltöötlus ja treenitakse masinõppemudelid. Mudelite loomine toimub kahes etapis, millest esimese käigus kasutatakse täielikult ühendatud pärilevivõrku tunnuste ekstraheerimiseks. Teises etapis treenitakse loodud tunnuste põhjal anomaaliatuvastuse mudelid, et hinnata nende kasutatavust veebirobotite tuvastamiseks.

Läbi viidud eksperimendid näitavad, et täielikult ühendatud pärilevivõrgu- ja anomaaliatuvastuse mudeleid kombineerides on võimalik veebiroboteid üksnes hiire andmete põhjal tuvastada. Eksperimentide käigus saavutas parim mudel, mille sisendiks on vaja vähemalt 7,3 sekundi jagu hiire kursori liikumise andmeid, tulemuseks: Kordustäpsus = 0,90; Saagis = 0,88; F1 skoor = 0,88 ja ROC AUC = 0,99.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 9 peatükki, 16 joonist, 16 tabelit.

Abstract

Identifying web bots using mouse dynamics

The aim of this Master's thesis is to explore the possibilities of using mouse dynamics for web bot detection and also bring out its limitations.

The technical capabilities of bots have significantly increased over the last decade. Therefore, traditional bot detection methods have become less effective. This has led to an increase in focus on utilizing behavioral dynamics, including mouse dynamics, to resolve this issue.

The primary objective of the paper is to analyze the methods of mouse data collection and pre-processing. In addition, the author provides an overview of the types of web robots and the methods used to identify them. Also, the shortcomings of those methods are described.

The second goal of the paper is to conduct experiments on the data collected by the author, which includes data pre-processing and machine learning model training. The models are trained in two steps. First, a fully convolutional network is used for feature extraction. And second, anomaly detection models are trained based on the extracted features to assess their effectiveness in identifying web bots.

The experiments show that by combining a fully convolutional network and anomaly detection models, it is possible to detect web robots based only on mouse data. The best model, which requires at least 7.3 seconds of mouse cursor movement data, achieved the following results: Precision = 0.90; Recall = 0.88; F1 score = 0.88 and ROC AUC = 0.99.

The thesis is in Estonian and contains 47 pages of text, 9 chapters, 16 figures, 16 tables.

Lühendite ja mõistete sõnastik

API	*Application programming interface*, rakendusprogrammiliides. Kogum käske, funktsioone ja protokolle, mis spetsifitseerivad tarkvarakomponentide interaktsiooni.
AUC	*Area Under the Curve*, ROC-kõvera alune pindala.
CNN	*Convolutional Neural Network*, konvolutsiooniline närvivõrk
EER	*Equal Error Rate*, võrdne veamäär.
FCN	*Fully Convolutional Network*, täielikult ühendatud pärilevivõrk.
FN	*False Negative*, vale negatiivne.
FP	*False Positive*, vale positiivne.
HTTP	*Hypertext Transfer Protocol*, hüperteksti edastusprotokoll.
JSON	*JavaScript Object Notation*, JavaScripti objektide notatsioon. Lihtne andmevahetusvorming.
LSTM	*Long Short-Term Memory*, pikk lühiajaline mälu.
ReLU	*Rectified Linear Unit*, mittelineaarne aktivatsioonifunktsioon.
REST	*Representational State Transfer*, arhitektuuristiil.
TP	*True Positive*, õige positiivne.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Veebirobotid on automatiseeritud arvutiprogrammid, mille peamiseks eesmärgiks on lihtsustada eelnevalt määratletud tegevuste täitmist. Ühelt poolt kasutatakse neid tuhandete veebilehtede indekseerimiseks, mille tulemusena paraneb ligipääs informatsioonile otsingumootorite kaudu. Teiselt poolt kasutatakse veebiroboteid libakontode loomiseks, reklaami või pahavara levitamiseks, teenustõkkerünnakute tegemiseks ja turvanõrkuste ärakasutamiseks [1, 2, 3].

1.1 Probleemi kirjeldus

Robotite tuvastamiseks veebilehtedel kasutatakse tehnilist lahendust, mida nimetatakse robotilõksuks, ehk *CAPTCHA*-ks. Kõige laialdasemalt on kasutuses traditsioonilised robotilõksud, mille läbimiseks tuleb valida kirjeldusele vastavad pildid, või sisestada etteantud vormi lahtrisse robotilõksu poolt esitatud raskesti arusaadav tekst [4, 5].

Viimase kümne aastaga on toimunud veebirobotite tehnilises võimekuses suur areng. Kui varasemalt olid veebirobotid võimelised teostama lihtsamaid päringuid [6], siis tänapäeval on neil võimekus erinevate tööriistade abil veebilehtedel ringi liikuda. Lisaks sellele on mitmetel veebirobotitel võimekus automatiseeritult robotilõkse lahendada [2]. Seetõttu võib tekkida olukord, kus traditsiooniline robotilõks ei täida enam oma eesmärki. Sellest tulenevalt on veebilehtede autorid ja robotilõksu teenusepakkujad robotilõksus kuvatavaid ülesandeid muutnud üha keerulisemaks, et takistada veebirobotite tööd. See on viinud olukorrani, kus traditsioonilise robotilõksu lahendamine on keeruline inimestele, kuid lihtne robotitele [5].

Eelpool kirjeldatud probleemi lahendamiseks on üha enam tähelepanu pööratud inimese käitumisele iseloomulike omaduste tuvastamisele, ehk käitumise dünaamikale. Käesolevas töös keskendutakse peamiselt hiire dünaamikale, mis on üks käitumise dünaamika osa [7].

Käitumise dünaamika kasutamine robotilõksudes võimaldab robotilõksudest eemaldada traditsioonilised visuaalsed ülesanded. Selle asemel jälgitakse kasutaja käitumist veebilehel ja tuvastatakse nende andmete põhjal inimesele iseloomulikud tunnused, mida iseloomustavad juhuslikkus ja ebatäpsus. Veebirobotite poolt teostatavad tegevused on enamasti sujuvad ja ettearvatavad, kuid leidub ka erandeid [6].

1.2 Autori roll

Magistritöö autor on käesolevas töös teinud koostööd kolme ettevõttega, kes on huvitatud kasutajasõbralikest ja kaasaegsetest veebirobotite tuvastamise vahenditest, mis ei riiva kasutajate privaatsust ja oleksid võimalusel ettevõtte siseselt majutatavad. Koostöö vorm seisneb andmete kogumise võimaldamises ja järelduste ning tulemuste tutvustamises, milleni töö käigus jõuti.

Autor on teadlik turul pakutavatest robotituvastuse teenustest ja nende võimalustest. Seetõttu ei ole töös pakutavate lahenduste esmane eesmärk turul pakutavate teenuste väljavahetamine, vaid tuvastusmeetodite täiendamine.

1.3 Eesmärk

Töö eesmärgiks on luua hiire dünaamikal põhinevad masinõppe mudelid, mis võimaldavad veebilehe külastuse ajal kogutud andmete põhjal teha kindlaks, kas tegu on inimese või robotiga.

Eesmärgini jõudmiseks on püstitatud järgmised küsimused, millele leitakse vastus töö käigus:

1. Milliseid hiire dünaamika andmeid tuleks probleemi lahendamiseks koguda ja kuidas tuleks neid edasiseks kasutamiseks töödelda?

2. Milliste meetodite kasutamist veebirobotite tuvastamiseks hiire dünaamika põhjal tuleks kaaluda?

3. Kui suurel hulgal andmeid (sekundites) on loodavate mudelite kasutamiseks minimaalselt tarvis?

4. Milliseid probleeme valminud mudelite kasutuselevõtt aitab ära hoida või leevendada?

Antud töö võib huvi pakkuda andmeanalüütikutele ja tarkvaraarendajatele, kes robotite tuvastamise vahenditega kokku puutuvad, või kaaluvad selle kasutusele võtmist.

1.4 Töö struktuur

Töö esimene peatükk kirjeldab probleemi olemust ja selgitab autori rolli käesolevas töös. Peatüki viimases osas püstitatakse eesmärgid, millele leitakse vastus töö käigus.

Teises peatükis on antud ülevaade hiire andmete kasutamisest kasutajate käitumise analüüsimiseks. Samuti on kirjeldatud veebiroboteid ja nende tuvastamise vahendeid, et probleemi olemus oleks üheselt mõistetav.

Kolmandas ja neljandas peatükis on kirjeldatud töös kasutatud vahendeid ja andmeid.

Viiendas peatükis on tutvustatud andmete kogumiseks loodud rakendust ja andmete eeltöötlust. Järgnevalt on välja toodud tehislike trajektooride genereerimiseks kasutatud vahendid.

Kuuendas peatükis on põhjalikult kirjeldatud täielikult ühendatud pärilevivõrgu ja anomaaliatuvastuse mudelite loomist.

Seitsmendas peatükis on esitatud töös saadud tulemused.

Kaheksandas peatükis on esitatud ülevaade valminud lahendusest. Lisaks on analüüsitud töö käigus loodud mudeleid ja võrreldud neid teiste töödega. Peatüki teises osas on selgitatud töö käigus tekkinud väljakutseid ja jätkutegevusena planeeritud töid. Samuti on lühidalt hinnatud loodud lahenduse kasulikkust ja kasumlikkust.

Viimases peatükis on kokku võetud töö eesmärk, peamised tulemused ja järeldused.

2 Taust ja kirjanduse ülevaade

Selles peatükis tutvustatakse esmalt hiire dünaamika olemust ja selle kasutamist kasutaja käitumise analüüsimiseks. Seejärel kirjeldatakse tehislike hiire andmete kasutamist täiendavalt kasutajate käitumise andmetele. Järgnevas kahes alapeatükis antakse ülevaade veebirobotite liikidest ja nende tuvastamiseks kasutatavatest meetoditest ning tuuakse välja nende meetodite puudused. Viimases alapeatükis tutvustatakse tulemuste valideerimise meetodeid.

2.1 Hiire dünaamika kasutamine kasutaja käitumise analüüsimiseks

Hiire dünaamika kasutamist kasutaja käitumise analüüsimiseks on laialdasemalt uuritud alates 21. sajandi algusest. Seda on kasutatud nii veebirobotite- kui kasutajakontode kaaperdamise tuvastamise kontekstis, kus ühe kasutaja tavapärast käitumist on võrreldud teiste sama süsteemi kasutajatega [7, 8, 9, 10].

Hiire dünaamika kasutamine eelmainitud probleemide lahendamiseks eeldab esmalt andmete kogumist. Hiire kasutamisega tekitatakse nii operatsioonisüsteemi- kui ka veebibrauseri tasandil mitmeid sündmusi, mis iseloomustavad hiire kasutamist. Nendeks on hiire nupuvajutused ja kursori asukoht ekraanil konkreetsel ajahetkel. Kasutaja käitumise analüüsimiseks kuulab veebirakendus selliseid sündmusi pealt ja edastab need veebiserverisse käitumise analüüsimiseks [6, 11].

Andmete kogumise metoodika jaguneb kaheks. Esimene, ehk juhendatud andmete kogumine, põhineb konkreetse ülesande korduval täitmisel, mida teostavad katses osalevad isikud. Ülesandeks võib olla näiteks spetsiaalse veebivormi täitmine või hiire liigutamine etteantud trajektoori mööda. Ülesande ajal kogutakse veebilehel andmeid kasutaja käitumise kohta, mida kasutatakse seejärel seatud eesmärgi täitmiseks. Ülesande lahendamise pikkus või korduste arv sõltub testi eesmärgist ja vaja minevast andmete hulgast. Teine viis andmete kogumiseks on juhendamata andmete kogumine. Sellisel juhul lisatakse andmete koguja olemasolevale veebilehele ja jälgitakse kasutajate käitumist nende tegevusse sekkumata [12].

Kogutud andmed puhastatakse ja luuakse tunnused peamiselt kolmel meetodil: traditsiooniliselt, närvivõrkude abil ja piltide abil.

Traditsioonilisel viisil tunnuste loomine tähendab üksikute tunnuste, näiteks hiire kursori suund, kiirus, kiirendus jms, loomist käsitsi. Sellisel meetodil tunnuste loomine on ajakulukas ja ei pruugi kõiki hiire omadusi piisavalt hästi kirjeldada. Lisaks eeldab see suurt teadlikkust kõigi võimalike hiire liikumise ja nupuvajutustega seotud omapärade kohta [13, 14, 8, 10, 15, 16].

Närvivõrkude abil tunnuste loomine võimaldab tunnuste loomise protsessi oluliselt lihtsustada ja kiirendada. Sellisel meetodil on tunnuste loomiseks kasutatud peamiselt konvolutsioonivõrke, mille viimase konvolutsiooni- või ahenduskihi väljundit, ehk pudelikaela tunnuseid, kasutatakse uute mudelite sisendiks. Samuti on leitud, et optimaalse närvivõrgu mudeli abil loodud tunnused võivad anda paremaid tulemusi kui traditsioonilisel meetodil loodud tunnused [9, 10, 17, 18].

Piltide abil tunnuste loomine põhineb hiire kursori trajektooride kujutamisel piltidel (Joonis 10), millest seejärel tuletatakse tunnused konvolutsioonilise närvivõrgu (CNN) abil. Kuna hiire kursori trajektoorid talletatakse üksikute punktidena, mis sisaldavad x- ja y-koordinaadi asukohta konkreetsel ajahetkel, kantakse need punktid piltidele vastavalt segmenteerimise protsessile, mida kirjeldatakse järgmises lõigus. Punktide vahele kantakse lisaks sirgjooned, mis annavad ülevaate hiire kursori trajektoorist. Mõnel juhul on kursori kiiruse informatsiooni edasi andmiseks sirgjooned ja punktid pildile kantud teise värviga [19, 8].

Kõigi kolme meetodi ühiseks osaks on andmete segmenteerimise protsess. Andmete segmenteerimine võimaldab kursori liikumise trajektoorid ja nupuvajutused eraldada üksikuteks loogilisteks osadeks. Segmenteerimine põhineb kahel faktoril. Esimene faktor, ehk ajakünnis, määratleb maksimaalse lubatud aja, mis on kahe järjestikuse sündmuse vahel lubatud. Teine faktor määratleb andmete töötlemise liigi.

Ajakünnise suurima väärtuse määratlemist on käsitletud väga erinevalt. Üldjuhul on ajakünnise väärtus fikseeritud (näiteks 100ms ja 400ms), kuid mõnel juhul arvutatakse see iga kasutaja sessiooni kohta dünaamiliselt. Juhul, kui kahe järjestikku asetseva sündmuse vahele jääv aeg jääb alla ajakünnise, arvatakse nad ühe loogilise osa sisse. Vastasel juhul on tegemist sündmustega, mida käsitletakse eraldiseisvana. Dünaamiliselt kasutaja sessiooni ajakünnise arvutamine eeldab veebilehe külastuse ajal kogutud andmete analüüsimist ühe tervikuna iga

kasutaja kohta. Seetõttu võib selle rakendamine praktikas osutuda keeruliseks ja ressursinõudlikuks. Ühes loogilises osas paiknevad andmed töödeldakse ühe näitena selliselt, et leitakse x- ja y-koordinaatide vahe järjestikku asetsevate sündmuste vahel [19, 16, 6].

Lisaks ajakünnise piirangule segmenteeritakse sündmuseid ka hiirega teostatud tegevuse või sündmuste arvu alusel.

Tegevuse alusel sündmuste segmenteerimisel võetakse arvesse nii hiire nupuvajutusi kui ka kursori liikumist. Kõige lihtsama meetodi puhul segmenteeritakse sündmused ainult ajakünnise ja hiire kursori liikumise alusel. Teise meetodi puhul teostatakse segmenteerimist näiteks mõne (veebi)elemendi lohistamise alusel (nupuvajutus, millele järgnevad kursori liikumise sündmused ja seejärel nupu vabastamise sündmus). Samuti võib kasutada meetodit, mis teostab segmenteerimist kahe nupuvajutuse vahele jäävatest kursori liikumise sündmustest [15, 16, 6, 13, 19].

Sündmuste arvu alusel sündmuste segmenteerimisel arvestatakse enamasti ainult ajakünnise piirangut ja hiire kursori liikumise sündmusi. Liikumise sündmused segmenteeritakse fikseeritud pikkusega vektoriteks, mille moodustavad x- ja y-koordinaatide vahe, või mõni muu tunnus järjestikku asetsevate sündmuste vahel. Vektori pikkus ei ole standardiseeritud, mistõttu on selle pikkus erinevates töödes erinev. Näiteks on kasutatud vektoreid pikkusega: 100, 128, 200, 300 ja 500 [17, 16, 20, 21].

2.2 Tehislike hiire andmete kasutamine

Mitmes töös [21, 22, 23, 24, 25] on edukalt kasutatud tehislike hiire andmete (trajektooride) loomise meetodit. See loob sobivad tingimused, et mudelite treenimisel, testimisel ja valideerimisel oleks võimalik arvesse võtta erinevaid ründeid, või antud juhul erineva tehnilise võimekusega veebiroboteid. Lisaks on tehislikke andmeid võimalik luua suures koguses väikese ajakuluga. Tehislike hiire kursori trajektooride loomise aluseks on enamasti Bézier' kõverad, ehk parameetrilised kõverad, mille kumerus sõltub kontrollpunktidest [26].

2.3 Veebirobotite liigid

Veebiroboteid võib kirjeldada nii neile seatud eesmärgi- kui ka tööpõhimõtete alusel.

Eesmärgi alusel liigitatakse veebirobotid positiivset- ja negatiivset mõju avaldavateks. Positiivset mõju avaldavate robotite toel paraneb veebilehtede kraapimise (*web crawling*) tulemusel ligipääs informatsioonile otsingumootorite kaudu. Lisaks saab neid kasutada ka teenuste jõudluse ja korrasoleku kontrollimiseks [3, 2, 27].

Negatiivset mõju avaldavad veebirobotid teostavad teenustõkkerünnakuid, kaaperdavad kasutajakontosid, koguvad pahavara saatmise eesmärgil e-posti aadresse, tuvastavad teenustes turvanõrkusi nende ärakasutamiseks ja võimaldavad sotsiaalmeedias automatiseeritult pahatahtlikku sisu postitada [2, 8, 27, 1, 3].

Tööpõhimõtete alusel võib veebirobotid jagada kolmeks:

1. **Elementaarsed robotid.** Kõige laialdasemalt kasutusel olev robot, mis oma ülesannete täitmiseks ei kasuta veebilehitsejaid. Selle asemel on kasutusel skriptid ja HTTP päringuid teostada võimaldavad rakendused, näiteks *curl* ja *wget*. Sellist liiki veebilehe külastus erineb teistest näiteks selle poolest, et külastuse ajal ei registreerita lehel ühtegi sündmust, mis iseloomustaksid tavapärast külastust. Nende hulka kuuluvad ka hiire- ja klaviatuuri sündmused [2, 6].

2. **Mõõdukad robotid.** Mõõdukad robotid kasutavad veebilehtede külastamiseks spetsiaalseid graafilise režiimita (*headless*) veebilehitsejaid, mida on võimalik skriptide abil automatiseerida. Eelnimetatud brauserid võimaldavad veebilehtedel käivitada JavaScripti [28] koodi ja tekitada tavapärasele külastusele iseloomulikke sündmusi, sealhulgas hiire kursori liikumise sündmusi. Erinevalt täiustatud robotitest ei kasutata mõõdukate robotite puhul erilahendusi inimese käitumise jäljendamiseks [2, 6, 29].

3. **Täiustatud robotid.** Tehniliselt kõige keerukamad robotid, mis kasutavad oma eesmärgi täitmiseks spetsiaalseid automatiseeritud brausereid, või tavapäraseks kasutuseks mõeldud brausereid (Google Chrome, Mozilla Firefox jmt), kuhu on paigaldatud lisatööriistad nende kontrollimiseks [2, 23, 25, 6, 29].

Täiustatud robotid kasutavad inimese käitumise jäljendamiseks peamiselt kahte erinevat meetodit. Esimene meetod hõlmab spetsiaalsete teekide või skriptide kasutamist, mis võimaldavad mitmeid süsteeme petta ja jätta mulje, et hiire kursori liikumine, klaviatuuri nuppude klahvivajutused ja mitmed teised sündmused on tekitatud inimese poolt [2, 25, 23, 30, 6, 21, 19, 31]. Tehislikke hiire liikumise trajektoore luua võimaldavaid teeke on kasutatud ka käesolevas töös.

Teine meetod hõlmab inimese veebikasutuse salvestamist konkreetsel veebilehel, kuhu soovitakse hiljem automatiseeritud päringuid hakata teostama. Meetodi peamine eelis esimese meetodiga võrreldes seisneb selles, et käitumise jäljendamise asemel korduv kasutatakse konkreetset tegevust, näiteks veebivormi täitmist, mis on varasemalt läbi viidud inimese poolt. Seetõttu on selliste robotite tuvastamiseks vajalik kasutada mitmeid erinevaid meetodeid [8, 23, 6].

Lisaks eelpool kirjeldatud kahele meetodile võivad täiustatud veebirobotid kasutada täiendavaid meetodeid tuvastamise vältimiseks. Nende hulka kuuluvad näiteks IP-aadressi vahetamine, proksiserverite kasutamine, brauseri ja arvuti seadistuse kohta valeinfo jagamine ning robotilõksude automaatne lahendamine. Täiendavalt võidakse tehislikult piirata roboti kiirust, et mitte liigselt teiste kasutajate veebiliiklusest eristuda [2, 3, 31].

2.4 Veebirobotite tuvastamiseks kasutatavad meetodid ja nende puudused

Veebirobotite tuvastamiseks on viimase paarikümne aasta jooksul leitud mitmeid erinevaid võimalusi. Järgnevalt on analüüsitud nelja kõige tuntumat meetodit, mida on aktiivselt kasutatud ka robotite tuvastamise teenust osutavate ettevõtete poolt [31].

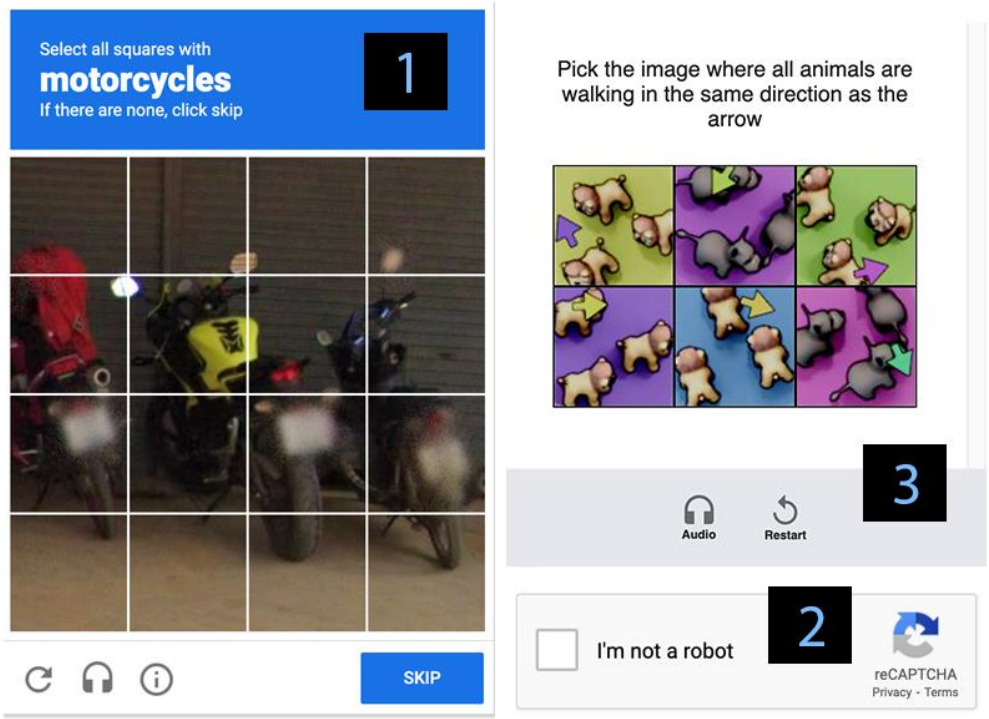
Esimene meetod põhineb kasutaja käitumise jälgimisel veebiserverisse tehtavate päringute alusel. Veebiserveri logid sisaldavad kronoloogilises järjekorras informatsiooni selle kohta, milliseid lehekülgi või faile veebilehitseja serverist küsis. Logidesse talletava informatsiooni hulka võib kuuluda näiteks IP aadress, staatuskoodid, kasutajaagent jmt. Kogutud infot analüüsitakse näiteks statistilist analüüsi kasutades, tuvastades kõrvalekalded andmete pärimise kiiruses või -liigis [27, 3, 32, 33, 34, 35, 36].

Logide analüüsimise suurimaks puuduseks veebirobotite tuvastamise kontekstis seisneb selles, et veebiserveri logid sisaldavad liiga vähe informatsiooni teenust kasutava seadme ja

tarkvara kohta. Täiustatud robotid võivad piirata oma tegevuse kiirust ja võltsida kasutajaagenti, et inimest paremini jäljendada. Sellisel juhul ei erine robot päringute tegemise osas inimesest, ehk robot jääb tuvastamata [24, 31].

Teine meetod seisneb päringute ja brauseri API-de kaudu ligipääsetavate atribuutide põhjal unikaalse võtme, ehk brauseri sõrmejälje loomisel. Atribuutide hulka kuuluvad näiteks kasutajaagent, ekraanisuurus, fontide nimekiri ja palju muud. Teatud juhtudel on võimalik tuvastada ka kasutatav operatsioonisüsteem ja riistvara. Kuna selliseid atribuute on üle saja, võimaldab valitud atribuutide põhjal koostatud sõrmejälg eristada ka väikest hulka mõõdukaid- ja täiustatud roboteid [25, 37, 31, 38, 3, 39].

Peamised puudused brauseri sõrmejälje kasutamisel veebirobotite tuvastamiseks seisnevad asjaolus, et brauseri API-de kaudu ligipääsetavate atribuutide väärtusi on võimalik võltsida. Täiustatud robotitel on tehniline võimekus automatiseerimiseks kasutatavaid lisatööriistu peita ning veebisaidile välja paista kui tavaline kasutaja. Kaasaegsemad veebirobotid kasutavad lisaks iOS ja Android brauserite privaatsusvõimalusi ära selleks, et API-de kaudu väljastatavate atribuutide hulka piirata ja tuvastust vältida [2, 39, 31, 3, 25].



Joonis 1. Robotilõksud (1., 2. Google reCAPTCHA, 3. Sony Captcha).

Kolmas meetod, ehk täielikult automatiseeritud avalik Turingi test (*CAPTCHA*, Joonis 1) on üks tuntumaid viise veebirobotite tuvastamiseks. Selle peamiseks põhjuseks võib lugeda tema kuluefektiivsust. *CAPTCHA* läbimiseks tuleb läbida arvuti poolt esitatud test. Eelnimetatud test peaks olema lihtne inimese jaoks, kuid keeruline robotite jaoks. Testi sisuks võib olla näiteks piltide äratundmine, kuvatava teksti dešifreerimine, etteantud elementidel klõpsamine või tugeva taustamüraga helifailist arusaamine. Sõltuvalt *CAPTCHA* tüübist võidakse taustal täiendavalt rakendada ka kasutaja käitumist analüüsivaid lahendusi [29, 40, 4].

CAPTCHA-de suurimaks probleemiks on asjaolu, et testid, mis olid varasemalt lihtsad inimestele ja keerulised robotitele, on tänapäeval keerulised inimestele, kuid lihtsad robotitele. Masinõppe kiire arenguga on loodud mitmeid tehnilisi vahendeid piltidest, tekstist ja helifailidest info kätte saamiseks ilma inimese abita. Seetõttu on *CAPTCHA*-des kuvatavad testid sageli seadistatud raskemini arusaadavamaks, et veebirobotid ei oleks võimelised neid automaatselt lahendama. See võib omakorda põhjustada olukorra, mis takistab erivajadustega inimestel teatud teenuseid kasutamast. [29, 25, 41, 31].

Neljas meetod seisneb kasutaja käitumise analüüsimisel ja sellest ebatavaliste mustrite tuvastamisel. Ebatavalisi mustreid või kõrvalekaldeid on võimalik tuvastada anomaalia- või uudsuse tuvastamise (*novelty detection*) mudelitega. Selliseid mudeleid treenitakse üheklassiliselt, ehk antud juhul ainult inimeste poolt tekitatud andmete põhjal. Ebatavaliste mustrite alla kuuluvad näiteks ühtlased ja sirgjoonelised hiire kursori liikumised ja klaviatuuri klahvivajutused, kus mitme järjestikku alla vajutatud klahvi vahele jääv aeg on konstantne, ehk mittejuhuslik [22, 20, 42, 35].

Kuigi kasutaja käitumise analüüsimine võib olla väga efektiivne meetod robotite tuvastamiseks, leidub ka selle meetodi puhul puudusi, millega tuleb arvestada. Täiustatud veebirobotid võivad kasutaja käitumist taas-esitada, või võltsida keerukamate tehislike hiire kursori liikumistega ja klahvivajutustega [24, 21, 22, 31, 43, 10].

2.5 Tulemuste valideerimise meetodid

Hiire andmed liigituvad käitumusliku biomeetria valdkonda. Biomeetrilistele süsteemidele on omane teha kahte liiki otsuseid: lugeda isik legitiimseks või mitte [13]. Käesoleva töö kontekstis võrreldakse inimestele ja veebirobotitele omaseid käitumismustreid, kasutades

hiire andmetel treenitud mudeleid, mille üheks headuse hindamise meetodiks on ROC-analüüs [17, 44].

ROC-analüüs võimaldab graafiliselt, ehk ROC-kõveral, hinnata mudelite tundlikkust ja spetsiifilisust erinevatel lõikepunktidel. ROC-kõvera all paiknev ala (AUC) võimaldab hinnata mudeli täpsust tervikuna. AUC väärtus paikneb vahemikus 0 kuni 1. Mida suurem on AUC väärtus, seda parem on mudeli headus. Binaarsete mudelite täpsuse hindamisel tuleb lisaks arvestada, et AUC väärtus 0.5 on võrdeline juhusliku ennustusega [44].

Lisaks ROC AUC väärtusele kasutatakse mudelite hindamiseks kordustäpsuse (*precision*), saagise (*recall*) ja F1 skoori väärtusi.

Täpsus näitab, kui suur osa positiivseks ennustatud tulemustest on ka tegelikult positiivsed [45]. Täpsuse valem on järgmine [46]:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

Saagis näitab positiivsete tulemuste osakaalu kõigist tõestest positiivsetest tulemustest [45]. Saagise valem on järgmine [46]:

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

F1 skoor võtab arvesse nii täpsust kui saagist, et anda tulemus, mis võtaks arvesse mõlema mõõdiku omapärasid [45]. F1 skoor leitakse järgmiselt [46]:

$$F1\ score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (3)$$

3 Töös kasutatud vahendid

Töös kirjeldatud katsed on teostatud Apple MacBook Pro sülearvutil, millel on Apple M1 Pro protsessor ning 32GB muutmälu. Rakenduste ja skriptide loomiseks on kasutatud programmeerimiskeeli TypeScript, Go ja Python [47, 48, 49]. Masinõppega seotud koodi haldamiseks on loodud Jupyter Notebook [50] töölehed. Kogutud andmete talletamiseks on kasutatud MongoDB [51] andmebaasi.

Lisaks on kasutatud järgmisi Python teeke andmete ettevalmistamiseks, mudelite treenimiseks ja valideerimiseks ning testimiseks:

- *Pandas* – Andmeteaduses ja masinõppes laialdaselt kasutuses olev teek, mis võimaldab suuri andmekogusid analüüsida ning neid töödelda [52].
- *Numpy* – Laialdaselt kasutuses olev teek, mis lihtsustab tööd andmemassiividega [53].
- *Matplotlib* – Visualiseerimistööriist [54].
- *Scikit-learn* – Kogum masinõppetööriistadest, mis lihtsustavad klassifitseerimise, otsustusmetsa, tugivektormasina ja teiste mudelite loomist. Lisaks sellele pakub teek mitmeid vahendeid andmete ettevalmistamiseks ning visualiseerimiseks [55].
- *TensorFlow* – Ökosüsteem, mis pakub tehnilisi lahendusi nii andmete ettevalmistamiseks kui ka masinõppemudelite treenimiseks ja nende rakendamiseks [56].
- *PyOD* – Kogum anomaaliatuvastuse vahenditest, mis lihtsustavad tööd mudelitega, pakkudes universaalset API kihti nende kasutamiseks [57].

4 Töös kasutatud andmed

Käesolevas töös uuritakse veebirobotite tuvastamise võimalusi hiire dünaamika abil, kasutades autori poolt kogutud- ja tehislikult loodud andmeid. Tehislike andmete loomise vajadus seisneb täiustatud veebirobotite jäljendamises. Järgnevalt on kirjeldatud mõlemat andmeliiki.

4.1.1 Veebilehtedelt kogutud andmed

Andmete kogumine toimus kolme erineva ettevõtte (edaspidi Ettevõte 1, Ettevõte 2 ja Ettevõte 3) veebisaitide olemasolevatel vormidel ja sisulehtedel ajavahemikul 01.11.2022 – 21.07.2023. Andmete kogumise meetodiks oli valitud juhendamata andmete kogumine. Vormid ja sisulehed, kuhu andmekoguja paigaldati, kooskõlastati veebisaitide eest vastutavate isikutega. Samuti lepiti kokku andmete töötlemise põhimõtted. Seadustest ja määrustest tulenevad võimalikud nõuded, mis käsitlevad lõppkasutajalt nõusoleku küsimist seda liiki andmete kogumise kohta, ei kuulu antud magistritöö skoopi ning jäid ettevõtete kanda.

Esimene veebisait võimaldab Ettevõttel 1, mis tegeleb hulgimüügiga, hallata tooteid, tellimusi ja muud ettevõtte jaoks olulist infot. Tegemist on kinnise halduskeskkonnaga, mille kasutajaskonna moodustavad ettevõtte 46 töötajat. Kogutud andmete osas on võimalik kinnitada, et veebirobotite või muude skriptide kasutamine andmete kogumise hetkel oli välistatud. Andmeid on kogutud ajavahemikul 01.11.2022 – 01.06.2023.

Teine veebisait, kuhu andmekoguja paigaldati, on sama ettevõtte e-pood, kus klientidel on võimalik tutvuda toodete ja ettevõtte poolt pakutava informatsiooniga. Registreeritud klientidel on võimalik esitada ka tellimusi. Andmete kogumist on teostatud halduskeskkonnaga samal ajavahemikul sisulehtedel, millel on keskmiselt 500 unikaalset külastust kuus.

Kolmas osa andmetest koguti Ettevõte 2 veebisaidil, kus külastajatel on võimalik tutvuda lehel pakutava informatsiooniga ning vajadusel teha erinevaid infopäringuid. Lehekülgedel, kus andmeid koguti, on keskmiselt 10 000 unikaalset külastust kuus. Andmeid on kogutud ajavahemikul 13.02.2023 – 15.06.2023.

Neljas osa andmetest on kogutud Ettevõttes 3 kasutuses olevast sisemisest veebirakendusest, mida kasutatakse asutusesisese info haldamiseks ja levitamiseks. Sarnaselt eelpool kirjeldatud

halduskeskkonnale on ka sellel veebisaidil veebirobotite kasutamine välistatud. Andmete kogumist on teostatud ajavahemikul 03.07.2023 – 21.07.2023.

4.1.2 Tehislike hiire kursori trajektooride andmed

Tehislike trajektooride loomiseks kasutas autor erinevaid Python skripte ja teeke, mille abil luuakse viite erinevat tüüpi tehislikud hiire trajektoorid.

BezierCurve [58] – Bézier' kõver tüüpi hiire trajektoorid. Sobivad jäljendamaks mõõdukaid veebiroboteid. Tulemuseks on sujuvad hiire trajektoorid, mille kumerust saab mõjutada kontrollpunktide kaudu [26].

GhostCursor [59] – Trajektooride generaator, mis kasutab Bézier' kõveraid ja Fittsi seadust [60], et välja selgitada vajalike kontrollpunktide arv ning liikumiskiirus. Trajektoorid sobivad lihtsamate täiustatud veebirobotite jäljendamiseks.

HumanCurve [58] – *BezierCurve* täiendus, mis võimaldab lisaks kontrollpunktidele simuleerida ebatäpsust ja liikumiskiirust. Sobilik täiustatud veebirobotite jäljendamiseks.

WindMouse [61] – Algoritm realistlike hiire trajektooride loomiseks, mida on kasutatud inimese käitumise simuleerimiseks näiteks võrgumängudes. Sarnaselt eelmisele sobivad loodud trajektoorid täiustatud veebirobotite jäljendamiseks.

SapiAgent [18] – Teadustöö tulemusel avaldatud vahend inimese käitumisele sarnaste tehislike hiire trajektooride genereerimiseks. *SapiAgent-i* kasutamiseks treenitakse esmalt inimeste andmete põhjal süvaõppemudel, mida rakendatakse tehislike hiire trajektooride loomiseks. Antud lahendus on sobilik täiustatud veebirobotite jäljendamiseks [22].

Käesolevas töös kasutatakse *SapiAgent* mudeli treenimiseks *SapiMouse* andmeid [62], mida kasutasid ka M. Antal, K. Buza ja N. Fejer oma teadustöös.

SapiMouse andmestik on loodud 2020. aastal ja sisaldab infot 120 inimese kohta, kelle vanus on vahemikus 18 kuni 53 aastat. Andmete kogumiseks sooritasid kasutajad kahel korral konkreetseid ülesandeid etteantud aja jooksul. Andmeid koguti kahes osas, millest esimese sessiooni pikkus oli 3 minutit ja teise sessiooni pikkus 1 minut [62].

5 Andmete kogumine ja ettevalmistamine

Järgnevas peatükis on kirjeldatud andmekoguja rakendust, andmete eeltöötlemist ja tehislike andmete genereerimist, mida kasutatakse mudelite loomiseks ja nende testimiseks.

5.1 Andmekoguja rakendus

Andmete kogumiseks on loodud andmekoguja rakendus, mis koosneb klient- ja serverrakendusest. Andmete kogumist teostatakse sessioonipõhiselt. Igale kasutajale genereeritakse veebisaiti külastades sessioonivõti, mis talletatakse brauseri küpsisesse. Küpsis aegub veebilehitseja akna sulgemisel, või 1h möödumisel, misjärel genereeritakse külastajale uus sessioonivõti. Eelnimetatud võtme kasutamine loob sobivad tingimused konkreetse külastusega seotud sündmuste analüüsimiseks, kuid ei võimalda tuvastada konkreetset isikut.

Rakenduse poolt kogutavad andmed, mille kasutamist käesolevas töös kaaluti, on näidatud järgnevas tabelis. Muud informatsiooni, näiteks seadme IP-aadress või veebilehel sisestatud andmed, kasutaja kohta ei koguta.

Tabel 1. Kogutavad andmed.

Atribuut	**Kirjeldus**
type	Sündmuse liik. Väärtusteks saavad olla: • **mm** – Hiire liikumise sündmus. • **md** – Hiire nupu alla vajutus. • **mu** – Hiire nupu vabastamine pärast alla vajutamist. • **dc** – Hiire nupul topeltklõpsu tegemine. • **cm** – Kontekstimenüü avamine (tavaliselt hiire parempoolse nupu vajutamine). • **ts** – Puutesündmuse algus (näiteks näpu asetamine puuteekraanile). • **tm** – Puutepunkti sündmus. • **te** – Puutesündmuse lõpp (näiteks näpu eemaldamine puuteekraanilt) • **wu** – Lehe kerimine üles (üldjuhul hiire rullikut kasutades). • **wd** – Lehe kerimine alla (üldjuhul hiire rullikut kasutades).
client_x	Hiire kursori või puutepunkti x koordinaat veebilehitseja akna suhtes. Väärtus 0 tähistab veebilehitseja akna vasakut serva.
client_y	Hiire kursori või puutepunkti y koordinaat veebilehitseja akna suhtes. Väärtus 0 tähistab veebilehitseja akna ülemist serva.

button	Arvuline väärtus alla vajutatud hiire nupu kohta. Väärtusteks saavad olla: • **0** – Hiire põhinupp või vaikeväärtus juhul, kui ükski nupp ei olnud alla vajutatud olekus. • **1** – Hiire keskmine nupp (kui on olemas). • **2** – Hiire parempoolne nupp. • **3** – Hiire neljas nupp (kui on olemas). • **4** – Hiire viies nupp (kui on olemas).
identifier	Puutepunkti identifikaator, mis võimaldab eristada samaaegselt toimuvaid puutesündmusi.
client_started_at	Ajatempel andmete kogumise algusaja kohta.
frame_created_at	Ajatempel konkreetse sündmuse talletamise hetkel.
created_at	Sündmuse ajatempel andmete talletamise hetkel serverrakenduses.
href	Lehekülje aadress, kus sündmus registreeriti.
session_id	Unikaalne sessioonivõti.

5.1.1 Klientrakendus

Klientrakenduse lähtekood on kirjutatud TypeScript-s, mis transpileeritakse veebilehitsejas käivitamiseks JavaScript-ks kasutades Rollup-i[1]. Brauseris andmete kogumiseks kasutatakse JavaScripti meetodit *addEventListener*, mis võimaldab ettemääratud sündmusi (Tabel 2) pealt kuulata ja neile reageerida. Valitud sündmustele (mm, tm, wu, wd) reageerimine on piiratud ~13 sündmusele sekundis (käivitamine minimaalselt iga 75ms järel), et minimiseerida andmekoguja ressursivajadust vanematel seadmetel ning ühtlasi vähendada sündmuste hulka.

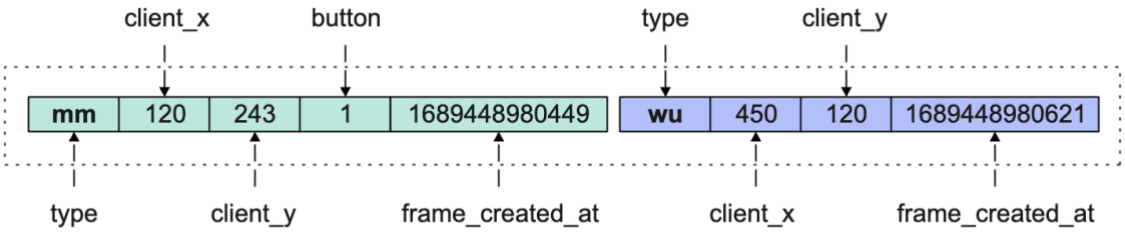
Sündmustelt kogutud info talletatakse massiivi, mille sisu edastatakse serverrakendusele iga 250 sündmuse täitumise hetkel. Erandiks on juhtumid kui kasutaja liigub teisele lehele (või veebisaidile), kuhu ei ole andmekogujat paigaldatud. Sellisel juhul edastatakse andmed antud ajahetkel massiivis olevate sündmuste kohta.

Tabel 2. Pealt kuulatavad veebibrauseri sündmused.

Sündmus	**Kirjeldus**
beforeunload	Sündmus kutsutakse välja vahetult enne seda, kui kasutaja liigub näiteks ühelt leheküljelt teisele. Kasutatakse erandjuhtudel andmete edastamise alustamiseks.
mousemove	Kasutatakse hiire kursori koordinaatide talletamiseks konkreetsel ajahetkel.
touchmove	Kasutatakse puutepunkti(de) informatsiooni talletamiseks konkreetsel ajahetkel.

[1] https://rollupjs.org/

Serverrakendusele saadetavate andmete mahu vähendamiseks talletatakse- ja edastatakse sündmuste andmed pesastatud massiivina. Iga sündmuse liigi massiiv sisaldab erinevat hulka atribuute. Joonis 2 selgitab eelnimetatud massiivi olemust. Kogutud info edastatakse JSON-vormingus üle REST API liidese, kasutades JavaScripti funktsiooni *navigator.sendBeacon* (kui veebibrauser seda toetab) või *XMLHttpRequest* meetodeid.



Joonis 2. Sündmuste andmed pesastatud massiivina.

5.1.2 Serverrakendus

Serverrakenduse lähtekood on kirjutatud programmeerimiskeeles Go. Serverrakenduse ülesanne on klientrakenduse serveerimine ja andmete vastu võtmine REST API liidese kaudu. Andmete talletamiseks on kasutatud MongoDB andmebaasi, kuid tehniliselt ei ole piiranguid mõne teise andmebaasimootori kasutusele võtmiseks.

Serverrakenduses on defineeritud järgmised teenuse lõpp-punktid:

1. **/api/v1/frame** – HTTP POST meetod sündmuste andmete vastu võtmiseks klientrakenduselt ja nende talletamine andmebaasi.
2. **/assets/v1/arc** – HTTP GET meetod klientrakenduse JavaScripti koodi serveerimiseks.

5.1.3 Andmekoguja lisamine veebilehele

Pärast mõlema rakenduse kokku ehitamist paigaldatakse see vabalt valitud (virtuaal)serverisse, millel on võimekus jooksutada kompileeritud binaarset rakendust. Antud juhul on kasutatud kolme Linux-põhist virtuaalserveri instantsi, mis majutati ettevõtetes, kus andmete kogumist teostati.

Kui rakendus on käivitatud, saab andmekoguja lisamiseks veebilehele kasutada järgmist HTML koodi:

```
<script>
   function ready() {
      window.arc.startCollector();
   }
</script>

<script src="https://<rakenduse aadress>/assets/v1/arc" async defer
onload="ready()"></script>
```

Joonis 3. Andmekoguja lisamine veebilehele.

5.2 Andmete eeltöötlus

Andmete eeltöötluse käigus eraldatakse kõikide Tabel 1 kirjeldatud sündmuste hulgast esmalt sellised sessioonid, mis ei sisalda puutesündmusi. Puutesündmuste olemasolu võib viidata asjaolule, et hiire kursori liikumise sündmus on tekitatud puutepaneeli abil, näiteks nutitelefonil. Selliste sündmuste analüüs ei kuulu antud töö skoopi ning tuleb käsitleda eraldi. Järgmisena on sündmuste hulgast välja filtreeritud hiire liikumise sündmused (*type=mm*), mida kasutatakse järgmises andmete töötlemise etapis.

Masinõppemudelite loomiseks on eeltöötluse käigus saadud andmed segmenteeritud fikseeritud suurusega tunnusvektoriteks, kasutades iga sündmuse x- ja y-koordinaati ning ajatemplit (*client_x, client_y, frame_created_at*). Optimaalse tunnusvektori pikkuse väljaselgitamiseks on segmenteerimist läbi viidud seitsmes erinevas pikkuses – 16, 24, 32, 48, 64, 96 ja 128.

Tunnusvektorid (7) koosnevad sessioonipõhiste andmete põhjal loodud uutest atribuutidest, milleks on järjestikuste sündmuste x- ja y-koordinaatide vahe absoluutväärtus (4), (5). Lisaks kalkuleeritakse järjestikuste sündmuste ajavahe (6). Kaks järjestikust sündmust ühes sessioonis moodustavad seega ühe atribuutide hulga ($|\Delta x_i|, |\Delta y_i|, \Delta t_i$). Absoluutväärtuse kasutamine tuleneb eeltööna läbi viidud katsetest, mis näitas mudelite üldist kvaliteedi tõusu absoluutväärtuse kasutamisel.

$$|\Delta x_i| = |client\_x_{i+1} - client\_x_i| \quad (4)$$

$$|\Delta y_i| = |client\_y_{i+1} - client\_y_i| \quad (5)$$

$$\Delta t_i = frame\_created\_at_{i+1} - frame\_created\_at_i \quad (6)$$

Leitud ajavahemiku Δt_i alusel jäetakse kõrvale kõik sellised atribuutide hulgad, mis ületavad ettemääratud ajakünnist. Käesolevas töös on katsed läbi viidud kolme ajakünnisega – 250ms, 500ms ja 1000ms, et hinnata kas tunnusvektorite hulga tõus ajakünnise suurendamise tulemusena võiks loodavate mudelite kvaliteedile mõju avaldada.

$$fv = [|\Delta x_1|, \dots, |\Delta x_n|, |\Delta y_1|, \dots, |\Delta y_n|], kus\ n = 16, 24, 32, 48, 64, 96, 128 \quad (7)$$

Kõige lühema tunnusvektori ($n = 16$) loomiseks on tarvis 16 atribuutide hulka ehk 17 hiire liikumise sündmust, mille ajakünnis Δt jääb etteantud piiridesse. Ideaaltingimustes kirjeldab selline atribuutide hulk ligikaudu 1.3 sekundi jooksul toimunud hiire püsivat liikumist. Selleks, et maksimeerida atribuutide hulkade kasutamist, luuakse tunnusvektorid toorandmetest juhul kui esindatud on vähemalt 75% vaja minevatest andmetest, mis kõige lühema tunnusvektori puhul on võrdeline kaheteistkümnega. Puuduvad andmed asendatakse nulliga ning lisatakse vektori väärtuste $|\Delta x|\ ja\ |\Delta y|$ lõppu.

Loodud tunnusvektorid talletatakse CSV failidesse, mille nimetus sisaldab infot andmestiku nime, tunnusvektori tüübi ja -pikkuse ning ajakünnise kohta (näiteks *dx-dy_andmestik-1_FV64_DTS500.csv*). Lisaks tunnusvektoritele talletatakse eraldi failidesse info iga tunnusvektoris kasutatud hiire kursori trajektoori algus- ja lõpp-punkt ja tema sessiooni nimi.

5.3 Tehislike hiire kursori trajektooride loomine

Tehislike trajektooride loomiseks eraldatakse esmalt 1. ja 2. andmestikust 50% sessioonide nimedest, mida kasutatakse hiire kursori algus- ja lõpp-punktide lugemiseks eelmises etapis loodud CSV failidest. Reaalsete algus- ja lõpp-punktide kasutamine tehislike trajektooride loomiseks võimaldab vältida probleeme, mis on seotud trajektoori pikkusega. Näiteks võib tehislikult selliseid punkte luues tekkida olukord, kus loodava trajektoori algus- ja lõpp-punkt asuvad üksteisest kaugusel, mida ei ole valitud tunnusvektori pikkuse ja ajakünnise juures hiirt liigutades võimalik saavutada.

Pärast eelnimetatud andmete sisse lugemist algab erinevat tüüpi tehislike trajektooride loomine. Kõik trajektooride generaatorid kasutavad algandmetena samu algus- ja lõpp-punkte. Seetõttu on unikaalsuse tõstmiseks kasutusele võetud meetod, mis nihutab igat koordinaati 100 piksli piires x- ja y-teljel vahetult enne tehisliku trajektoori loomist.

Esimese ja teise andmestiku põhjal loodud tehislikud trajektoorid talletatakse üksteisest eraldi, millest esimest (edaspidi koondnimega „Tehislik 1“) kasutatakse edaspidiselt anomaaliatuvastuse mudelite testimiseks ja teist (edaspidi koondnimega „Tehislik 2“) FCN mudelite treenimiseks.

5.3.1 *BezierCurve* trajektoorid

BezierCurve trajektooride loomiseks on kasutatud *pyclick* paketist klassi *BezierCurve* funktsiooni *curvePoints*, mille argumentideks on loodavate tehislike punktide kogus ja kontrollpunktide definitsioonid. Tehislike punktide kogus määratakse iga tunnusvektori pikkuse jaoks eraldi. Kontrollpunkte luuakse kolme tüüpi – ühe, kahe ja kolme juhusliku punktiga, mis valitakse esialgsete algus- ja lõpp-punkti vahele jäävatest väärtustest mõlema telje kohta eraldi. Kontrollpunktide esimeseks ja viimaseks väärtuseks on alati esialgsed algus- ja lõpp-punktid, mida on 100 ühiku piires nihutatud.

Trajektooride genereerimisel tagastab *curvePoints* funktsioon uued x- ja y-koordinaadid, mis seejärel segmenteeritakse tunnusvektoriteks alapeatükis 5.2 kirjeldatud meetodil. Tunnusvektorid talletatakse seejärel CSV failidesse iga andmestiku, kontrollpunkti tüübi, tunnusvektori pikkuse ja -tüübi ning ajakünnise kohta eraldi (näiteks: *dx-dy_andmestik-1_synth-bc_triple_FV48_DTS500.csv*).

5.3.2 *GhostCursor* trajektoorid

GhostCursor trajektooride loomiseks on kasutatud paketi *python_ghost_cursor* funktsiooni *path*, mille argumentideks on trajektoori algus- ja lõpp-punkt. Täiendavalt on käesoleva töö jaoks kolmandaks argumendiks lisatud loodavate tehislike punktide minimaalne kogus, mis teegi siseselt seadistab minimaalse genereeritavate punktide koguse. See on vajalik selleks, et teek genereeriks tehislikud trajektoorid tunnusvektori pikkuste 96 ja 128 jaoks.

Trajektooride genereerimisel tagastab *path* funktsioon uued x- ja y-koordinaadid, mis segmenteeritakse tunnusvektoriteks ja talletatakse CSV failidesse sarnaselt *BezierCurve* meetodile.

5.3.3 *HumanCurve* trajektoorid

HumanCurve trajektooride loomiseks on kasutatud *pyclick* paketist klassi *HumanCurve*, mis initsialiseeritakse kahe argumendiga, milleks on trajektoori algus- ja lõpp-punkt. Uute koordinaatide genereerimiseks kutsutakse välja funktsioon *generateCurve*, mida on võimalik

peenhäälestada 12 erineva parameetri alusel, mis mõjutavad genereeritavaid trajektoore. Antud juhul on katsete tulemusena kasutatud kahte parameetrit. Esimene neist on genereeritavate punktide arv (*targetPoints*), mille aluseks on tunnusvektori pikkus. Teine parameeter on sõlmede arv (*knotsCount*), mis leitakse valemiga, kus sõlmede arv tunnusvektori pikkuse suurenedes võib kasvada:

```
knotsCount = randint(2, math.ceil((tunnusvektori_pikkus / 128) * 8) + 1)
```

Joonis 4. Sõlmede arvu leidmine *HumanCurve* funktsioonile.

Funktsioon *generateCurve* tagastab uued x- ja y-koordinaadid, mis segmenteeritakse tunnusvektoriteks ja talletatakse CSV failidesse sarnaselt eelmistele trajektooride generaatoritele.

5.3.4 *WindMouse* trajektoorid

WindMouse trajektooride loomiseks on kasutatud avatud lähtekoodiga funktsiooni *wind_mouse*, mille parameetriteks on trajektoori algus- ja lõpp-punkt, gravitatsiooniline jõud (*G_0*), tuule jõud (*W_0*), maksimaalne sammu pikkus (*M_0*) ja distants, kus tuule käitumine muutub (*D_0*). Eeltööna läbi viidud katsete tulemusena on muudetud kahe parameetri väärtusi selliselt, et genereeritavad trajektoorid oleksid inimeste omadele võimalikult sarnased:

```
W_MIN = math.ceil((tunnusvektori_pikkus / 128) * 16)
W_MAX = math.ceil((tunnusvektori_pikkus / 128) * 75)
G_0 = randint(6, 12)
W_0 = randint(W_MIN, W_MAX)
```

Joonis 5. *WindMouse* trajektoori generaatori parameetrite väärtuste valemid.

Funktsiooni tulemiks on uued x- ja y-koordinaadid, mis segmenteeritakse tunnusvektoriteks ja talletatakse CSV failidesse sarnaselt eelmistele.

5.3.5 *SapiAgent* trajektoorid

SapiAgent trajektooride loomiseks on esmalt juhendi [18] järgi treenitud *SapiAgent* autokooder. Selleks, et autokoodrit tehislike trajektooride loomiseks kasutada, tuleb sisendandmed ette valmistada. Andmete ettevalmistamiseks genereeritakse trajektoori algus- ja lõpp-punkti vahele uued andmepunktid, mis asuvad üksteisest võrdsel kaugusel. Visuaalselt kujutatuna moodustab see ühtlase sirge joone. Punktide koguseks on 128, mis tuleneb autokoodri treenimise protsessist.

Pärast uute andmepunktide genereerimist teisendatakse need kahemõõtmeliseks massiiviks, mille esimene veerg sisaldab x- ja teine y-koordinaate. Seejärel antakse massiiv autokoodrile sisendiks ja väljundiks saadakse uued x- ja y-koordinaadid, mis segmenteeritakse erineva pikkusega tunnusvektoriteks ja talletatakse CSV failidesse. Erinevalt eelmistest trajektooride generaatoritest kasutatakse *SapiAgent* trajektooride sisendiks ainult tunnusvektori pikkuse $n = 128$ algandmeid. Lühemate tunnusvektori pikkuste loomiseks kasutatakse sama autokoodri väljundit, millest luuakse lühemad tunnusvektorid.

6 Mudelite loomine

Töös püstitatud eesmärkide täitmiseks kirjeldatakse käesolevas peatükis täielikult ühendatud pärilevivõrgu ja anomaaliatuvastuse mudelite loomist.

Mudelid luuakse kõikide tunnusvektori pikkuste ja ajakünniste kohta, et hinnata, millist ajakünnist tuleks andmete töötlemiseks kasutada ja kui suurel hulgal andmeid on mudelite kasutamiseks minimaalselt tarvis.

6.1 Täielikult ühendatud pärilevivõrk *(FCN)*

Käesolevas töös kasutatakse täielikult ühendatud pärilevivõrku hiire kursori liikumise andmetest oluliste tunnuste ekstraheerimiseks läbi pudelikaela tunnuste. FCN mudeleid luuakse kokku 21 tk.

6.1.1 Tunnusvektorite kasutamine sisendandmetena

Mudelite loomiseks laaditakse esmalt eelmistes etappides loodud tunnusvektorid *Pandas DataFrame* muutujasse. Mudeli tasakaalu säilitamiseks liigitatakse kõik andmetes esinevad variatsioonid eraldi klassidesse (0 – 7), mis sisaldavad võrdses koguses andmeid. Variatsioonideks on inimeste andmete põhjal loodud tunnusvektorid ja kõik tehislikult genereeritud andmete põhjal loodud tunnusvektorid.

Andmete koguse määramisel on arvestatud kõikide tunnusvektori pikkuste võrreldavust ühe ajakünnise raames. Kuna töö üheks eesmärgiks on välja selgitada minimaalne sündmuste hulk mudelite kasutamiseks, on sisendandmete koguse määramisel aluseks võetud sündmuste hulk. Kõikide FCN mudelite sisendandmete kogus on sündmuste koguarvu mõistes ligilähedane või samaväärne, kuid tunnusvektorite koguarvu mõistes erinev. Näiteks üks tunnusvektor, mille pikkus $n = 128$, on loodud 129 sündmuse põhjal. Sama hulga sündmuste juures luuakse 7 tunnusvektorit, mille pikkus $n = 16$.

6.1.2 Mudelite treenimise protsess

Mudelite treenimiseks on loodud funktsioon *train_model*, millel on 6 parameetrit: *data*, *fv_size*, *dts*, *filter_sizes*, *kernel_sizes* ja *n_dimensions*. *Data* kaudu antakse edasi sisendandmed, mida on kirjeldatud eelmises alapeatükis. Parameetrid *fv_size* ja *dts* määratlevad tunnusvektori pikkuse ja ajakünnise. Parameetrid *filter_sizes* ja *kernel_sizes*

määratlevad järjestikuste ühedimensiooniliste konvolutsiooniliste kihtide hüperparameetrite väärtused, milleks on filtrite arv ja konvolutsiooni suurus. Viimane parameeter *n_dimensions* seadistab dimensioonide koguse, mis antud juhul on alati 2 ja tähistab tunnusvektoris esinevaid eri liiki andmeid ($|\Delta x|$ *ja* $|\Delta y|$ väärtused).

Näiteks, tunnusvektori suuruse 64, ajakünnise 500 ja kahe konvolutsiooni kihiga mudeli treenimine, mille filtrite arv ja konvolutsiooni suurus on esimesel kihil 32 ja 4 ning teisel kihil 64 ja 8, käivitatakse järgmiselt:

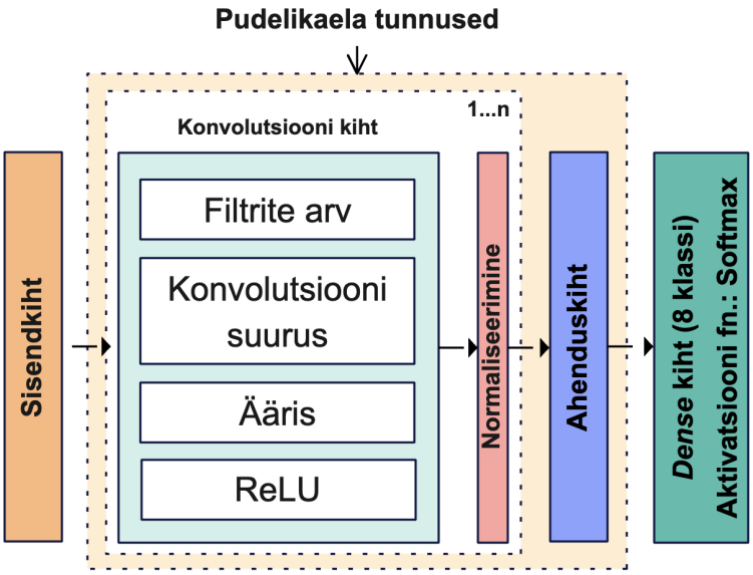
```
train_model(<andmed Pandas DataFrame kujul>, 64, 500, [32, 64], [4, 8], 2)
```

Joonis 6. FCN mudeli treenimise funktsiooni parameetrid.

Funktsiooni **esimeses etapis** sisendandmed normaliseeritakse Z-skoor meetodit kasutades. Järgmiseks leitakse klasside arv, mis antud juhul on 8 ja defineeritakse mudeli nimi, mis võtab arvesse kõiki eelnevalt kirjeldatud parameetreid. Näiteks: *fv-64—dts-500—kernel-4.8.6—filter-256.256.64*.

Teises etapis kodeeritakse klassid binaarseteks vektoriteks, mille pikkus on võrdeline klasside kogusega, kasutades *sklearn* paketi *OneHotEncoder* klassi. Näiteks, esimene klass „0“ teisendatakse vektoriks $[1, 0, 0, 0, 0, 0, 0, 0]$, teine klass „1“ vektoriks $[0, 1, 0, 0, 0, 0, 0, 0]$ jne. Seejärel restruktureeritakse andmed kolmemõõtmeliseks massiiviks, mis on üheks FCN mudeli treenimise eelduseks. Restruktureerimine toimub järgmiselt: andmete hulk ehk ridade arv jääb samaks, tulpade koguseks on tunnusvektori pikkus ning sügavuseks (pikkuseks) on 2, ehk omaduste arv ühes tunnusvektoris. Lisaks sisendandmed segatakse ja 15% eraldatakse mudeli valideerimiseks sama paketi funktsiooni *train_test_split* abil.

Funktsiooni **kolmandas etapis** konverteeritakse treening- ja valideerimisandmed *Tensorflow* andmestikuks, mis võimaldab *Tensorflow-d* kasutades andmeid efektiivsemalt kasutada ja töödelda. Konverteerimiseks kasutatakse funktsiooni *from_tensor_slices*.



Joonis 7. FCN mudel.

Joonis 7 annab ülevaate **neljandas etapis** loodava *Tensorflow* järjestikuse mudeli (*Sequential model*) kohta, mille kihtide arv sõltub funktsiooni sisendiks antud filtrite ja konvolutsiooni suuruse väärtustest. Mudeli esimeseks kihiks on sisendkiht (*Input*), mis määratleb sisendandmete kuju ja vastab eelpool kirjeldatule. Järgmised $1 \ldots n$ kihti on ühedimensioonilised konvolutsiooni kihid (*Conv1D*), millele järgneb normaliseerimise kiht (*BatchNormalization*). Igale konvolutsiooni kihile määratakse filtri- ja konvolutsiooni suurus vastavalt funktsiooni parameetritele. Lisaks kasutatakse ReLU aktivatsioonifunktsiooni ja äärist (*padding*) väärtusega „*same*“. Mudeli konvolutsiooni kihte kasutatakse peatükis 6.2 pudelikaela tunnuste (*bottleneck features*) leidmiseks.

Mudeli eelviimaseks kihiks on ahenduskiht (*GlobalAveragePooling1D*), mis vähendab andmete hulka, säilitades olulisemad tunnused. Viimaseks, ehk väljundkihiks on tihenduskiht (*Dense)*, mis koondab eelmiste kihtide informatsiooni ja võimaldab tulemi jagada klassideks. Viimases kihis kasutatakse *softmax* aktivatsioonifunktsiooni ja ühikute arvuks on klasside koguarv. Seejärel mudel kompileeritakse, kasutades kaofunktsiooni väärtuseks *categorical_crossentropy* ja optimiseerija väärtuseks *Adam*-it.

Viimases, ehk **viiendas etapis**, alustatakse mudeli treenimist, kasutades *Tensorflow* funktsiooni *fit*. Funktsiooni sisendiks on järgmised parameetrid: treeningandmed, valideerimisandmed, epohhide arv ja tagasikutsed (*callbacks*). Epohhide arvuks on määratud 1000, et võimaldada mudelil saavutada minimaalne kadu.

Tagasikutseid on defineeritud kolm. Esimene, ehk varajane treeningu lõpetamine (*EarlyStopping*), jälgib treenimise käigus mudeli kadu valideerimisandmetel ja peatab treenimise protsessi kui kadu ei vähene 30 epohhi jooksul. Teine, ehk kontrollpunkti automaatne salvestus (*ModelCheckpoint*) jälgib, sarnaselt esimesele tagasikutsele, mudeli kadu valideerimisandmetel ja salvestab mudeli seisu iga kord kui kadu väheneb. Viimane tagasikutse vähendab mudeli õppimiskiirust 0.15 ühiku võrra, kui mudeli kadu ei vähene 5 epohhi jooksul. Minimaalseks õppimiskiiruseks on seadistatud $1.5 \cdot 10^{-6}$.

6.1.3 Kihtide arvu- ja hüperparameetrite häälestamine

Optimaalse kihtide- ja filtrite arvu ning konvolutsiooni suuruse leidmiseks on kasutatud nn. toore jõu meetodit. Filtrite suuruseid 32, 64, 128, 256 ja konvolutsiooni suuruseid 2, 4, 6, 8, 10 kombineerides on loodud kõik võimalikud kombinatsioonid, mida kasutatakse ühe kuni kolme konvolutsiooni kihiga mudeli treenimiseks. Igale kihile rakendatakse kõiki kombinatsioone. Treenimise kiirendamiseks kasutatakse $\approx 10\%$ andmetest ja ühte ajakünnist – 500. Lisaks treenitakse mudeleid eelmises alapeatükis kirjeldatud meetodil paralleelselt mitme protsessori tuuma peal.

6.2 Anomaaliatuvastuse mudelid

Anomaaliatuvastuse mudelite põhjal hinnatakse, kas hiire trajektoorid kuuluvad inimesele, ehk normaalklassi (negatiivne klass), või robotitele, ehk anomaalsesse klassi (positiivne klass). Seetõttu pööratakse erilist tähelepanu saagise tulemusele, et minimiseerida valenegatiivseid (normaalklassi ennustamist anomaalsesse klassi). Antud juhul loeb töö autor usaldusväärseks selliseid anomaaliatuvastuse mudeleid, mille saagis ja F1 skoori väärtus on $\geq$ 0.8.

Anomaaliatuvastuse mudelite treenimiseks ja testimiseks kasutatakse *PyOD* teeki. Töö autor viis eeltööna läbi katsed kõikide potentsiaalselt sobivate anomaaliatuvastuse (uudsuse tuvastamise) vahendite tuvastamiseks ja selekteeris tulemuste põhjal välja 3 parimat: KPCA, OCSVM ja LOF. Eelnevale tuginedes luuakse anomaaliatuvastuse mudeleid kokku 63 tk (3 mudelit iga tunnusvektori pikkuse ja ajakünnise kohta).

KPCA (*Kernel Principal Component Analysis*), ehk kernel peakomponentide analüüs täiendab standardset peakomponentide analüüsi, võimaldades mittelineaarse dimensionaalsuse vähendamist kernelite abil. Meetodi põhimõte seisneb standardsete

vahenditega mitteeristatavate andmete projekteerimist suurema mõõtmega ruumi, kus andmed on üksteisest paremini eristatavad [55, 57, 63].

LOF (*Local Outlier Factor*) on tiheduspõhine anomaaliatuvastuse meetod, mis kasutab erindite tuvastamiseks andmepunktide omavahelist tihedust. LOF algoritmi kõrge väärtus andmepunktides viitab punktide madalale tihedusele, ehk erindite olemasolule. [64, 65, 57]

OCSVM (*One-Class Support Vector Machine*), ehk üheklassiline tugivektormasin on kauguspõhine anomaaliatuvastuse meetod, mis eraldab erindid normaalväärtustest hüpertasandi abil [57].

6.2.1 Täielikult ühendatud pärilevivõrgu mudelite kasutamine pudelikaela tunnuste loomiseks

FCN mudelitest pudelikaela tunnuste ekstraheerimiseks esmalt normaliseeritakse tunnusvektorid Z-skoor meetodit kasutades ja seejärel restruktureeritakse nad kolmemõõtmeliseks massiiviks, mis moodustab FCN mudeli sisendi.

Ekstraheerimiseks laaditakse failist sisse valitud FCN mudel, millelt eemaldatakse viimane tihenduskiht (*Dense layer)*. Järgmiseks kutsutakse välja mudeli *predict* funktsioon, mille sisendiks on restruktureeritud tunnusvektorid. Funktsiooni väljundiks saadakse pudelikaela tunnused, mida kasutatakse seejärel anomaaliatuvastuse mudelite loomiseks.

6.2.2 Mudelite treenimise protsess

Sarnaselt FCN mudeli loomisele on ka anomaaliatuvastuse mudelite treenimise lihtsustamiseks loodud funktsioon *train_model*, mille peamisteks parameetriteks on tunnusvektori pikkus, ajakünnise väärtus ja treenitav mudel (KPCA, OCSVM, LOF). Täiendavalt on lisatud parameetrid mudeli nimetuse (logimise eesmärgil) ja treening- ning testandmete koguse määratlemiseks. Viimased kaks võimaldavad mudeleid valideerida sama hulga sündmuste juures.

Funktsiooni *train_model* **esimeses etapis** laaditakse tunnusvektorid failidest *Pandas DataFrame* muutujatesse, mille järel teisendatakse need pudelikaela tunnusteks täielikult ühendatud pärilevivõrgu abil. Treening-, test- ja tehislike andmete pudelikaela tunnused talletakse eraldi muutujatesse.

Teises etapis treenitakse üheklassiline mudel *fit* funktsiooni abil, kasutades sisendandmeteks treeningandmete pudelikaela tunnuseid.

Funktsiooni viimases, ehk **kolmandas etapis** kombineeritakse inimeste hiire trajektooride pudelikaela tunnused ja tehislikult genereeritud hiire trajektooride pudelikaela tunnused üheks terviklikuks testandmestikuks, millest 50% moodustavad inimeste poolt tekitatud andmed ja 50% tehislikud andmed. Testimise eesmärgil leitakse igale kirjele tulemus mudeli funktsiooni *decision_function* abil ja ennustus (*label*) funktsiooni *predict* abil. Saadud tulemused edastatakse *sklearn* paketi funktsioonidele *classification_report* ja *roc_auc_score*, mudelite kvaliteedi analüüsimiseks.

Mudeli rakendamiseks väljaspool magistritööd tuleks kasutada ainult *decision_function* funktsiooni, mis tagastab tulemused arvuliste väärtustena. Inimese või roboti tuvastamiseks saadud väärtustest, võrreldakse neid vabalt valitud lävega (*threshold*), millest suuremate väärtuste puhul on tegemist robotiga.

6.2.3 Hüperparameetrite häälestamine

Mudelite- ja hüperparameetrite hulga ja nende häälestamisele kuluva suure ajamahu tõttu ei kuulu käesoleva töö skoopi parimate hüperparameetrite väärtuste leidmine. Selle asemel on hüperparameetrite väärtusi kohandatud manuaalselt iga tunnusvektori pikkuse kohta. Kõikide mudelite hüperparameetrid seadistatakse selliselt, et vältida inimeste kategoriseerimist robotiks. Selle arvelt võib teatud juhtudel tõenäosus tehislike trajektooride, ehk robotite kategoriseerimisel inimesteks suureneda, kuid see on antud juhul aktsepteeritav risk.

7 Tulemused

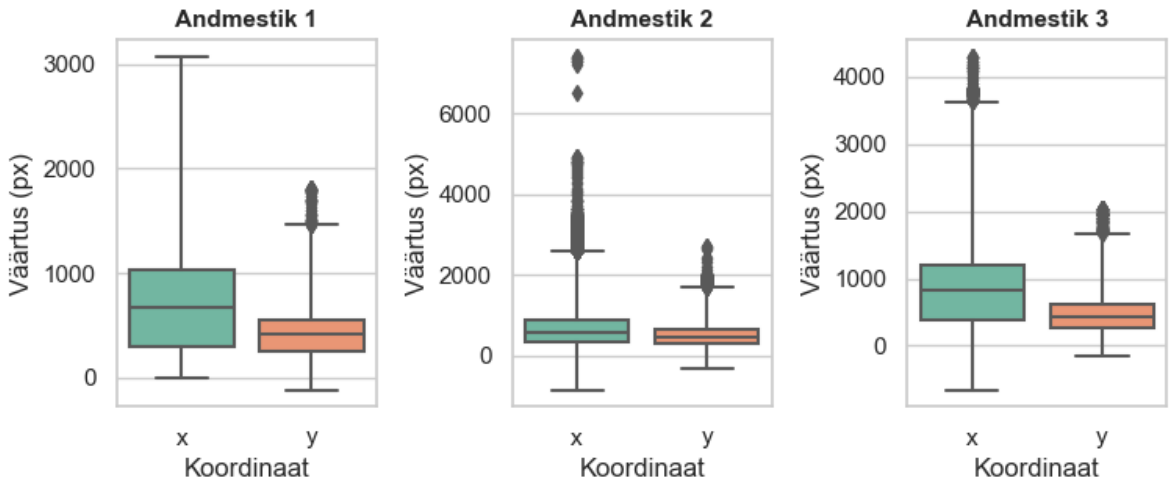
Selles peatükis antakse esmalt ülevaade kogutud andmetest ja nende põhjal loodud tunnusvektoritest. Seejärel kirjeldatakse töös kasutatud hiire kursori trajektoore jooniste abil. Järgmiseks tutvustatakse täielikult ühendatud pärilevi võrgu mudeleid, millele järgneb anomaaliatuvastuse mudelite kirjeldamine.

7.1 Kogutud andmed

Ajavahemikul 01.11.2022 – 21.07.2023 on kõikide Tabel 1 kirjeldatud sündmuste liikide kohta kokku kogutud **20.3 mln** kirjet. Nendest **5,1 mln** Ettevõtte 1 halduskeskkonnast (edaspidi „Andmestik 1“), **11,9 mln** avalikelt veebilehtedelt (edaspidi „Andmestik 2“), kus automaatika kasutamist külastajate poolt ei saa välistada ja **3,3 mln** (edaspidi „Andmestik 3“) Ettevõte 3 sisemisest veebirakendusest.

Käesolevas töös kaaluti kõikide eelpool viidatud tabelis olevate sündmuste liikide andmete kasutamist. Pärast eksperimentide läbi viimist osutus, et hiire nupuvajutuste ja lehe kerimiste sündmuste kasutamine raskendaks olulisel määral tehislike andmete loomist ja seaks ohtu nende kvaliteedi. Seetõttu keskendutakse edaspidi sündmustele, mis viitavad kursori liikumisele.

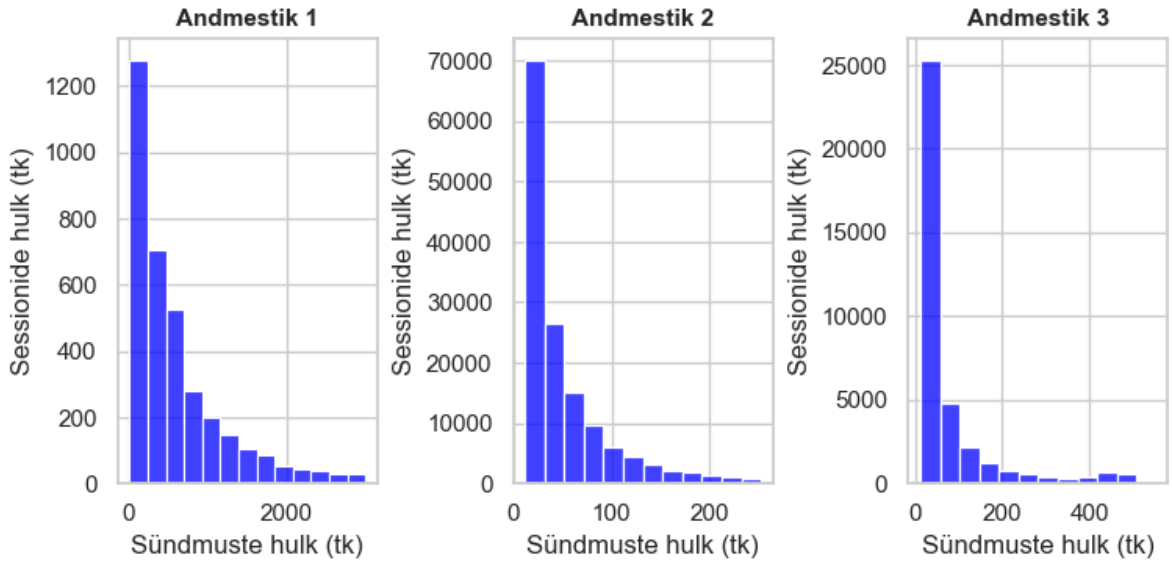
Kursori liikumise sündmusi esineb **3,1 mln** esimesel-, **9,7 mln** teisel- ja **2,9 mln** kolmandal andmestikul. Puutesündmused moodustavad vastavalt 51 350 esimesel, 969 000 teisel ja 0 kolmandal andmestikul.



Joonis 8. x- ja y-koordinaatide varieeruvus algandmetes.

Kursori liikumise sündmuste x- ja y-koordinaatide varieeruvus on esitatud Joonis 8. X-koordinaadi mediaanväärtus on vahemikus 580-819, olles väikseim andmestikul 2 ja suurim andmestikul 3. Y-koordinaadi mediaanväärtus paikneb vahemikus 414-450. Selle alampiir kuulub esimesele andmestikule ning ülempiir teisele andmestikule.

Lisaks mediaanväärtustele osutub karp-vurrud jooniselt, et esineb andmepunkte, ehk erindeid, mis asuvad väljaspool tavapärast jaotust. Antud juhul loetakse erinditeks selliseid väärtusi, mis paiknevad alumisest või ülemisest kvartiilist kaugemal kui 3-kordne kvartiilide vahe. Sellised väärtused moodustavad 0% (79 tk) esimesel andmestikul, 0.02% (1997 tk) teisel andmestikul ja 0.01% (363 tk) kolmandal andmestikul. Eelmainitud väärtuste tekkimine on tingitud suure eraldusvõimega kuvarite- või brauseri suumi kasutamisest. Samuti esineb väärtusi, mis on negatiivsed. Nende tekkimine on võimalik hiirenuppu all hoides ning väljaspoole veebilehitseja akna piire lohistades. Eelnevale tuginedes võib selliseid andmeid lugeda oluliseks, kuna nende tekkimine on võimalik ka hilisemal mudeli rakendamisel.



Joonis 9. Sessioonide hulk sündmuste kohta.

Joonis 9 annab ülevaate sessioonide hulgast sündmuste kohta. Esindatud on sellised sessioonid, mis sisaldavad vähemalt 12 hiire liikumise sündmust. Antud väärtus tuleneb andmete segmenteerimise protsessist.

Lisaks järeldub, et kuigi esimene andmestik on sessioonide hulga poolest kõige väiksem, sisaldab see kõige pikemaid sessioone, ulatudes teatud juhtudel üle 2000 sündmuse ühe

sessiooni kohta. Tuginedes Ettevõtte 1 kirjeldusele, on tegemist oodatud tulemusega, kuna haldusliideses läbi viidavad tegevused võivad nõuda tavapärasest rohkem hiireliigutusi.

Teine andmestik on sündmuste arvu poolest küll suurim, kuid sessioonide pikkus on kõige lühem. Andmestiku kirjeldusele tuginedes on ka antud juhul tegemist oodatud tulemusega, kuna avalikele lehtedele võivad otsingumootorite kaudu jõuda inimesed, kes eksisid lehe valikul või ei leidnud sealt otsitavat informatsiooni ja lahkuvad lehelt.

Kolmas andmestik on sündmuste arvu poolest väikseim, sisaldades sarnaselt teisele andmestikule keskmiselt vähem kui 200 sündmust ühe sessiooni kohta. Kolmanda andmestiku tulemused vastavad samuti oodatud tulemustele, kuna Ettevõtte 3 sisemine veebirakendus on seadistatud töötajate brauseri avaleheks. Sellest tulenevalt võib esineda selliseid sessioone, kus töötaja liigub mõne sekundi jooksul üle veebirakenduse ja seejärel lahkub lehelt. Teisalt sisaldab andmestik ka selliseid sessioone, kus töötaja on veebirakendust aktiivselt kasutanud. Sellest tingituna on antud andmestik sobilik testandmetena kasutamiseks, kuna sisaldab nii pikemaid kui ka lühemaid sessioone.

7.2 Andmetest loodud tunnusvektorid

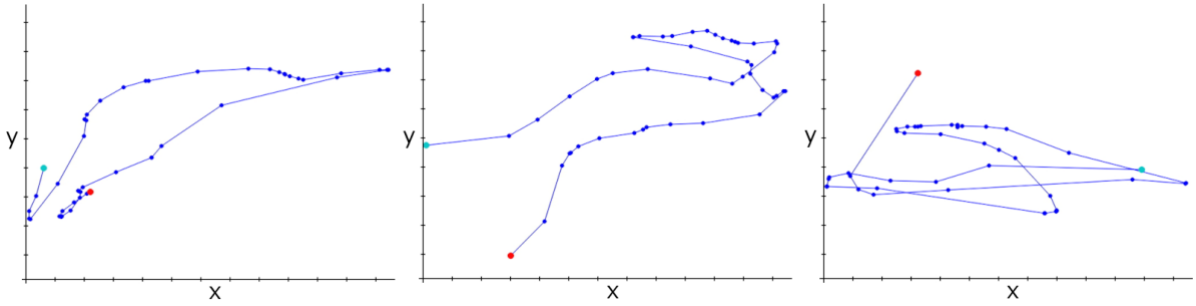
Tabel 3. Tunnusvektorite koguarv sündmuste vahele jääva aja ja tunnusvektori pikkuse kohta.

Andmestik	**Ajakünnis**	**Tunnusvektorite koguarv tunnusvektori pikkuse kohta**						
		16	**24**	**32**	**48**	**64**	**96**	**128**
1	$\Delta t = 250ms$	125 827	69 865	42 541	19 456	10 834	4 777	2 780
2	$\Delta t = 250ms$	166 848	69 591	35 947	13 357	6 655	2 369	1 148
3	$\Delta t = 250ms$	52 728	21 617	10 634	3 459	1 469	384	138
1	$\Delta t = 500ms$	146 902	87 216	57 356	29 215	17 539	8 216	4 765
2	$\Delta t = 500ms$	259 184	124 873	71 557	30 865	16 524	6 365	3 131
3	$\Delta t = 500ms$	86 576	43 058	24 959	10 341	5 283	1 885	827
1	$\Delta t = 1000ms$	160 719	99 561	68 777	38 547	24 823	12 672	7 601
2	$\Delta t = 1000ms$	320 457	167 151	101 037	47 513	27 043	11 364	5 837
3	$\Delta t = 1000ms$	108 731	59 708	37 607	18 179	10 471	4 349	2 225

Tabel 3 annab ülevaate tunnusvektorite koguarvu kohta erinevate tunnusvektori pikkuste ja ajakünniste korral. Andmetest järeldub, et ajakünnise suurendamisel on tunnusvektorite koguarvule positiivne mõju, suurendades nende kogust teatud juhtudel ligi kaks korda.

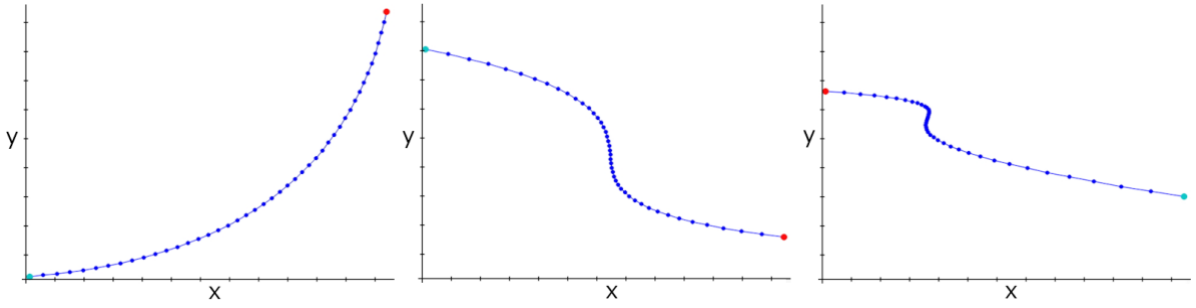
Lisaks osutub, et sõltumata edasistest tulemustest ei ole lühikest ajakünnist ja pikka tunnusvektorit otstarbekas kasutada, kuna kriteeriumile vastavate sündmuste hulk võrreldes esialgsete sündmuste hulgaga on väga väike. See võib tekitada olukorra, kus isegi pikema veebilehe külastuse käigus tehtavad lühiajalised hiire liigutused ei anna segmenteerimisel kokku mitte ühtegi tunnusvektorit.

7.3 Hiire kursori trajektoorid



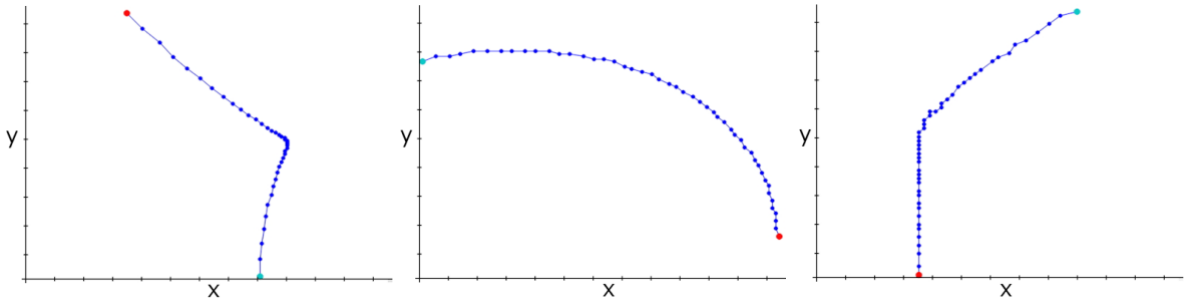
Joonis 10. Inimese poolt tekitatud hiire kursori trajektoorid (tunnusvektori pikkus $n = 48$).

Joonis 10, kus trajektoori alguspunkt on tähistatud sinakas-rohelise värvusega ja lõpp-punkt punase värvusega, kirjeldab hiire kursori trajektoore, mis on tekitatud inimese poolt. Jooniselt järeldub, et inimese poolt tekitatud hiire kursori trajektoorid on oma olemuselt väga dünaamilised, muutes lühikese aja jooksul nii suunda kui kiirust. Kiiruse olemasolule viitab punktide vaheline kaugus.



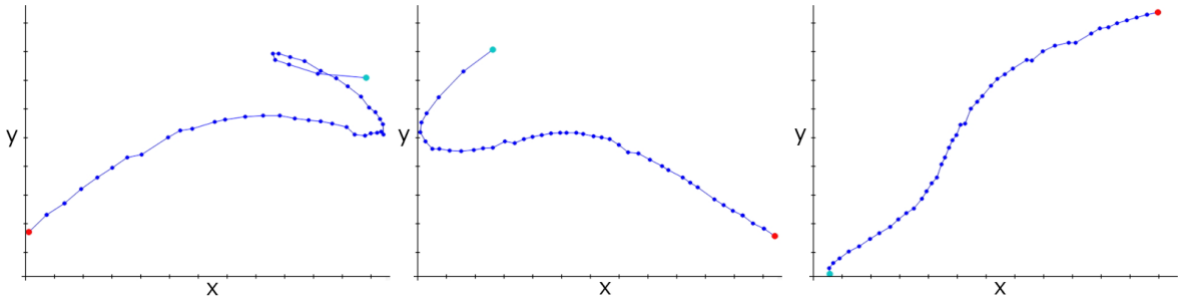
Joonis 11. Ühe, kahe ja kolme kontrollpunktiga *BezierCurve* trajektoorid (tunnusvektori pikkus $n = 48$).

Joonis 11 kirjeldab *BezierCurve* tehislikke trajektoore, mis on loodud ühe, kahe ja kolme kontrollpunktiga. Tulemuseks on sujuvad hiire trajektoorid, mis erinevad oluliselt inimeste poolt tekitatud trajektooridest.



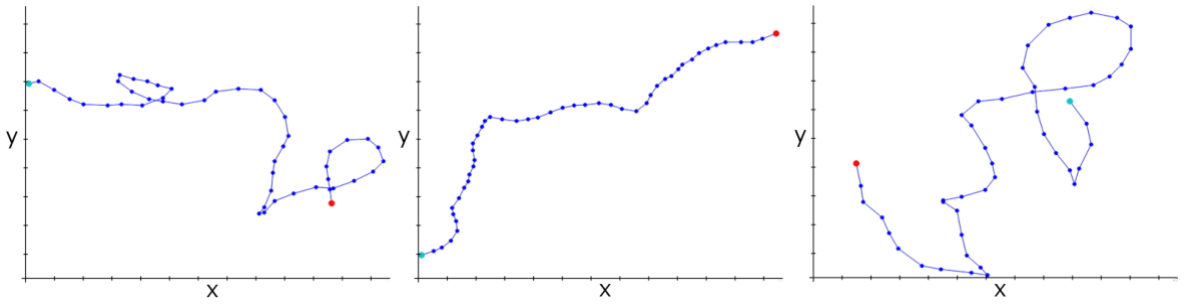
Joonis 12. *GhostCursor* trajektoorid (tunnusvektori pikkus $n = 48$).

GhostCursor tehislikud trajektoorid (Joonis 12) on *BezierCurve* trajektooridega sarnased, kuid lisandunud on sirgjooneline liikumine ja üksikud kõrvalekalded põhitrajektoorist.



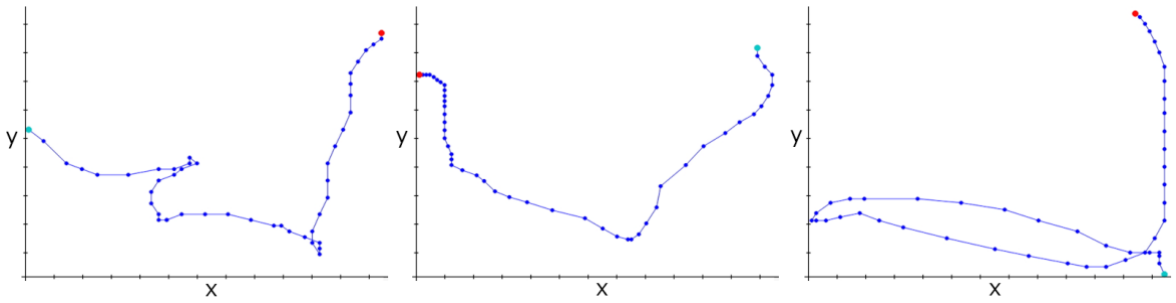
Joonis 13. *HumanCurve* trajektoorid (tunnusvektori pikkus $n = 48$).

HumanCurve tehislikud trajektoorid (Joonis 13) on oluliselt dünaamilisemad kui eelmised tehislike trajektooride generaatorid. Visuaalsel vaatlusel võib joonisel esimesena kujutatud trajektoori pidada inimese poolt tekitatud trajektooriks.



Joonis 14. *WindMouse* trajektoorid (tunnusvektori pikkus $n = 48$).

Joonis 14 kirjeldab *WindMouse* tehislike trajektooride generaatori tulemit. Erinevalt eelmistest trajektooride generaatoritest on antud trajektooride generaator loonud kõige realistlikumad tehislikud trajektoorid, mis sarnanevad inimeste poolt tekitatud trajektooridele.



Joonis 15. *SapiAgent* trajektoorid (tunnusvektori pikkus $n = 48$).

SapiAgent trajektoorid (Joonis 15) on samuti inimeste poolt tekitatud trajektooridele visuaalsel vaatlusel väga sarnased, mistõttu ainult joonisel toodud näidete põhjal on antud trajektoori generaatorit raske tuvastada.

7.4 Täielikult ühendatud pärilevivõrgu mudelid

Lõplike mudelite treenimiseks on kasutatud alapeatükis 6.1.2 kirjeldatud *train_model* funktsiooni koos järgnevas tabelis kirjeldatud parameetritega ja tunnusvektoritega, mis on loodud andmestike „Andmestik 2“ ja „Tehislik 2“ põhjal. Nimetatud tunnusvektoreid ei ole teistes tööetappides kasutatud.

Tabel 4. Toore jõu meetodil leitud FCN mudelite optimaalsed parameetrid.

Tunnusvektori pikkus	**Kihtide arv**	**Filtrite arv kihtide kaupa**	**Konvolutsiooni suurus kihtide kaupa**
16	3	256, 256, 64	4, 8, 6
24	3	256, 256, 256	4, 4, 8
32	3	128, 128, 64	8, 10, 6
48	3	256, 256, 128	10, 10, 6
64	3	64, 256, 64	8, 8, 10
96	3	64, 128, 64	8, 10, 10
128	3	64, 128, 256	10, 10, 6

Treenimiseks kasutatud tunnusvektorite kogus on kirjeldatud Tabel 5. Sündmuste koguarv ühe variatsiooni kohta on leitud tunnusvektori pikkuse ja koguse korrutisena. Ühe variatsiooni koguse baasväärtuseks on võetud väikseim variatsiooni andmestik, milleks antud juhul osutus *SapiAgent* tehislike trajektooride andmestik tunnusvektori pikkusel $n = 128$ kõikidel ajakünniste väärtustel.

Tabel 5. FCN mudeli sisendandmete kogus võrrelduna sündmuste hulgaga.

Tunnusvektori pikkus	**Sisendandmete kogus tunnusvektorites ühe variatsiooni kohta**		
	Ajakünnis $\Delta t = 250ms$	**Ajakünnis $\Delta t = 500ms$**	**Ajakünnis $\Delta t = 1000ms$**
16	3 217	7 535	14 994
24	2 188	5 124	10 196
32	1 657	3 882	7 723
48	1 116	2 614	5 202
64	841	1 971	3 922
96	564	1 320	2 628
128	424	993	1 976
Sündmuste koguarv	≈54 696	≈128 097	≈254 904

Lõplike täielikult ühendatud pärilevivõrgu mudelite kao väärtused on esitatud Tabel 6. Tulemustest järeldub, et kao väärtus on väiksem lühemate tunnusvektori pikkuste puhul, mis võib olla tingitud suuremast sisendandmete, ehk tunnusvektorite kogusest.

Tabel 6. Lõplike FCN mudelite kao väärtused.

Tunnusvektori pikkus	**Mudeli kao väärtus (*model loss*)**		
	Ajakünnis $\Delta t = 250ms$	**Ajakünnis $\Delta t = 500ms$**	**Ajakünnis $\Delta t = 1000ms$**
16	0,2154	0,1613	0,1500
24	0,2376	0,1739	0,1739
32	0,2246	0,1747	0,1424
48	0,2193	0,1552	0,1436
64	0,3257	0,2093	0,1732
96	0,3563	0,2392	0,1627
128	0,3498	0,3277	0,2617

7.5 Anomaaliatuvastuse mudelid

Anomaaliatuvastuse mudelite treenimiseks on kasutatud täielikult ühendatud pärilevivõrgu pudelikaela tunnuseid, mis on loodud andmestiku „Andmestik 1“ nende sessioonide põhjal, mida ei ole kasutatud tehislike trajektooride loomiseks ega FCN mudelite treenimiseks.

Mudelite headuse hindamiseks on testandmetena kasutatud 50% inimeste- ja 50% tehislikke andmeid. Testandmestikena on kasutatud varasemalt kirjeldatud andmestikke „Andmestik 3“ ja „Tehislik 1“, mida ei ole kasutatud FCN mudelite treenimiseks.

Testandmete hulga baasväärtuseks sündmuste kontekstis on võetud väikseim testandmestik, ehk andmestik, mille tunnusvektori pikkus $n = 128$ ja ajakünnis $\Delta t = 250ms$ (138 rida andmeid, ehk 17 802 sündmust).

Järgnevas tabelis on välja toodud kõikide mudelite hüperparameetrite väärtused iga tunnusvektori pikkuse kohta. Erinevatele ajakünniste väärtustele eraldi hüperparameetreid ei seadistatud.

Tabel 7. Anomaaliatuvastuse mudelite hüperparameetrid.

Tunnusvektori pikkus	**Mudel**	**Hüperparameetrid**
16	KPCA	contamination=0.025, gamma=0.01, kernel='rbf'
16	LOF	contamination=0.025, n_neighbors=30, novelty=True
16	OCSVM	contamination=0.025, gamma=0.15, kernel=’rbf’
24	KPCA	contamination=0.025, gamma=0.01, kernel='rbf'
24	LOF	contamination=0.025, n_neighbors=10, novelty=True
24	OCSVM	contamination=0.025, gamma=0.1, kernel=’rbf’
32	KPCA	contamination=0.025, gamma=0.001, kernel='rbf'
32	LOF	contamination=0.025, n_neighbors=40, novelty=True
32	OCSVM	contamination=0.025, gamma=0.1, kernel=’rbf’
48	KPCA	contamination=0.01, gamma=0.0001, kernel='rbf'
48	LOF	contamination=0.025, n_neighbors=20, novelty=True
48	OCSVM	contamination=0.01, gamma='auto', kernel=’rbf’
64	KPCA	contamination=0.01, gamma=0.001, kernel='rbf'
64	LOF	contamination=0.01, n_neighbors=50, novelty=True
64	OCSVM	contamination=0.01, gamma='auto', kernel=’rbf’
96	KPCA	contamination=0.01, gamma=0.001, kernel='rbf'
96	LOF	contamination=0.01, n_neighbors=80, novelty=True
96	OCSVM	contamination=0.01, gamma=0.0001, kernel=’rbf’
128	KPCA	contamination=0.0001, gamma=0.00002, kernel='rbf'
128	LOF	contamination=0.005, n_neighbors=50, novelty=True
128	OCSVM	contamination=0.00015, gamma=0.13, kernel=’rbf’

Järgnevalt on välja toodud kolme parima mudeli tulemused kõikide tunnusvektori pikkuste ja ajakünniste kohta, mis on treenitud alapeatükis 6.2.2 kirjeldatud funktsiooni *train_model* abil.

Tabel 8. Mudelite tulemused testandmetel tunnusvektori pikkusel $n = 16$.

Mudel	**Ajakünnis**	**Kordustäpsus (*precision*)**	**Saagis (*recall*)**	**F1 skoor**	**ROC AUC**
KPCA	$\Delta t = 250ms$	0.74	0.59	0.51	0.89
LOF	$\Delta t = 250ms$	0.73	0.59	0.52	0.83
OCSVM	$\boldsymbol{\Delta t = 250ms}$	**0.76**	**0.65**	**0.60**	**0.89**

KPCA	$\Delta t = 500ms$	0.73	0.56	0.45	0.86
LOF	$\Delta t = 500ms$	0.73	0.59	0.51	0.78
OCSVM	$\boldsymbol{\Delta t = 500ms}$	**0.76**	**0.63**	**0.57**	**0.87**
KPCA	$\Delta t = 1000ms$	Treenimine ebaõnnestus			
LOF	$\Delta t = 1000ms$	0.71	0.57	0.48	0.76
OCSVM	$\boldsymbol{\Delta t = 1000ms}$	**0.74**	**0.57**	**0.47**	**0.85**

Tunnusvektori pikkuse $n = 16$ korral (Tabel 8) osutus parimaks mudeliks OCSVM kõikide ajakünniste väärtuste korral. KPCA mudeli treenimine ajakünnisel $\Delta t = 1000ms$ ei õnnestunud tõenäoliselt andmete hulga tõttu – treenimine katkestati pärast 45 minutit. Teiste mudelite treenimine kestis alla kahe minuti.

Tabel 9. Mudelite tulemused testandmetel tunnusvektori pikkusel $n = 24$.

Mudel	**Ajakünnis**	**Kordustäpsus (*precision*)**	**Saagis (*recall*)**	**F1 skoor**	**ROC AUC**
KPCA	$\boldsymbol{\Delta t = 250ms}$	**0.81**	**0.79**	**0.79**	**0.92**
LOF	$\Delta t = 250ms$	0.76	0.63	0.58	0.84
OCSVM	$\Delta t = 250ms$	0.79	0.74	0.72	0.91
KPCA	$\Delta t = 500ms$	0.76	0.64	0.60	0.88
LOF	$\Delta t = 500ms$	0.77	0.65	0.60	0.82
OCSVM	$\boldsymbol{\Delta t = 500ms}$	**0.76**	**0.67**	**0.65**	**0.88**
KPCA	$\Delta t = 1000ms$	Treenimine ebaõnnestus			
LOF	$\boldsymbol{\Delta t = 1000ms}$	**0.79**	**0.70**	**0.67**	**0.91**
OCSVM	$\Delta t = 1000ms$	0.77	0.65	0.60	0.90

Tunnusvektori pikkuse $n = 24$ korral (Tabel 9) osutusid parimateks mudeliteks: KPCA ($\Delta t = 250ms$), OCSVM ($\Delta t = 500ms$) ja LOF ($\Delta t = 1000ms$). KPCA mudeli treenimine ajakünnisel $\Delta t = 1000ms$ ei õnnestunud tõenäoliselt andmete hulga tõttu, sarnaselt tunnusvektori pikkusele $n = 16$.

Tabel 10. Mudelite tulemused testandmetel tunnusvektori pikkusel $n = 32$.

Mudel	**Ajakünnis**	**Kordustäpsus (*precision*)**	**Saagis (*recall*)**	**F1 skoor**	**ROC AUC**
KPCA	$\boldsymbol{\Delta t = 250ms}$	**0.82**	**0.73**	**0.71**	**0.95**
LOF	$\Delta t = 250ms$	0.80	0.71	0.68	0.93
OCSVM	$\Delta t = 250ms$	0.79	0.68	0.65	0.94
KPCA	$\boldsymbol{\Delta t = 500ms}$	**0.82**	**0.74**	**0.72**	**0.95**
LOF	$\Delta t = 500ms$	0.78	0.67	0.64	0.91
OCSVM	$\Delta t = 500ms$	0.81	0.72	0.70	0.95
KPCA	$\Delta t = 1000ms$	0.78	0.65	0.61	0.93
LOF	$\Delta t = 1000ms$	0.75	0.62	0.56	0.84

OCSVM	$\boldsymbol{\Delta t = 1000ms}$	**0.79**	**0.65**	**0.60**	**0.93**

Tunnusvektori pikkuse $n = 32$ korral (Tabel 10) osutus parimaks mudeliks KPCA ($\Delta t = 250$ms ja $\Delta t = 500$ms) ning OCSVM ($\Delta t = 1000$ms).

Tabel 11. Mudelite tulemused testandmetel tunnusvektori pikkusel $n = 48$.

Mudel	**Ajakünnis**	**Kordustäpsus (*precision*)**	**Saagis (*recall*)**	**F1 skoor**	**ROC AUC**
KPCA	$\boldsymbol{\Delta t = 250ms}$	**0.85**	**0.81**	**0.80**	**0.97**
LOF	$\Delta t = 250ms$	0.82	0.76	0.75	0.95
OCSVM	$\Delta t = 250ms$	0.79	0.65	0.60	0.94
KPCA	$\Delta t = 500ms$	0.84	0.77	0.75	0.93
LOF	$\boldsymbol{\Delta t = 500ms}$	**0.85**	**0.79**	**0.78**	**0.93**
OCSVM	$\Delta t = 500ms$	0.81	0.71	0.69	0.92
KPCA	$\Delta t = 1000ms$	0.83	0.75	0.73	0.95
LOF	$\boldsymbol{\Delta t = 1000ms}$	**0.86**	**0.82**	**0.81**	**0.95**
OCSVM	$\Delta t = 1000ms$	0.80	0.69	0.65	0.94

Tunnusvektori pikkuse $n = 48$ korral (Tabel 11) osutusid parimateks mudeliteks: KPCA ($\Delta t = 250$ms) ja LOF ($\Delta t = 500$ms ning $\Delta t = 1000$ms).

Tabel 12. Mudelite tulemused testandmetel tunnusvektori pikkusel $n = 64$.

Mudel	**Ajakünnis**	**Kordustäpsus (*precision*)**	**Saagis (*recall*)**	**F1 skoor**	**ROC AUC**
KPCA	$\Delta t = 250ms$	0.83	0.77	0.76	0.96
LOF	$\Delta t = 250ms$	0.77	0.61	0.54	0.95
OCSVM	$\boldsymbol{\Delta t = 250ms}$	**0.84**	**0.78**	**0.77**	**0.96**
KPCA	$\boldsymbol{\Delta t = 500ms}$	**0.85**	**0.80**	**0.79**	**0.96**
LOF	$\Delta t = 500ms$	0.83	0.75	0.73	0.92
OCSVM	$\Delta t = 500ms$	0.84	0.78	0.76	0.95
KPCA	$\Delta t = 1000ms$	0.79	0.65	0.61	0.91
LOF	$\boldsymbol{\Delta t = 1000ms}$	**0.82**	**0.74**	**0.72**	**0.92**
OCSVM	$\Delta t = 1000ms$	0.81	0.71	0.68	0.92

Tunnusvektori pikkuse $n = 64$ korral (Tabel 12) osutusid parimateks mudeliteks: OCSVM ($\Delta t = 250$ms), KPCA ($\Delta t = 500$ms) ja LOF ($\Delta t = 1000$ms).

Tabel 13. Mudelite tulemused testandmetel tunnusvektori pikkusel $n = 96$.

Mudel	**Ajakünnis**	**Kordustäpsus (*precision*)**	**Saagis (*recall*)**	**F1 skoor**	**ROC AUC**
KPCA	$\boldsymbol{\Delta t = 250ms}$	**0.90**	**0.89**	**0.89**	**0.98**
LOF	$\Delta t = 250ms$	0.87	0.83	0.83	0.95
OCSVM	$\Delta t = 250ms$	0.86	0.81	0.80	0.91
KPCA	$\Delta t = 500ms$	0.86	0.81	0.80	0.96
LOF	$\Delta t = 500ms$	0.86	0.81	0.81	0.95
OCSVM	$\boldsymbol{\Delta t = 500ms}$	**0.87**	**0.82**	**0.81**	**0.94**
KPCA	$\Delta t = 1000ms$	0.86	0.80	0.80	0.96
LOF	$\boldsymbol{\Delta t = 1000ms}$	**0.87**	**0.84**	**0.83**	**0.96**
OCSVM	$\Delta t = 1000ms$	0.86	0.81	0.81	0.96

Tunnusvektori pikkuse $n = 96$ korral (Tabel 13) osutusid parimateks mudeliteks: KPCA ($\Delta t = 250ms$), OCSVM ($\Delta t = 500ms$) ja LOF ($\Delta t = 1000ms$).

Tabel 14. Mudelite tulemused testandmetel tunnusvektori pikkusel $n = 128$.

Mudel	**Ajakünnis**	**Kordustäpsus (*precision*)**	**Saagis (*recall*)**	**F1 skoor**	**ROC AUC**
KPCA	$\Delta t = 250ms$	0.86	0.85	0.85	0.95
LOF	$\Delta t = 250ms$	0.89	0.88	0.87	0.96
OCSVM	$\boldsymbol{\Delta t = 250ms}$	**0.92**	**0.92**	**0.92**	**0.97**
KPCA	$\boldsymbol{\Delta t = 500ms}$	**0.92**	**0.91**	**0.91**	**0.99**
LOF	$\Delta t = 500ms$	0.90	0.88	0.88	0.99
OCSVM	$\Delta t = 500ms$	0.92	0.90	0.90	0.99
KPCA	$\Delta t = 1000ms$	0.86	0.80	0.79	0.98
LOF	$\boldsymbol{\Delta t = 1000ms}$	**0.88**	**0.84**	**0.84**	**0.99**
OCSVM	$\Delta t = 1000ms$	0.86	0.81	0.80	1

Tunnusvektori pikkuse $n = 128$ korral (Tabel 14) osutusid parimateks mudeliteks: OCSVM ($\Delta t = 250ms$), KPCA ($\Delta t = 500ms$) ja LOF ($\Delta t = 1000ms$).

Antud juhul osutus parimaks mudeliks OCSVM mudel tunnusvektori pikkusel $n = 128$ ja ajakünnisel $\Delta t = 250ms$. Alapeatükis 7.2 välja toodud selgituste põhjal ei saa seda aga parimaks mudeliks lugeda liiga lühikese ajakünnise tõttu. Seetõttu valis töö autor parimaks mudeliks KPCA mudeli samal tunnusvektori pikkusel, kuid ajakünnise väärtusega $\Delta t = 500ms$. Tulemusi võrreldes võib sama heaks mudeliks lugeda ka sama ajakünnise OCSVM mudelit.

Järgnevas tabelis on välja toodud parima mudeli (KPCA) tulemused iga tehisliku trajektoori generaatori kohta eraldi. Andmete osakaalu ja kogust ei ole muudetud. Tulemustest järeldub selgelt, et *WindMouse* trajektooride generaator eristub teistest ja võimaldab luua inimesele sarnaseid hiire liikumise trajektoore, mis suudavad autori poolt loodud mudeleid petta.

Tabel 15. Parima mudeli (KPCA) tulemused erinevate tehislike trajektooride generaatorite lõikes (treeningandmeid tunnusvektorites: 993, testandmeid tunnusvektorites: 827).

Tehisliku trajektoori generaator	**Kordustäpsus (*precision*)**	**Saagis (*recall*)**	**F1 skoor**	**ROC AUC**
Kombineeritud	**0.92**	**0.91**	**0.91**	**0.99**
BezierCurve	1	1	1	1
GhostCursor	1	1	1	1
HumanCurve	0.98	0.98	0.98	1
WindMouse	0.77	0.59	0.51	0.98
SapiAgent	1	1	1	1

Järgnevas tabelis on välja toodud parima anomaaliatuvastuse mudeli tulemused, mille treenimiseks on kasutatud kogu treeningandmestikku.

Tabel 16. Parima mudeli (KPCA) tulemused erinevate tehislike trajektooride generaatorite lõikes (treeningandmeid tunnusvektorites: 2389, testandmeid tunnusvektorites: 827).

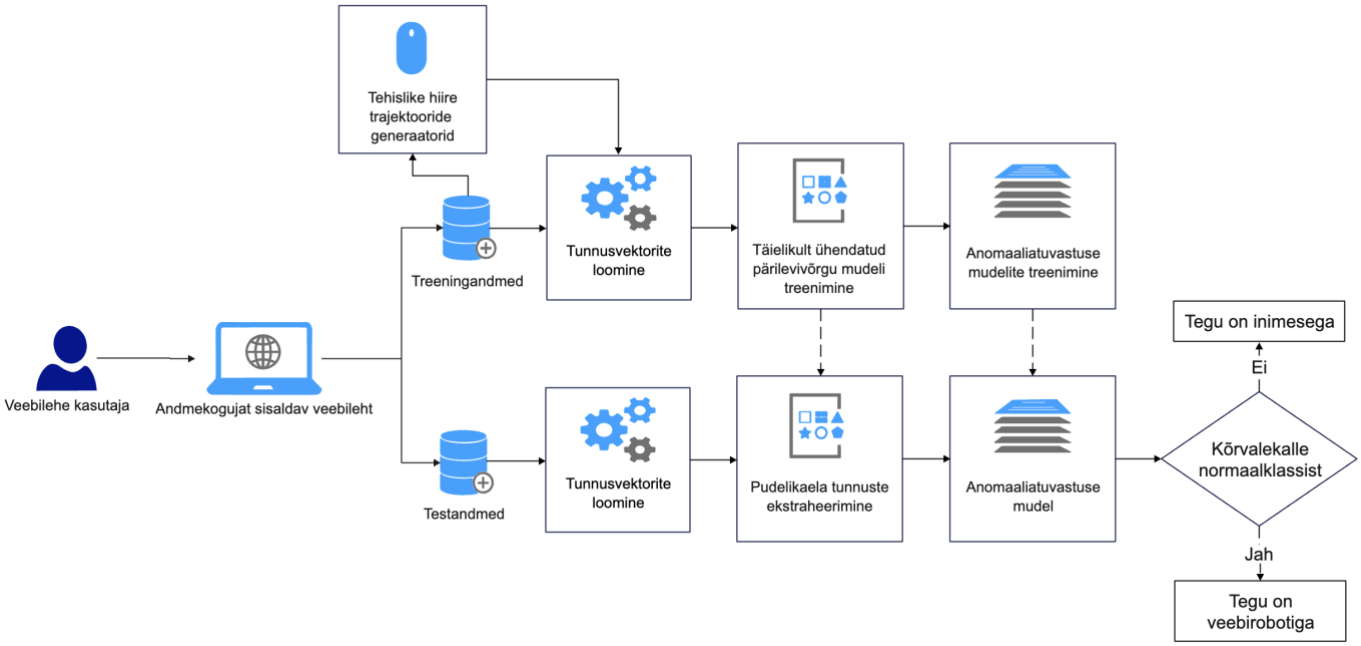
Tehisliku trajektoori generaator	**Kordustäpsus (*precision*)**	**Saagis (*recall*)**	**F1 skoor**	**ROC AUC**
Kombineeritud	**0.90**	**0.88**	**0.88**	**0.99**
BezierCurve	0.98	0.98	0.98	1
GhostCursor	0.99	0.99	0.99	1
HumanCurve	0.91	0.89	0.89	1
WindMouse	0.73	0.51	0.36	0.95
SapiAgent	0.99	0.99	0.99	1

Osutub, et anomaaliatuvastuse treeningandmete suurendamisel parima mudeli tulemus mõnevõrra langeb, mis võib viidata sellele, et hüperparameetreid tuleks uuesti häälestada. Mudeli kombineeritud tulemus sellegipoolest oluliselt ei muutunud.

8 Arutelu ja järeldused

Arutelu ja järelduste peatükk annab esmalt ülevaate valminud lahendusest ja selle piirangutest. Teiseks võrreldakse valminud mudeleid ja tuuakse välja parima mudeli tulemused. Kolmandaks võrreldakse saadud tulemusi teiste töödega. Neljandaks tuuakse välja töö käigus tekkinud väljakutsed ja seejärel tutvustatakse järeltegevusena planeeritud töid. Viimases alapeatükis hinnatakse lühidalt loodud lahenduse kasulikkust ja kasumlikkust.

8.1 Valminud lahendus



Joonis 16. Valminud lahenduse ülevaade.

Joonis 16 annab ülevaate valminud lahendusest, mis koosneb viiest suuremast etapist.

Esimeses etapis toimub hiire andmete kogumine autori poolt loodud andmekoguja rakendusega. Katsete tulemusel võeti vastu otsus, et esialgse lahenduse väljatöötamisel võetakse arvesse ainult hiire liikumise sündmusi ja sündmuste ajatemplit (hiire asukoht ekraanil konkreetsel ajahetkel). Kogutud andmeid kasutatakse treening- ja testandmetena vastavalt keskkonnale, kus andmeid koguti.

Teises etapis toimub tehislike hiire trajektooride genereerimine treeningandmete põhjal. Tehislike andmete kasutamine võimaldab simuleerida veebirobotite käitumist, tänu millele on võimalik luua suures koguses andmeid mudelite efektiivsuse tõstmiseks. Treeningandmete

osaline kasutamine tehislike andmete loomiseks võimaldab tehislike trajektooride genereerimisel arvesse võtta realistlikke kaugusi etteantud ajaühikus.

Kolmandas etapis luuakse kogutud ja genereeritud andmetest tunnusvektorid, mille sisuks esialgse lahenduse puhul on x- ja y-koordinaatide vahe absoluutväärtus. Lisaks võetakse arvesse sündmuste vahele jäävat aega, ehk ajakünnist, mille ületamisel sündmused eraldatakse üksteisest. Tunnusvektorid luuakse seitsmes erinevas pikkuses: 16, 24, 32, 48, 64, 96 ja 128.

Neljandas etapis luuakse treeningandmete ja tehislike andmete põhjal loodud tunnusvektoritest täielikult ühendatud pärilevivõrgu (FCN) mudelid, mida kasutatakse pudelikaela tunnuste ekstraheerimiseks. Pudelikaela tunnused moodustuvad mudeli eelviimase kihi väljundist, mis ei sisalda klassi.

Viiendas etapis luuakse *PyOD* paketi abil anomaaliatuvastuse mudelid, kasutades sisendiks neljandas etapis loodud pudelikaela tunnuseid. Käesoleva töö kontekstis on anomaaliatuvastuse mudelite eesmärgiks tuvastada normaalandmetest (inimese hiire liikumise andmetest) olulised kõrvalekalded, ehk erindid, mis viitavad veebiroboti olemasolule. Katsed viidi läbi mitmete anomaaliatuvastuse mudelitega, millest tulemuste põhjal valiti välja 3 parimat: KPCA, LOF ja OCSVM.

Valminud mudeleid on võimalik kasutada veebirobotite tuvastamiseks hiire dünaamika abil. Parima mudeli kasutamine nõuab olemasoleva lahenduse korral vähemalt 7,3 sekundi jagu hiire andmeid. Andmekogujat täiendavalt optimeerides, ehk eemaldades hiire liikumise sündmuste kogumise piirangu, on võimalik seda hinnanguliselt lühendada ligi poole võrra. Piirangut eemaldades tuleks viia läbi ka täiendavad testimised.

Loodud mudelid võimaldavad tuvastada mõõdukaid roboteid ja teatud liiki täiustatud roboteid. Mudelid ei ole võimelised tuvastama elementaarseid roboteid, kuna nad ei tekita hiire sündmusi. Lisaks ei suuda loodud mudelid tuvastada ka täiustatud roboteid, mis taas esitavad inimeste poolt tekitatud hiire trajektoore, või kasutavad inimese käitumise jäljendamiseks skripte või teeke, millel on võimekus luua väga realistlikke hiire trajektoore. Selliste veebirobotite tuvastamiseks üksnes käitumise dünaamikast ja traditsioonilistest robotilõksudest enamasti ei piisa, mis tähendab, et kaasaegne robotilõks peaks sisendina kasutama mitmeid erinevaid andmeallikaid, mille hulka kuuluvad näiteks veebiserveri liikluse logid ja brauseri API-de kaudu kättesaadavad atribuudid [5, 2, 31]. Seetõttu saab töös loodud

mudeleid kasutada eelkõige täiendava vahendina robotite tuvastamiseks, kuid mitte nende asendamiseks.

8.2 Mudelite võrdlus

Mudelite tulemustele tuginedes võib väita, et tunnusvektori pikkust suurendades paraneb mudelite kvaliteet sama hulga sündmuste juures statistiliselt olulisel määral.

Erinevate ajakünniste pikkuste mudelite tulemusi võrreldes võib järeldada, et ajakünnise pikkuse suurendamine, töös katsetatud väärtuste juures, mudelitele olulist negatiivset mõju üldjuhul ei avalda. Suurim negatiivne mõju avaldus tunnusvektori pikkustel $n = 64$ ja $n = 128$, kuid see võib olla tingitud lisaks ajakünnise pikkusele ka teistest faktoritest. Ajakünnise pikkuse suurendamine võimaldab sisendandmeid efektiivsemalt kasutada, tõstes tunnusvektorite hulka ja seeläbi ka tulemuste usaldusväärsust.

Alapeatükis 6.2 seatud usaldusväärsustaseme, ehk saagise ja F1 skoori väärtuse ≥ 0.8 ületavad kõik mudelid alates tunnusvektori pikkusest $n = 96$. Tulemustele tuginedes võib antud piiri ületava mudeli treenida tunnusvektoritega, mille pikkus paikneb vahemikus $64 < \mathrm{n} \leq 96$.

Parimaks mudeliks valis autor ajakünnisel $\Delta t = 500ms$ ja tunnusvektori pikkusel $n = 128$ oleva KPCA mudeli, mille tulemuseks osutus: **Kordustäpsus = 0,90; Saagis = 0,88; F1 skoor = 0,88 ja ROC AUC = 0,99**. Statistiliselt võib sama heaks mudeliks lugeda ka sama ajakünnise ja tunnusvektori pikkuse väärtusel olevat OCSVM mudelit.

Parima mudeli tulemusi detailsemalt uurides selgus, et üks tehislike trajektooride generaatoritest (*WindMouse*) on võimeline looma inimesele sarnaseid hiire kursori trajektoore, mis suudavad loodud mudelit petta. Autor analüüsis antud trajektoori generaatori tulemusi ka teistel tunnusvektori pikkustel ja tuvastas sealt sarnased tulemused. Seetõttu võib järeldada, et lihtsamate veebirobotite tuvastamiseks on võimalik kasutada ka lühema tunnusvektori pikkusega mudeleid, kuna antud juhul mõjutab ühe trajektoori generaatori tulemus oluliselt mudeli üldist tulemust. Keerukamate trajektoori generaatorite (näiteks *WindMouse*) tuvastamiseks tuleks läbi viia täiendav analüüs, et mõista, kas ja millistel tingimustel on võimalik nende tuvastamise täpsust suurendada.

8.3 Võrdlus teiste töödega

A. Acien, A. Morales, J. Fierrez ja R. Vera-Rodríguez poolt koostatud töös [21] on hiire andmete töötlemisel kasutatud *Sigma-Lognormal* mudelit, mille eesmärk olulised tunnused ekstraheerida inimeste poolt tekitatud hiire liigutuste kiirusprofiilist. Andmete kogumisel osales 58 inimest. Täiendavalt genereeritakse funktsioonipõhised ja vastandgeneratiivse võrgu põhised tehislikud hiire trajektoorid loodavate mudelite kvaliteedi tõstmiseks ja nende testimiseks. Kiirusprofiilist ekstraheeritud tunnused ja tehislikud trajektoorid teisendatakse fikseeritud pikkusega tunnusvektoriteks. Parima mudeli, kus on kasutatud nii inimeste kui ka tehislikke trajektoore, tulemused juhusliku metsa mudelil osutusid järgmiseks: Kordustäpsus = 0,99; Saagis = 0,99; F1 skoor = 0,99 ja ROC AUC = 1. Saadud tulemused on paremad kui käesolevas töös. Peamised põhjused paremate tulemuste saavutamisel on tõenäoliselt tingitud *Sigma-Lognormal* mudeli kasutamisest ja tehniliselt keerukamatest tehislike trajektooride generaatoritest, mis tõstsid treenimisel mudelite kvaliteeti.

Y. Zhao, A. Wei ja Z. Cai töös [19] kujutati hiire trajektoorid piltidel ja kasutati seejärel *ResNet* CNN mudelit klassifitseerimiseks. Andmete kogumises osales 50 inimest. Veebirobotite andmeteks genereeriti nelja tüüpi tehislikud trajektoorid: sirgjooned, kumerad jooned, terava servaga ühendatud sirgjooned ja sirgjooned väikese kumerusega. Tulemuste kontrollimiseks kasutatakse andmestiku poolitamist. Parima tulemusena tuvastati 96.2% veebirobotitest. Magistritöö autori hinnangul on käesolevas töös esitatud parim mudel vähemalt samaväärne käesoleva tööga, kuna käesolev töö võtab arvesse ka keerukamaid tehislike trajektooride generaatoreid.

H. Niu, J. Chen, Z. Zhang kasutasid oma töös [66] 120 inimese hiire andmeid, mis segmenteeriti fikseeritud pikkusega tunnusvektoriteks pikkusega n = 60. Tehislikeks trajektoorideks genereeriti: sirgjooned, sirgjooned kumerusega, kumerad jooned ja kumerad jooned mitme kontrollpunktiga. Tulemuste kontrollimiseks kasutatakse ristvalideerimist. Parima mudeli LSTM+1D-CNN tõeliseks positiivseks määraks osutus 99.78% (robotite tuvastamise osas) ja tõeliseks negatiivseks määraks 99.71% (inimeste tuvastamise osas). Tulemustele tuginedes võib saadud tulemusi pidada paremaks kui käesolevas töös esitatud mudelitel, kuid tehislike trajektooride lihtsuse tõttu on seda keeruline tõestada.

Autor soovib võrdluse alapeatükki kokku võttes rõhutada, et käesolevas töös sisalduvad treening- ja testandmed on pärit täiesti erinevatest veebikeskkondadest ja sisaldavad

veebilehtede statistika põhjal kokku tuhandete erinevate inimeste andmeid. Seetõttu esineb risk, et seotud töödes kasutatud väikesed andmekogud (alla 150 unikaalse inimese) ei ole antud magistritööga otseselt võrreldavad. Üks meetod selle tõestamiseks oleks magistritöös kasutatud andmekogu põhjal läbi viia korduskatsed seotud töödes kirjeldatud mudelite põhjal, mis on aga väga ajamahukad ja ei kuulu antud töö skoopi.

8.4 Töö käigus tekkinud väljakutsed

Magistritöö esimeses etapis viis autor läbi katsed tunnuste ekstraheerimiseks (ja otse klassifitseerimiseks) CNN-i abil, mille sisendiks olid hiire trajektooride pildid. Katsed viidi läbi piltidega, kus oli kujutatud: 1) ainult koordinaatide punkte; 2) koordinaatide punkte koos nendevahelise sirgjoonega; 3) 1. ja 2. koos nupuvajutuste märgetega; 4) erinevaid värve kasutades. Katsete käigus selgus, et saadud mudelite treenimine võrreldes töös esitatud mudelitega oli ligikaudu 8 korda aeglasem ja saadud mudelid ei andnud häid tulemusi.

Teine väljakutse seisnes anomaaliatuvastuse mudelite hüperparameetrite häälestamises. Mudelite rohkuse tõttu ei mahtunud antud töö ajaraami parimate hüperparameetrite väärtuste leidmine.

Kolmas väljakutse on seotud mudelite testimisega. Mudelite testimiseks oli täiendavalt planeeritud kasutada sõltumatuid testandmeid tehislike trajektooride näol *BeCAPTCHA-Mouse* andmestikust [21]. Tegemist on andmetega, mille kasutamiseks tuleb esitada taotlus. Magistritöö autor proovis andmete kasutamiseks korduvalt andmete autoritega ühendust saada, kuid vastust töö esitamise ajaks ei saabunud.

8.5 Edasised tööd

Järeltegevusena on autoril planeeritud parima tulemuse saavutanud mudelite hüperparameetrid häälestada ja mudelid integreerida olemasolevatesse rakendustesse, kasutades rööpjuurutuse strateegiat nende testimiseks ja kvaliteedi tagamiseks [67]. Integreerimise eesmärk on täiendavalt testida töös esitatud tulemusi. Lisaks on planeeritud laialdaselt kasutuses olevate automatiseerimise tööriistadega simuleerida negatiivset mõju avaldavat robotit (täiustatud robotit), et täiendavalt testida robotite tuvastamise efektiivsust. Lisaks eelnevale tuleb mudeli rakendamisel arvestada ka inimestega, kellel esinevad erivajadused, mis mõjutavad arvuti hiire kasutamist.

Täiendavalt on autoril planeeritud *SapiAgent* tehisliku trajektoori generaatori mudeli treenimiseks kasutada andmekoguja rakendusega kogutud andmeid. Eesmärgiks on valideerida, kas sisendandmete muutmisel on võimalik *SapiAgent* tehisliku trajektoori generaatori mudeli tõhusust tõsta, ehk luua trajektoorid mis suudavad magistritöö autori poolt loodud mudeleid petta, sarnaselt *WindMouse* trajektoori generaatorile.

Üheks järeltegevuseks võiks pidada ka kiiruse arvesse võtmist tunnusvektorite loomisel, mis võib töös kasutatud kirjandusele tuginedes mudelite kvaliteedile avaldada olulist (positiivset) mõju.

8.6 Loodud lahenduse kasulikkus ja kasumlikkus

Töös esitatud mudeleid on mõistlik kasutada eelkõige olemasolevate veebirobotite tuvastamise teenuste kõrval, täiendava lahendusena. See võimaldaks veebiroboteid tuvastades arvesse võtta mitme erineva tehnilise võimekusega teenuse või lahenduse tulemusi, mille põhjal saaks rakendus tulemusi kombineerides langetada otsuse, kas lõppkasutaja (või veebiroboti) poolt teostatud tegevus blokeerida, või mitte. Eraldiseisvana ei soovita autor töös loodud mudeleid kasutada, kuna võib tekkida olukordi, kus hiire andmeid ei teki tuvastuse jaoks piisavas koguses, või üldse. Samuti ei võeta hetkel arvesse erivajadustega inimeste arvutikasutuse eripära.

Kasumlikkuse poolelt võib magistritöös esitatud mudelite töös hoidmine nõuda suures koguses ajalist ressurssi võimalike probleemide lahendamiseks ja mudelite efektiivsuse tõstmiseks. Seetõttu on töös esitatud mudeleid mõistlik rakendada ainult suuremates ettevõtetes, kus on vastavat kompetentsi omav arendusmeeskond ja suure külastatavusega veebilehed, kus on tarvis veebirobotite tegevust piirata. Sellisel juhul võib töös esitatud mudelite kasutamine vähendada väliste veebirobotite tuvastamise teenuste kasutamise vajadust. Vastasel juhul on ärilises mõttes kasulikum robotituvastuse funktsionaalsust tarbida valmis teenusena.

9 Kokkuvõte

Veebirobotite tehniline areng on tekitanud olukorra, kus robotite tuvastamiseks kasutatavad tehnilised vahendid, ehk robotilõksud, on muutunud keerulisteks inimestele, kuid lihtsaks robotitele.

Käesoleva magistritöö eesmärgiks oli analüüsida hiire dünaamika kasutamise võimalusi veebirobotite tuvastamise kontekstis, et hinnata kas selle kasutamine võiks robotilõksud inimestele taas lihtsamaks muuta. Samuti tõi autor välja hiire dünaamika kasutamisega seotud piirangud.

Töös uuriti hiire andmete kogumise- ja tehislike andmete loomise meetodeid ning nende töötlemise võimalusi veebirobotite tuvastamise kontekstis. Samuti kirjeldati veebiroboteid nende tehnilise võimekuse alusel ja analüüsiti nende tuvastamiseks kasutatavaid meetodeid ning nende puudusi.

Töös teostatud eksperimentide käigus segmenteeriti kogutud hiire andmed ja tehislikult loodud hiire andmed erineva fikseeritud pikkusega tunnusvektoriteks, mida kasutati täielikult ühendatud pärilevivõrgu, ehk FCN mudelite loomiseks. FCN mudeleid kasutati tunnusvektoritest pudelikaela tunnuste ekstraheerimiseks, mille põhjal loodi KPCA, LOF ja OCSVM mudelid, mis täitsid anomaaliatuvastuse eesmärki. Anomaaliaks peetakse antud juhul kõrvalekallet tavapärasest inimese käitumisest hiire kasutamisel. Parimaks mudeliks valis autor KPCA mudeli, mille sisendiks on antud juhul vaja vähemalt 7,3 sekundi jagu hiire kursori liikumise andmeid. Parima mudeli tulemuseks osutus: Kordustäpsus = 0,90; Saagis = 0,88; F1 skoor = 0,88 ja ROC AUC = 0,99.

Saadud tulemustele ja hiire dünaamika eripäradele tuginedes soovitas autor loodud mudeleid kasutada eelkõige olemasolevate robotilõksude kõrval täiendava lahendusena. Lisaks analüüsiti loodud mudelite kasumlikkust, milles jõuti järeldusele, et loodud mudeleid on äriliselt mõistlik arendada ja kasutada ainult suuremates ettevõtetes.

Täiendavalt kirjeldas autor järeltegevusena planeeritud töid mudelite efektiivsuse tõstmiseks.

Kasutatud kirjandus

[1] E. Ferrara, O. Varol, C. A. Davis, F. Menczer ja A. Flammini, „The rise of social bots," *Communications of the ACM,* kd. 59, nr 7, pp. 96-104, 2016.

[2] Imperva, „2023 Bad Bot Report," 2023. [Võrgumaterjal]. Available: https://www.imperva.com/resources/reports/2023-Imperva-Bad-Bot-Report.pdf.

[3] X. Li, B. A. Azad, A. Rahmati ja N. Nikiforakis, „Good Bot, Bad Bot: Characterizing Automated Browsing Activity," %1 *2021 IEEE Symposium on Security and Privacy (SP)*, San Francisco, 2021.

[4] M. Moradi ja M. R. Keyvanpour, „CAPTCHA and its Alternatives: A Review," *Security and Communication Networks,* kd. 8, nr 12, pp. 2135-2156, 2014.

[5] DataDome, „ReCAPTCHA v2 vs. v3: Efficient bot protection? [2024 Update]," 30 November 2023. [Võrgumaterjal]. Available: https://datadome.co/guides/captcha/recaptchav2-recaptchav3-efficient-bot-protection. [Kasutatud 18 Detsember 2023].

[6] Z. Chu, S. Gianvecchio ja H. Wang, „Bot or Human? A Behavior-Based Online Bot Detection System," %1 *From Database to Cyber Security*, kd. 11170, Springer, Cham, 2018, pp. 432-449.

[7] S. Khan ja D. Hou, „Mouse Dynamics Behavioral Biometrics: A Survey," *ACM Computing Surveys,* kd. 37, nr 4, 2022.

[8] S. Sadeghpour ja N. Vlajic, „Analysis of Novel Mouse Dynamics Dataset with Repeat Sessions: Helpful Observations for Tackling Session-Replay Bot," %1 *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, Las Vegas, 2023.

[9] P. Chong, Y. Elovici ja A. Binder, „User Authentication Based on Mouse Dynamics Using Deep Neural Networks: A Comprehensive Study," *IEEE Transactions on Information Forensics and Security,* kd. 15, pp. 1086-1101, 2019.

[10] S. Sadeghpour ja N. Vlajic, „ReMouse Dataset: On the Efficacy of Measuring the Similarity of Human-Generated Trajectories for the Detection of Session-Replay Bots," *Cybersecurity and Privacy,* kd. 3, nr 1, 2023.

[11] C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis ja I. Kompatsiaris, „Detection of Advanced Web Bots by Combining Web Logs with Mouse Behavioural Biometrics," *Digital Threats: Research and Practice,* kd. 2, nr 3, pp. 1-26, 2021.

[12] M. Karim ja M. Hasanuzzaman, „A Study on Mouse Movement Features to Identify User," *Scientific Research Journal,* kd. 8, nr 4, pp. 77-82, 2020.

[13] H. Gamboa ja A. Fred, „A Behavioural Biometric System Based on Human Computer Interaction," %1 *Proceedings of SPIE*, Bellingham, 2004.

[14] A. Acien, A. Morales, J. Fierrez, R. Vera-Rodriguez ja O. Delgado-Mohatar, „BeCAPTCHA: Behavioral bot detection using touchscreen and mobile sensors

benchmarked on HuMIdb,“ *Engineering Applications of Artificial Intelligence,* kd. 98, nr 104058, 2021.

[15] M. Antal ja E. Egyed-Zsigmond, „Intrusion Detection Using Mouse Dynamics,“ *IET Biometrics,* kd. 8, nr 5, pp. 285-294, 2019.

[16] F. J. Nóvoa, D. Garabato, C. Dafonte, R. Santoveña ja A. Silvelo, „AI-based user authentication reinforcement by continuous extraction of behavioral interaction features,“ *Neural Computing and Applications,* kd. 34, pp. 11691-11705, 2022.

[17] M. Antal ja N. Fejér, „Mouse dynamics based user recognition using deep learning,“ *Acta Universitatis Sapientiae Informatica,* kd. 12, pp. 39-50, 2020.

[18] A. M., B. K. ja F. N., „sapiagent,“ 4 August 2022. [Võrgumaterjal]. Available: https://github.com/margitantal68/sapiagent. [Kasutatud 21 Veebruar 2023].

[19] Y. Zhao, A. Wei ja Z. Cai, „A Deep Learning Approach to Web Bot Detection Using Mouse Behavioral Biometrics,“ %1 *Biometric Recognition*, Cham, 2019.

[20] M. Antal, N. Fejér ja K. Buza, „SapiMouse: Mouse Dynamics-based User Authentication Using Deep Feature Learning,“ %1 *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, Timisoara, 2021.

[21] A. Acien, A. Morales, J. Fierrez ja R. Vera-Rodríguez, „BeCAPTCHA-Mouse: Synthetic mouse trajectories and improved bot detection,“ *Pattern Recognition,* kd. 127, nr C, Juuli 2022.

[22] M. Antal, K. Buza ja N. Fejer, „SapiAgent: A Bot Based on Deep Learning to Generate Human-Like Mouse Trajectories,“ *IEEE Access,* kd. 9, pp. 124396-124408, 2021.

[23] J. Jin, J. Offutt, N. Zheng, F. Mao, A. Koehl ja H. Wang, „Evasive bots masquerading as human beings on the web,“ %1 *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Budapest, 2013.

[24] T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis, C. Iliou ja I. Kompatsiaris, „Web Bot Detection Evasion Using Deep Reinforcement Learning,“ %1 *ARES '22: Proceedings of the 17th International Conference on Availability, Reliability and Security*, Vienna, 2022.

[25] D. Goßen, H. L. Jonker, S. Karsch, B. Krumnow ja D. Roefs, „HLISA: towards a more reliable measurement tool,“ %1 *IMC '21: Proceedings of the 21st ACM Internet Measurement Conference,* New York, 2021.

[26] S. Baydas ja B. Karakas, „Defining a curve as a Bezier curve,“ *Journal of Taibah University for Science,* kd. 13, nr 1, pp. 522-528, 2019.

[27] D. S. Sisodia, S. Verma ja O. P. Vyas, „Agglomerative Approach for Identification and Elimination of Web Robots from Web Server Logs to Extract Knowledge about Actual Visitors,“ *Journal of Data Analysis and Information Processing,* nr 3, pp. 1-10, 2015.

[28] D. Flanagan, JavaScript: the definitive guide, O'Reilly Media, Inc., 2006.

[29] S. S. Brown, N. DiBari ja S. Bhatia, „I Am 'Totally' Human: Bypassing the reCaptcha,“ %1 *2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, Jaipur, 2017.

[30] C. Iliou, T. Tsikrika, S. Vrochidis ja I. Kompatsiaris, „Evasive Focused Crawling by Exploiting Human Browsing Behaviour: a Study on Terrorism-Related Content,“ %1 *1st International Workshop on Cyber Deviance Detection co-*

located with the Tenth International Conference on Web Search and Data Mining (CyberDD @ WSDM 2017), Cambridge, 2017.

[31] B. A. Azad, O. Starov, P. Laperdrix ja N. Nikiforakis, „Web Runner 2049: Evaluating Third-Party Anti-bot Services," %1 *Detection of Intrusions and Malware, and Vulnerability Assessment: 17th International Conference*, Lisbon, 2020.

[32] A. Hall, L. Terveen ja A. L. Halfaker, „Bot Detection in Wikidata Using Behavioral and Other Informal Cues," *Proceedings of the ACM on Human-Computer Interaction*, kd. 2, nr 64, pp. 1-18, 2018.

[33] G. Suchacka, A. Cabri, S. Rovetta ja F. Masulli, „Efficient on-the-fly Web bot detection," *Knowledge-Based Systems*, kd. 223, nr 107074, 2021.

[34] A. Felkner, M. Janiszewski ja P. Lewandowski, „SpiderTrap—An Innovative Approach to Analyze Activity of Internet Bots on a Website," *IEEE Access*, kd. 8, pp. 141292-141309, 2020.

[35] F. Masulli, G. Suchacka ja S. Rovetta, „Bot recognition in a Web store: An approach based on unsupervised learning," *Journal of Network and Computer Applications*, kd. 157, nr 102577, 2020.

[36] D. Doran ja S. S. Gokhale, „An integrated method for real time and offline web robot detection," *Expert Systems*, kd. 33, nr 6, pp. 592-606, 2016.

[37] P. Laperdrix, N. Bielova, B. Baudry ja G. Avoine, „Browser Fingerprinting: A Survey," *ACM Transactions on the Web*, kd. 14, nr 2, pp. 1-33, 2020.

[38] M. Schwarz, F. Lackner ja D. Gruss, „JavaScript Template Attacks: Automatically Inferring Host Information for Targeted Exploits," %1 *Network and Distributed System Security Symposium 2018*, San Diego, 2018.

[39] H. Jonker, B. Krumnow ja G. Vlot, „Fingerprint Surface-Based Detection of Web Bot Detectors," %1 *European Symposium on Research in Computer Security*, Luxembourg, 2019.

[40] Shivani ja R. K. Challa, „CAPTCHA: A Systematic Review," %1 *2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI)*, Buldhana, 2020.

[41] S. Mali, G. Srinivasan, A. Waghela, G. Thatte ja R. Mistry, „DeCaptcha: Cracking captcha using Deep Learning Techniques," %1 *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, Mathura, 2021.

[42] Z. Cai, X. Guan, Y. Du, R. A. Maxion ja C. Shen, „User Authentication Through Mouse Dynamics," *IEEE Transactions on Information Forensics and Security*, kd. 8, nr 1, pp. 16-30, 2013.

[43] C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis ja I. Kompatsiaris, „Web Bot Detection Evasion Using Generative Adversarial Networks," %1 *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, Rhodes, 2021.

[44] T. Fawcett, „An introduction to ROC analysis," *Pattern Recognition Letters*, kd. 27, nr 8, pp. 861-874, 2006.

[45] „Pildi kategoriseerimine masinõppega," [Võrgumaterjal]. Available: https://pydoc.pages.taltech.ee/extra/image_recognition/process_overview.html. [Kasutatud 12 12 2023].

[46] S. Hessner, „Why are precision, recall and F1 score equal when using micro averaging in a multi-class problem?," [Võrgumaterjal]. Available:

https://simonhessner.de/why-are-precision-recall-and-f1-score-equal-when-using-micro-averaging-in-a-multi-class-problem/. [Kasutatud 12 12 2023].

[47] G. Bierman, M. Torgersen ja M. Abadi, „Understanding typescript," %1 *European Conference on Object-Oriented Programming*, Uppsala, 2014.

[48] J. Meyerson, „The go programming language," *IEEE software,* kd. 31, nr 5, p. 104, 2014.

[49] G. Van Rossum ja F. L. Drake Jr, Python reference manual, Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[50] T. Kluyver, B. Ragan-Kelley, F. Perez, B. Granger, M. Bussonnier, J. Frederic ja C. Willing, „Jupyter Notebooks – a publishing format for reproducible computational workflows," %1 *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 2016, pp. 87-90.

[51] „MongoDB," 2023. [Võrgumaterjal]. Available: https://www.mongodb.com/.

[52] The pandas development team, „Data structures for statistical computing in python," %1 *Proceedings of the 9th Python in Science Conference*, McKinney, 2010.

[53] C. R. Harris, K. J. Millman, J. S. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk ja Br, „Array programming with NumPy," *Nature,* kd. 585, pp. 357-362, 2020.

[54] J. D. Hunter, „Matplotlib: A 2D graphics environment," *Computing in Science & Engineering,* kd. 9, nr 3, pp. 90-95, 2007.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau ja M. Brucher, „Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* kd. 12, pp. 2825-2830, 2011.

[56] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard ja R. Jozefowicz, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,* 2015.

[57] Y. Zhao, Z. Nasrullah ja Z. Li, „PyOD: A Python Toolbox for Scalable Outlier Detection," *Journal of Machine Learning Research,* kd. 20, nr 96, pp. 1-7, 2019.

[58] patrikoss, „pyclick," 21 September 2018. [Võrgumaterjal]. Available: https://github.com/patrikoss/pyclick. [Kasutatud 15 Detsember 2022].

[59] C. Matt, „Python Ghost Cursor," 11 August 2021. [Võrgumaterjal]. Available: https://github.com/mcolella14/python_ghost_cursor. [Kasutatud 2 Juuni 2023].

[60] R. A. Burno, B. Wu, R. Doherty, H. Colett ja R. Elnaggar, „Applying Fitts' Law to Gesture Based Computer Interactions," *Procedia Manufacturing,* kd. 3, pp. 4342-4349, 2015.

[61] B. J. Land, „WindMouse, an algorithm for generating human-like mouse motion," 25 Aprill 2021. [Võrgumaterjal]. Available: https://ben.land/post/2021/04/25/windmouse-human-mouse-movement/. [Kasutatud 23 Jaanuar 2023].

[62] A. M., B. K. ja F. N., „SapiMouse," 30 Jaanuar 2021. [Võrgumaterjal]. Available: https://github.com/margitantal68/sapimouse. [Kasutatud 6 Juuni 2023].

[63] H. Hoffman, „Kernel PCA for novelty detection,“ *Pattern Recognition,* kd. 40, pp. 863-874, 2007.

[64] M. M. Breunig, H.-P. Kriegel, R. Tak Yan Ng ja S. Jörg, „LOF: identifying density-based local outliers,“ *ACM SIGMOD Record,* kd. 29, nr 2, pp. 93-104, 2000.

[65] M. Nayak ja P. P. Das, „Outlier Detection Methods - An Analysis,“ *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT),* kd. 2, nr 9, pp. 1017-1027, 2013.

[66] H. Niu, J. Chen, Z. Zhang ja Z. Cai, „Mouse Dynamics Based Bot Detection Using Sequence Learning,“ %1 *Biometric Recognition,* 2021.

[67] I. Weber, S. Satyal, H.-y. Paik, C. D. Ciccio ja J. Mendling, „Shadow Testing for Business Process Improvement,“ %1 *On the Move to Meaningful Internet Systems. OTM 2018 Conferences,* Malta, 2018.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Elmo Egers

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebirobotite tuvastamine hiire dünaamika abil“, mille juhendaja on Avar Pentel
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

31.12.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.