

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK**



SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM

Egérdinamika-alapú bot detektálás

TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

Témavezető:
Antal Margit,
Egyetemi docens

Hallgató:
Kovács Andrea

2022

Kivonat

A botok detektálását egyre nagyobb figyelem övezi napjainkban, hiszen az automatizált feladatok elvégzésén túl számos kárt okoznak. Habár a támadások kivédése érdekében egyre több aspektusból vizsgálják a botok jelenlétét, ez egy komplex folyamatnak számít, hiszen a legmodernebb botok már az emberi viselkedés utánzását is megcélazzák.

Dolgozatomban a botok emberszerű viselkedését vizsgálom különböző módszerekkel generált egérmozgások szempontjából. Bézier görbék különbözően paraméterezett változataival, illetve egy idősorokra szakosodott generatív ellenséges hálózat (TimeGAN) segítségével szintetikus adathalmazokat generáltam. A nyers adathalmazokon jellemzőkinyerést hajtottam végre, mely összesen 29 jellemzőt eredményezett.

Rendelkezésemre állt a Sapimouse adathalmaz, amely 120 felhasználó egérdinamikai adatait tartalmazza. Ezt az adathalmazt, illetve a szintetikus adathalmazokat felhasználva mértem különböző detektorok teljesítményét. A legjobban teljesítő detektor, mely a legsikeresebbnek bizonyult a generált és valós emberi egérmozgások elkülönítésében az LOF lett, 1 egérmozgás alapján 0.81 fölötti, 10 egérmozgás esetében pedig 0.96 fölötti AUC (Area Under the ROC Curve) értékekkel.

A kutatási részt egy szoftver elkészítése követte, mely szemlélteti az egyetlen egérmozgáson alapuló bot detektálást.

Kulcsszavak: bot detektálás, egérmozgás, anomália detektor.

Abstract

Nowadays bot detection is gaining more and more attention due to the damage caused by bots besides the fact that they perform automated tasks. Although the presence of bots is being examined from more and more aspects to prevent attacks, it is a complex process as state-of-the-art bots are already aimed at mimicking human behavior.

In my research, I examine the human-like behavior of bots in terms of different generated mouse movements. I generated synthetic data sets based on different parameterized versions of Bézier curves and also by a generative adversarial network (TimeGAN) specialized in time series. I performed feature extraction on raw data sets, which resulted in a total of 29 features.

I had access to the Sapimouse data set, which contains mouse dynamics data for 120 users. Using this data set and the synthetic data sets, I measured the performance of different detectors. The best performing detector, which proved to be the most successful in distinguishing between generated and real human mouse movements was the LOF, achieving values above 0.81 for 1 mouse movement and above 0.96 for 10 mouse movements in terms of AUC (Area Under the ROC Curve) metric.

The research part was followed by the development of a software that illustrates single mouse movement-based bot detection.

Keywords: bot detection, mouse movement, anomaly detector.

Tartalomjegyzék

1. Bevezető	11
2. Szakirodalmi áttekintő	13
2.1. Bot detektálás	13
2.2. CAPTCHA	14
3. Adatok és módszerek	16
3.1. SapiMouse adathalmaz	16
3.1.1. Adathalmaz szegmentálása	16
3.2. Görbe generálási módszerek	17
3.2.1. Bézier görbék	17
3.2.2. Generált adathalmazok	18
3.3. Idősorok generálása TimeGAN neuronhálózval	19
3.3.1. TimeGAN	19
3.3.2. A TimeGAN fontosabb paraméterei	21
3.4. Bot detektálás anomália detektorokkal	22
3.4.1. Anomália detektálás	22
3.4.2. Osztályozás	23
3.4.3. Fontosabb detektorok	23
3.5. Adatelemzés	26
3.6. Jellemzőkinyerés	26
3.6.1. Szintetikus adatok minősége	28
4. Mérési eredmények	30
4.1. Mérési protokoll	30
4.2. Metrikák	30
4.3. Eredmények	31
4.3.1. Egyetlen trajektória alapú eredmények	32
4.3.2. 10 trajektória alapú eredmények	35
4.3.3. A <i>randomTimeGan</i> adathalmazzal végzett mérések eredményei	37
4.4. Vizualizáció	38
5. Bot detektáló szoftver	40
5.1. A szoftver áttekintése	40
5.1.1. Felhasználói követelmények	40
5.1.2. Rendszerkövetelmények	41
5.2. Tervezés	41

5.3. Kivitelezés	42
6. Összefoglaló	46
Ábrák jegyzéke	48
Irodalomjegyzék	50
Függelék	51

Jelölések

Rövidítés	Angol megnevezés	Magyar megnevezés
ROC curve	Receiver operating characteristic curve	ROC görbe
AUC	Area Under the ROC Curve	ROC görbe alatti terület
TPR	True Positive Rate	Pozitívak elfogadási aránya
FPR	False Positive Rate	Negatívak elfogadási aránya
COPOD	Copula-Based Outlier Detection	Copula-alapú kiugró érték detektálás
KNN	k Nearest Neighbors	k-legközelebbi szomszéd
PCA	Principal Component Analysis	Főkomponens analízis
LOF	Local Outlier Factor	Helyi kiugró érték tényező
OCSVM	One-Class Support Vector Machines	Egyosztályú tartóvektorgép
IForest	Isolation Forest	Elszigetelő erdő
t-SNE	t-distributed stochastic neighbourhood embedding	t-elosztott sztochasztikus szomszéd beágyazási módszer
GAN	Generative adversarial network	Generatív ellenséges neuronháló

1. fejezet

Bevezető

Napjainkban egyre nagyobb figyelmet kapnak a különböző webes botok, melyek számos változata ismert az egyszerű automatizált szkriptektől a fejlett webbotokig. Utóbbiak rendelkeznek a böngésző ujjlenyomatával, támogatják a böngésző főbb funkcióit és képesek akár emberszerű viselkedés tanúsítására is. Ezen tulajdonságaiknak köszönhetően észlelhetőségük egyre nehezebbé válik, ugyanakkor egy fontos problémát is jelent, hiszen a webes adatforgalom jelentős részét produkálják. Mindamellett, hogy számos hasznos funkcionalitással rendelkeznek (webindexelés, weboldal figyelés, adatkinyerés kereskedelmi célokra), számottevő rosszindulatú tevékenység mögött is megfigyelhető jelenlétük (különböző hitelkártyaszámok, ajándékkártya-számok és bejelentkezéshez szükséges adatok kipróbálása, az összes rendelkezésre álló készlet megvásárlása egy adott termékből, hogy később magasabb áron értékesítsék vagy éppen fiókok létrehozása spam üzenetek küldésére) [10].

A legmodernebb megközelítések napjainkban botok detektálására gépi tanuláson alapulnak. Amint azt a weboldalak felhasználói megszokhatták, az egyik legelterjedtebb technikája a webbotok észlelésének a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), melynek különböző változatai léteznek: egyszerű képfelismerési feladatok, torzított karakterek felismerése vagy egyszerű jelölőnégyzetbe való klikkelés, mely során egyéb adatot is gyűjtenek a felhasználóról, ilyen például az egérmozgás.

Dolgozatom a webes botok detektálásának problémáját egérmozgások vizsgálata szempontjából közelíti meg. Kihívást jelent minél többféle szintetikus adathalmaz generálása, továbbá egy bot detektáló rendszer kialakítása a különböző generált egérmozgások kivédése érdekében. Szintetikus adatok generálása számtalan megközelítés alapján történhet, melyeket szinte lehetetlen mind feltérképezni. Ilyenek például a Bézier görbén alapuló kísérletek, generatív neuronhálók, autoenkóderek vagy bármely megközelítés, melyet az emberi egérmozgásokat megfigyelve alakítanak ki. A szintetikus adatok változatosságából adódóan a detektorok a félig felügyelt tanulásnak megfelelően kizárólag emberi adatok alapján tanulnak. A generált egérmozgások idősorok, melyekből érdemes kinyerni különböző jellemzőket a detektorok teljesítményének, illetve hatékonyságának növelése érdekében.

Az volt a célom, hogy minél jobban feltérképezsem az emberi egérmozgások esszenciáját annak érdekében, hogy különböző stratégiákat dolgozhassak ki emberszerű egérmozgások generálására, illetve felépítsek egy olyan rendszert, amely nagy pontossággal

el tudja különíteni egymástól a rendelkezésemre álló emberi egérmozgásokat a különböző megközelítések alapján generált egérmozgásoktól. Az általam kipróbált szintetikus adatgenerálási módszerek Bézier görbéken, illetve generatív ellenséges neuronhálókra alapulnak. Az első megközelítés, amely alapján az emberi egérmozgást próbáltam megközelíteni, a hagyományos harmadfokú Bézier-görbe (Cubic Bézier curve). Egy másik megközelítés ennek egy módosított változata, mely emberszerűbb mozgássá teszi a Bézier görbét. A harmadik megközelítés a TimeGAN, mely egy idősorok generálására szakosodott korszerű neuronháló. A szintetikus adatok azonosítását anomália detektorokkal végeztem. A detektorok teljesítményét az AUC (Area under the ROC curve) metrika alapján határoztam meg. A generált adat minőségét a detektorok teljesítménye határozza meg, hiszen minél sikeresebb egy detektor, annál gyengébbnek számít a generált adathalmaz. Gyengébb teljesítmény esetén pedig nem megbízható a rendszer. Egy megbízható bot detektáló rendszer kialakítása érdekében tehát egyre erősebb adathalmazokat generáltam, majd próbáltam ezeknek megfelelően javítani a detektorok teljesítményét.

Összesen 5 szintetikus adathalmazt állítottam elő, melyek idősorokból épülnek fel. Az egérmozgások idősoroként való reprezentációja nem hozott minden adathalmaz esetében jó eredményt, azonban az idősorokból kinyert jellemzők segítségével már minden esetben jó eredményeket értek el a detektorok. Természetesen a detektorok teljesítményét illetően vannak eltérések, de összességében elmondható, hogy a jellemzők alapján előállított minták esetében az AUC metrika szerint több kiemelkedő teljesítményű detektort is sikerült feltérképeznem. Az LOF detektor 0.81 fölötti AUC értéket ért el a legerősebb adathalmaz esetében is.

A kutatásra épülő szoftver a Google reCaptcha-hoz hasonló funkcionalitást valósít meg. Egyetlen egérmozgás alapján próbálja megszűrni egy weboldal felhasználóit, továbbá biztosít regisztrációs és belépési lehetőséget is. A szoftver továbbá egy gombnyomásra képes arra, hogy lejátsszon egy egérmozgást a képernyőn, megpróbálva kijátszani a rendszert a különböző stratégiákkal előállított szintetikus mozgások révén. Ez az esetek nagyon kis százalékában sikerül is. Amennyiben a felhasználó sikeresen teljesíti a tesztet, azaz a megfelelő koordináta-hoz viszi a kurzort, majd klikkel egyet a megfelelő helyre, lehetősége van bejelentkezni, illetve regisztrálni.

Dolgozatom további fejezetei a következőképpen vannak felépítve: a II. fejezetben áttekintem a bot detektálással kapcsolatos legfontosabb munkákat, illetve részletesebb leírást adok a CAPTCHA-ról. A III. fejezetben bemutatom az emberszerű egérmozgásokhoz alkalmazott generálási stratégiáimat, a generált adathalmazokat, valamint az általam használt anomália detektorokat. Ezt követően mérési eredményeimről számolok be az IV. fejezetben. Az V. fejezetben egy bot detektáló weboldalt mutatok be, amelyben a mérési eredmények alapján legjobban teljesítő anomália detektort használom a botok észlelésére. Dolgozatom a VI. fejezettel zárul, amely összegzi az elvégzett kutatást és bemutatja a továbbfejlesztési lehetőségeket.

2. fejezet

Szakirodalmi áttekintő

2.1. Bot detektálás

Botok detektálására léteznek kognitív készségeken (karakterfelismerés, képfelismerés stb.), illetve viselkedési biometrián alapuló módszerek is. Az első kategóriába tartozó feladatok már nem jelentenek kihívást a mesterséges intelligencia tudományterületén, míg az utóbbi kategória igen. Utóbbihoz tartoznak az egérmozgásokon vagy egyéb adatok mellett egérmozgásokra is alapozó kutatások. A következőkben ezeket fogom tárgyalni.

Wei és tsai. [?] az egérmozgások térbeli és kinematikai tulajdonságait figyelembe véve bevezetnek egy ábrázolási módszert, mely segítségével a mozgásokat képekké formálják. Ez az eljárás két részből áll. Elsősorban az egérmozgások térbeli információit veszik figyelembe a képalkotásban. Meghatározzák a bemeneti kép méretét az egérmozgások x és y koordinátáinak legnagyobb különbsége alapján, ezt követően a mozgást eltolják a kép közepére és végrehajtanak egy nagyítást. Az egérmozgásokat a képen 4 pixel sugarú pontok ábrázolják az x és y koordinátáknak megfelelően. Másodsorban kinematikai információkat is bevisznek a képbe azáltal, hogy színezik a pontokat. Minden pontnak a színét (RGB) két függvény segítségével határozzák meg. Az egyik függvény a ponthoz tartozó időbéllyeg alapján meghatározza az R értékét, a másik függvény a ponthoz tartozó megtett távolságból meghatározza a G értékét. Az egérmozgások begyűjtése érdekében egy weboldalt készítettek, mely egyaránt szolgált az emberi és bot egérmozgások begyűjtésére. Bot detektálásra konvolúciós neurális hálót használtak, melynek segítségével a botok 96.2 százalékát sikerült detektálni.

Iliou és tsai. [10] egy webbot-észlelési keretrendszert javasolnak, melyben a botok észlelését két szempont alapján vizsgálják: webnaplók és egérmozgások. Ez a két szempont lehetővé teszi a behatolások idő- és térbeliségének vizsgálatát. Rendszerük hatékonyságát tesztelték olyan web-botokkal, melyek rendelkeznek a böngésző ujjlenyomattal, illetve fejlettebb web-botokkal is, melyek az emberi viselkedést is képesek utánózni. Ehhez egy webszervert használtak, amelyhez emberi látogatók és szimulált webbotok is egyaránt hozzáfértek. Bot detektálási keretrendszerük elsősorban kiszámít egy pontszámot az egérmozgás alapján, majd ha ez az érték egy előre meghatározott küszöbérték alá esik, akkor figyelembe veszi a naplózás eredményét is.

Chu és tsai. [7] egy olyan rendszert javasolnak, amely kliensoldalon naplózza a felhasználók egérmozgásait és billentyű leütési adatait, majd ezek alapján egy szerveroldali osztályozó segítségével megállapítja, hogy a felhasználó ember vagy bot. Rendsze-

rük előnye, hogy passzív, tehát nem igényel a felhasználótól plusz erőfeszítést, mivel a felhasználó egy szessziója alatt végzett tevékenységének monitorizálásán alapszik. Megközelítésük hatékonyságának alátámasztása érdekében elsősorban egy aktív blogoldalról gyűjtöttek adatokat. Ez alapján feltérképezték az emberek és blogbotok közti különbségeket a weboldalak böngészésének és kommentek posztolásának tekintetében. E kétféle adat segítségével 99%-nál magasabb detektálási pontosságot sikerült elérniük.

Acien és tsai. [3] kidolgoztak egy *BeCAPTCHA-Mouse* nevezetű bot detektort a generált és valós egérmozgások felügyelt osztályozására. A detektor az egérmozgások neuromotoros modellezése során előállított jellemzőkkel dolgozik. A jellemzőkinyerést a SigmaLognormal modell segítségével valósítják meg, mely képes az emberi egérmozgások sebességprofilját lognormális alakkal leírni, mely kiválóan jellemzi az emberi agyban található motoros kéreg által irányított egérmozgások természetét. Az osztályozók teljesítményének mérésére heurisztikus függvények és egy generatív neuronháló segítségével állítanak elő szintetikus egérmozgásokat. Rendszerük átlagosan 93%-os pontosságot ér el egyetlen egérmozgáson alapuló detektálás során.

Antal és tsai. a [5] cikkben bot detektálás elősegítése érdekében megalkottak egy SapiAgent nevezetű modellt, mely egy kiváló alternatívát nyújt emberszerű egérmozgások előállítására. A SapiAgent megvalósítása érdekében autoenkódereket és egy új tanító algoritmust alkalmaznak. Kutatásukban bizonyítják, hogy az általuk javasolt generálási módszerrel realiztikusabb egérmozgások generálhatók, mint a Bézier görbékkel vagy a hagyományos autoenkóderekkel. Dolgozatomban megvalósítom az általuk javasolt Bézier görbéken alapuló generálási módszert, illetve tanulmányozok egy random végpontok szerinti generálási stratégiát, mely szintén Bézier görbéken alapszik. Ezen kívül további mozgásokat generálok a TimeGAN neuronháló segítségével. Az így előállított adathalmazokat használok fel arra, hogy különböző detektorok teljesítményét mérjem és megállapítsam, hogy melyik a leghatékonyabb ennek a feladatnak a megoldására, továbbá összehasonlítom az x és y irányú mozgások szerinti bot detektálást.

2.2. CAPTCHA

Napjainkban minden weboldal esetében szükség van arra, hogy valamilyen hitelesítésnek vessük alá a felhasználót, annak érdekében, hogy elkerülhessük a visszaéléseket és biztosítsuk a spamekkel szembeni védelmet. Erre egy jó alternatíva a reCAPTCHA¹, a Google ingyenes szolgáltatása, mely együttműködik képernyőolvasó szoftverekkel is. Ez egy fordított Turing-teszt, amely az ember számára könnyen és gyorsan végrehajthatónak, míg egy robot vagy rosszindulatú szoftver számára nehéznek bizonyul. A reCAPTCHA korábbi verziói, melyek képfelismerést és torzított szövegek felismerését alkalmazták, már kijátszhatók. A legújabb a v3-as verzió, mely mindössze egy jelölőnégyzetbe való kattintást igényel, azonban ha robotszerűnek minősíti a felhasználót, akkor előfordul, hogy bizonyos tárgyak felismerését is kéri tőlünk az oldal.

Botok detektálása történhet aktív és passzív módon. A CAPTCHA korábbi verziói az aktív kategóriába tartoznak, mivel a rendszer a felhasználókat online feladatok elvégzésére kéri, majd ez alapján dönti el a felhasználóról, hogy ember vagy sem. A passzív detektorok azáltal gyűjtenek információt az aktuális felhasználókról, hogy megfigyelik

¹<https://www.google.com/recaptcha/about/>

őket webes aktivitásuk során. A reCAPTCHA v3 verziója is ezt teszi amellet, hogy egy jelölőnégyzetbe való kattintást kér a felhasználótól.

A reCAPTCHA a felhasználók akaratától függetlenül beszerzett információkra épít, mint például az IP-cím és a sütik, valamint figyeli az egerünk mozgását, amikor az közeledik a jelölőnégyzethez. Ezeket az információkat és más titkos paramétereket is egyaránt használ a Google az emberi viselkedés detektálásához, majd meghatároz egy 0 és 1 közötti pontszámot. A 0 közeleiek nagyobb valószínűséggel utalnak botokra.

3. fejezet

Adatok és módszerek

3.1. SapiMouse adathalmaz

Munkám során rendelkezésemre állt a SapiMouse adathalmaz¹, mely 120 felhasználó egérdinamikai adatait tartalmazza. A méréseken résztvevő személyek a Sapientia Erdélyi Magyar Tudományegyetem munkatársai és diákjai közül kerültek ki önkéntes alapon, összesen 92 férfi és 28 nő, életkorukat tekintve 18 és 53 év közöttiek. Az önkéntesek többsége jobb kezes volt, csupán 9 személy volt balkezes a 120-ból. Az adathalmaz készítői egy webes játékot hoztak létre a mérések lebonyolítása érdekében, amelyben a felhasználóknak egérmozgásokat kellett végrehajtaniuk a képernyő különböző célkoordinátái között. Az alkalmazás folyamatosan naplózta az egér pozícióját egy körülbelül 60 Hz-s mintavételezési frekvenciával. A játék több feladatból állt, különböző geometriai formákat jelenített meg véletlenszerű pozíciókban a képernyőn (háromszög, fordított háromszög, négyzet, kör). A felhasználóknak ezen geometriai alakzatokhoz kellett mozgatniuk a kurtort, majd egy bal, jobb vagy dupla kattintást, esetenként egy drag-and-drop műveletet kellett végrehajtaniuk. Ezen műveletek alatti mérések során létrejött egy adathalmaz, mely minden feladat esetén az egérkurzor pozíciójából (x, y) , gombtípusból, eseménytípusból (mozgatás, húzás, lenyomva vagy felengedve), és a megfelelő időbélyegből épült fel. Az önkéntesek saját számítógépükön hajtották végre a feladatokat és előzetesen tájékoztatva voltak arról, hogy tevékenységük naplózva lesz. Két munkamenetet kellett teljesíteniük, melyek egyike 3, másik 1 percet vett igénybe. Mindkét munkamenet során a felhasználóknak a fent említett feladatokból kellett megoldaniuk minél többet. Mind az adathalmazt, mind a webalkalmazást publikussá tették a szerzők [5].

3.1.1. Adathalmaz szegmentálása

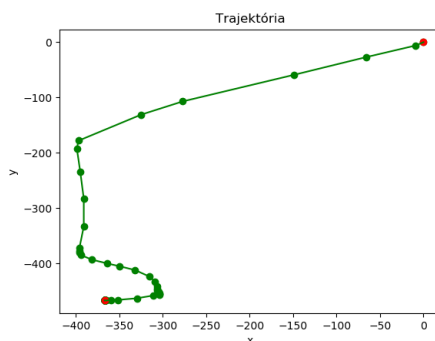
A szegmentálás során a 3.1 fejezetben ismertetett feladatokban előforduló jobb, bal és dupla klikkeléseket vettem figyelembe, valamint két ilyen művelet között végrehajtott egérmozgást tekintettem egy-egy trajektóriának (lásd 3.1 ábra). Egy egérmozgást reprezentáló 2D-s pontsorozat tehát egy trajektória: $\{(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)\}$. Mivel a trajektóriák hossza változó és egyes algoritmusok esetében fix hosszúságú bemenetre van szükség, 128 hosszúságú mintákkal dolgoztam. A mintákat úgy állítottam elő, hogy a 10

¹<https://www.ms.sapientia.ro/~manyi/sapimouse/sapimouse.html>

pontnál több pontot tartalmazó trajektóriákból számítottam elsőrendű különbségeket:

$$D = \{dx_1, dx_2, \dots, dx_{128}; dy_1, dy_2, \dots, dy_{128}\}, \quad (3.1)$$

ahol $dx_i = x_i - x_{i-1}$ és $dy_i = y_i - y_{i-1}$. A 128-nál rövidebb mintákat nullásokkal egészítettem ki, a hosszabbakat vágással rövidítettem le.



3.1. ábra. Egy emberi trajektória ábrázolása

Összesen 2 adathalmazt állítottam elő az emberi egérmozgások alapján, amelyeket a következőképpen fogok jelölni:

1. *1min* - 1 perces munkamenetek egérmozgásai alapján vett 128 dx és 128 dy
2. *3min* - 3 perces munkamenetek egérmozgásai alapján vett 128 dx és 128 dy

A 3.1 táblázatból leolvasható, hogy az emberi egérmozgások esetében a medián értéket a 34 dx és ugyanennyi dy elmozdulásból álló minta adja, valamint a trajektóriák kevesebb, mint 1 százaléka tartalmaz 118-nál kevesebb pontot. Ez azt jelenti, hogy a vágások az adathalmaz kevesebb, mint 1 százalékát érintik.

	<i>1min</i>	<i>3min</i>
minimum	11	11
Q1 (25%)	27	27
medián	35	34
Q3 (75%)	44	44
99%	118	124
maximum	350	362

3.1. táblázat. Statisztika a trajektóriák hosszáról (hány darab $\{dx, dy\}$ elempárból tevődnek össze)

3.2. Görbe generálási módszerek

3.2.1. Bézier görbék

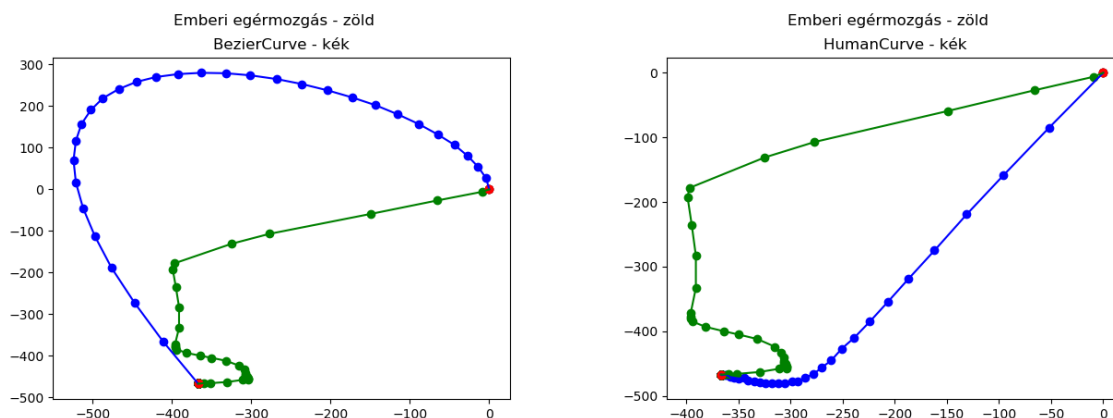
A Bézier görbe [1] egy elterjedt módszer két pont közti görbe előállítására. Ez egy parametrikus görbe, mely kontrollpontok és egy t paraméter függvényében generálódik.

Az első és utolsó kontrollpont rajta van a görbén, míg a többi a görbe alakulását befolyásolja. A t paraméter a $[0,1]$ intervallumból vesz fel egymástól egyenlő távolságra eső értékeket. A görbe köztes pontjainak száma ennek az intervallumnak a felosztásától függ.

3.2.2. Generált adathalmazok

Két pont közti trajektóriák generálása a *pyclick*² csomag segítségével történt, mely Bézier és emberszerű Bézier görbék generálását valósítja meg.

Az első adathalmazom, melyre *randomBezier* adathalmazként fogok hivatkozni, a kiinduló adathalmaz (baseline), melyhez a *pyclick* csomag *BezierCurve* osztályát alkalmaztam. Harmadfokú Bézier-görbéket (Cubic Bézier curve) állítottam elő a következő stratégiának megfelelően: generáltam véletlenszerűen 4 kontrollpontot (egyenletes eloszlást követő random generátort használva) abból az intervallumból, amelybe az emberi trajektóriák kezdő és végpontjai közti távolságok beleesnek. Ez tulajdonképpen egy téglalap alakú terület a képernyők rezolúciójának megfelelően. A görbe köztes pontjai számának a meghatározása érdekében kiszámítottam az első és utolsó kontrollpont közötti távolságot, majd figyelembe véve az emberi egérmozgások kezdő és végpontjai közötti összes meg egyező távolságot meghatároztam az átlagos köztes pontok számát, majd ezt az értéket alkalmaztam. A kapott egérmozgások esetében elsőrendű különbségeket számoltam, mint a korábban említett SapiMouse adathalmaz esetében (lásd 3.1.1 alfejezet).



(a) Emberi és generált (*Bézier*) egérmozgás ábrázolása (b) Emberi és generált (*HumanCurve*) egérmozgás ábrázolása

3.2. ábra

A második adathalmazom, melyre *humanLike* néven fogok hivatkozni, a *pyclick* csomag *HumanCurve* osztálya segítségével generáltam, mely harmadfokú Bézier-görbén (Cubic Bézier curve) alapul. Ez az osztály úgy volt megírva, hogy az emberi mozgásokhoz hasonló mozgásokat állítson elő. Annak érdekében, hogy a lehető leghatékonyabban használjam fel ezt az osztályt, a következő stratégia szerint generáltam az egérmozgásokat: a mozgások kezdő és végpontjait, valamint a köztük lévő pontok számát az emberi egérmozgások alapján határoztam meg, azaz minden generált mozgás egy emberi mozgás

²<https://github.com/patrikoss/pyclick>

kezdő és végpontja között jön létre, továbbá ugyanannyi pontból tevődik össze, mint az aktuális emberi mozgás. Kivételt képeznek azok az esetek, amikor az emberi mozgások köztes pontjainak száma meghaladta a 127-et, ebben az esetben ugyanis 127 köztes pontot generáltam a vágás elkerülése érdekében. A kapott egérmozgások esetében elsőrendű különbségeket számoltam, majd kiegészítettem nullás értékekkel a 128-nál kevesebb elmozdulásból álló mintákat, csakúgy, mint a korábban említett SapiMouse adathalmaz esetében (lásd 3.1.1).

A fent tárgyalt stratégiát alkalmaztam a *BezierCurve* osztályra is, így előállítottam a *bezier* adathalmazt. Az *randomBezier* adathalmaz előállításánál használt stratégiát alkalmazva a *HumanCurve* osztály esetében előállítottam a *randomHumanLike* adathalmazt.

A *pyclick* csomag segítségével összesen 4 adathalmazt állítottam elő, amelyeket a következőképpen fogok jelölni:

1. *bezier* - emberi egérmozgások végpontjai és köztes pontjai számán alapuló *BezierCurve* görbék alapján vett 128 dx és 128 dy
2. *humanLike* - emberi egérmozgások végpontjai és köztes pontjai számán alapuló *HumanCurve* görbék alapján vett 128 dx és 128 dy
3. *randomBezier* - random végpontok és statisztikailag becsült köztes pontok számán alapuló *BezierCurve* görbék alapján vett 128 dx és 128 dy
4. *randomHumanLike* - random végpontok és statisztikailag becsült köztes pontok számán alapuló *HumanCurve* görbék alapján vett 128 dx és 128 dy

A 3.2 ábrán a pirossal jelölt végpontok között megfigyelhető az emberi (zöld) és a generált(kék) egérmozgások alakulása. Az emberi egérmozgásokra jellemző a kezdeti gyorsulás, esetenként a végponthoz közeledve pedig a lassulás. Ahogy a 3.2b ábrán is látható, a szintetikus mozgás az emberi mozgáshoz hasonlóan kezdetben gyorsul, míg a 3.2a esetében ez a tulajdonság nincs jelen.

3.3. Idősorok generálása TimeGAN neuronhálóval

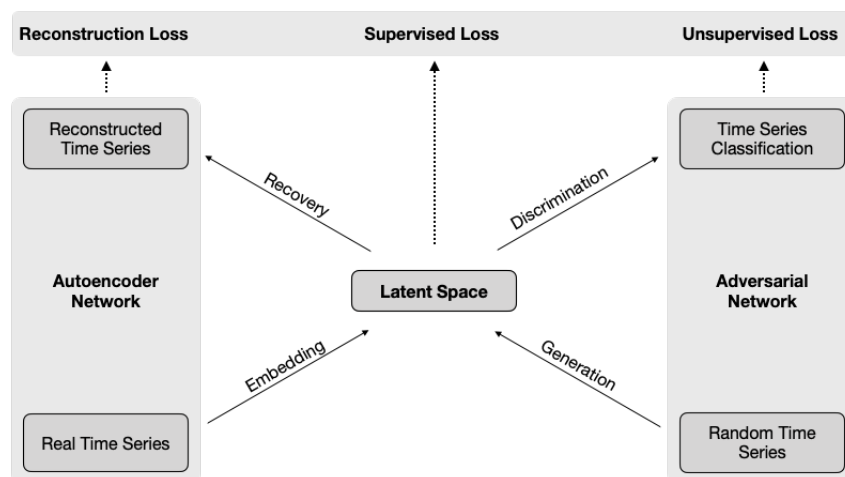
3.3.1. TimeGAN

A generatív ellenséges neuronhálók (GAN - Generative Adversarial Network) két kompetitív hálóból tevődnek össze. Az egyik a generátor, mely random zajból állít elő szintetikus adatokat, a másik a diszkriminátor, melynek célja a valós és generált adatok megkülönböztetése. A cél az, hogy a generátor olyan adatot tudjon előállítani, melyet a diszkriminátor nem tud megkülönböztetni a valós mintáktól.

Az idősorok szekvenciális adatok, melyek fontos tulajdonsága az időbeliség. Ennek következtében a változók közötti eloszláson túl a modellnek meg kell tanulnia az adatok temporális dinamikáját is. A TimeGAN egy generatív neuronháló, mely sikeresnek bizonyult szekvenciális adatok időbeliségének elsajátításában. Ezt a szerzők a felügyelt és

felügyelet nélküli tanítás ötvözésének technikájával támasztják alá. A TimeGAN³ a [12] cikkben bemutatott neuronháló implementációja.

A TimeGAN architektúra 2 fő komponensből áll (3.3 ábra). Az első rész egy autoenkóder tartalmaz (3.3 ábra bal oldala), ez két neuronhálóból épül fel: embedding (beágyazó) és recovery (visszanyerő) hálók. A második rész a tulajdonképpeni GAN (3.3 ábra jobb oldala).



3.3. ábra. TimeGAN architektúra

A TimeGAN architektúra kulcseleme, hogy mind a generátor, mind a beágyazó hálózat felelős a felügyelt veszteség minimalizálásáért, amely azt méri, hogy a modell mennyire tanulja meg a dinamikus kapcsolatot.

Az autoenkóder feladata a jellemzőtér leképezése egy látens térbe, majd ennek a fordítottja, annak érdekében, hogy lehetővé tegye a GAN számára az adatok időbeli dinamikájának egy alacsonyabb dimenziós térben történő tanulását, ezáltal megkönnyítve ezt a folyamatot.

A TimeGAN generátor eleme a szintetikus jellemzőket a látens térben állítja elő.

Az előbb ismertetett neuronhálók mellett van egy felügyelő neuronháló, mely arra ösztönzi a generátort, hogy tanulja meg az adathalmaz időbeli dinamikáját.

Amint a 3.3 ábrán látható, 3 költségfüggvény járul hozzá a TimeGAN optimalizálásához a közös tanítás során (joint training):

- az autoenkóderhez tartozó rekonstrukciós veszteség (reconstruction loss) - megmutatja, hogy mennyire volt sikeres a kódolt adatok rekonstrukciója az eredetiekkel összehasonlítva
- felügyelt veszteség - kifejezi, hogy a generátor mennyire jól közelíti meg a következő időpillanatot a látens térben
- nem felügyelt veszteség - a generátor és a diszkriminátor közötti kompetitív interakciót tükrözi

³https://github.com/stefan-jansen/machine-learning-for-trading/tree/main/21_gans_for_synthetic_time_series

A felhasznált veszteségfüggvények az MSE (Mean Squared Error) és a BCE (Binary Cross Entropy), az optimalizáló pedig az Adaptive Momentum (Adam). A közös tanítás eredményeként a TimeGAN egyszerre tanulja meg az adatok kódolását (látens tér), a reprezentációk generálását és az időbeli iterációt.

A tanítás során 3 fázist különítenek el:

- autoenkóder tanítása valós adatokkal egy optimális rekonstrukció elérése érdekében,
- felügyelt veszteség minimalizálása az adatok időbeli dinamikájának feltérképezése érdekében,
- a hálók közös tanítása (a 3 költségfüggvény minimalizálása).

A TimeGAN architektúráját illusztráló ábrák az 6. fejezetben tekinthetők meg.

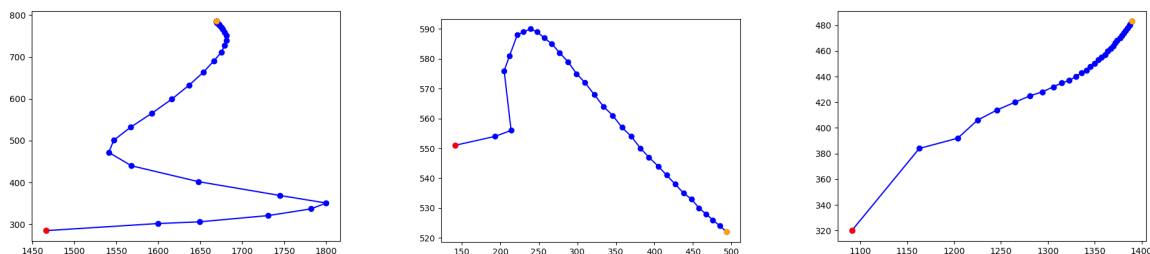
3.3.2. A TimeGAN fontosabb paraméterei

A tanítási adathalmaz az eredeti Sapimouse adathalmaz 3 perces adataiból épül fel, így a neuronháló bemenetét az egér pozíciójának x és y koordinátái jelentik 32 egymást követő időpillanatban. A szekvenciák hossza (32) az egérmozgások átlagos hosszával egyenlő. A köteg (batch) hossza 128. A felügyelő két rejtett réteget tartalmaz, míg a többi 4 háló esetében ez a paraméter 3-al egyenlő. A rejtett rétegek dimenziója 28, erről a 4.3.3 fejezetben lehet bővebben olvasni.

Mind az 5 neuronháló hasonló szerkezeti felépítéssel rendelkezik:

- beágyazó - 3 réteg, rejtett rétegek és kimeneti réteg 28 GRU (Gated Recurrent Unit) egységet tartalmaznak
- visszanyerő - 3 réteg, rejtett rétegek 28, kimeneti réteg 2 GRU egységet tartalmaz
- generátor - 3 réteg, rejtett rétegek és kimeneti réteg 28 GRU egységet tartalmaznak
- diszkriminátor - 3 réteg, rejtett rétegek 28, kimeneti réteg 1 GRU egységet tartalmaz
- felügyelő - 2 réteg, rejtett rétegek és kimeneti réteg 28 GRU egységet tartalmaznak

A GRU egység előnye, hogy fejleszti a neuronháló memóriakapacitását az egyszerű RNN egységekhez képest, mely nem alkalmas a túl korai bemenetekből való tanulásra.



3.4. ábra. TimeGAN-al generált egérmozgások

A TimeGan neuronhálózval előállított 32 pontpárból felépülő egérmozgásokat tartalmazó adathalmazra *randomTimeGan* néven fogok hivatkozni. A 3.4 ábra három szintetikus egérmozgást szemléltet ebből az adathalmazból.

3.4. Bot detektálás anomália detektorokkal

3.4.1. Anomália detektálás

Az anomália detektálás több szakterület esetében is fontos és kutatott témának számít. Alkalmazzák többek között a hitelkártyákkal kapcsolatos csalások felderítésében, képfeldolgozásban, behatolás észlelésnél, kiberbiztonságban és kritikus rendszerek hiba észlelésénél is. Lényege, hogy olyan mintákat találjunk egy adathalmazban, amelyek nem az elvárt módon viselkednek, azaz különböznek az adatok többségétől. Az adatelemzésben ezeket a ritkábban előforduló elemeket anomáliáknak nevezzük, azonban használatos az outlier, kivétel (exception) stb. megnevezés is. Jelentőségük abban nyilvánul meg, hogy jelenlétük egy-egy kritikus történés következménye, mint például lopás, adatszivárgás, rosszindulatú daganat vagy megnövekedett webes adatforgalom [6].

Anomália detektálás során definiálnunk kell egy régiót, amely a normális viselkedésű adatokat tartalmazza. Ez a folyamat számos nehézséget tartogat, mivel nehéz egy határvonalat képezni a helyes adatok és anomáliák között. Előfordulhat, hogy ez a határ eltolódik és helytelenül címkézünk meg egy mintát, illetve azt is figyelembe kell vennünk, hogy amikor az anomáliák rosszindulatú cselekedetek következményei, előfordulhat, hogy a rendellenes megfigyeléseket normálisnak álcázzák, ezáltal megnehezítve a normális viselkedés meghatározását. Emellett figyelni kell arra, hogy a normál viselkedés változhat, így adaptálódnunk kell a változásokhoz annak érdekében, hogy a normális fogalma reprezentatív maradjon [6].

Fontos megkülönböztetni a zajt az anomáliáktól. A zajos adatokat nem tudjuk hasznosítani, így nem érdekesek a feldolgozás szempontjából, az anomáliáknak azonban számottevő hozadékuk van egy detektálás esetén. Az aktuális adathalmaz, amellyel dolgozom nem tartalmaz zajt, mivel célzottan lett gyűjtve.

Anomália típusok [6]:

- Pont anomáliák: Ide tartoznak azok az adatpéldányok, amelyekre rámondható, hogy eltérnek az adathalmaz többi elemétől. Például egy ponthalmaz esetében egy ábrázolással is kimutatható, ha az egyik pont eltávolodik az egymáshoz közel elhelyezkedő pontok tartománytól.
- Kontextuális anomáliák: Abban az esetben beszélhetünk erről a kategóriáról, ha függ a kontextustól, hogy egy minta anomália vagy sem. Ez a jelenség megfigyelhető például a mérsékelt égöv hőmérsékleti adatainak idősorán (time series), ugyanis egy negatív hőmérséklet télen normálisnak, míg nyáron anomáliának tekinthető.
- Kollektív anomáliák: Ebben az esetben egymáshoz kapcsolódó adatpéldányok kollektíója mutat eltérő viselkedést a teljes adathalmazhoz képest, az adatpéldányok önmagukban nem számítanak anomáliáknak.

Ahhoz, hogy adatainkat besoroljuk a normál és anomália kategóriákba, címkézést kell végrehajtanunk. Ez egy komplex folyamat, mivel nehéz egy olyan halmazt kialakítani akár normál, akár anomáliákat tartalmazó halmazból, amely lefedi az összes lehetőséget az adott kategóriában. Címkézettség szempontjából három típusú anomália detektálási módszert említenek:

- Felügyelt anomália detektálás (Supervised anomaly detection): A tanítási adathalmaz normális adatokat és anomáliákat is tartalmaz és ezek fel vannak címkézve.
- Félig felügyelt anomália detektálás (Semi-Supervised anomaly detection): A tanítási adathalmaz csak normális adatokat tartalmaz.
- Felügyelet nélküli anomália detektálás (Unsupervised anomaly detection): A tanítási adathalmaz címkézetlen és tartalmaz normális adatokat illetve anomáliákat is. Fontos, hogy a normális adatok gyakrabban fordulnak elő az adathalmazban, mint az anomáliák.

A különböző anomália detektáló algoritmusok kimenete kétféle lehet: pontszám vagy címke. Utóbbi esetben címkéket kapunk kimenetként, melyek azt jelölik, hogy az adat anomália volt vagy sem. Az általam használt algoritmusok az előbbi típushoz tartoznak, azaz minden adatnak megfelelően egy pontszámot (score) térítenek vissza. Ez a pontszám számszerűsíti az eltérés, vagyis a "kiugrás" mértékét [4].

3.4.2. Osztályozás

Minden anomália detektáló algoritmus egy modellt épít fel a normál minták alapján, majd ezt felhasználva határozza meg az újabb minták pontszámát (outlier score). Az anomália detektoroknak számos típusa létezik. Megkülönböztetünk valószínűség alapúakat (pl. COPOD), közelség alapúakat (kNN, LOF), léteznek lineárisak modellek (PCA, OCSVM), valamint ötvözöttek (IForest). Az említett detektorokat a PyOD [13] szolgáltatja, mely egy Python-eszközkészlet a többváltozós adatokban található anomáliák észlelésére.

3.4.3. Fontosabb detektorok

COPOD

A Copula-alapú kiugró érték detektálás (COPOD - Copula-Based Outlier Detection) egy olyan algoritmuson alapszik, amelyet a többváltozós adateloszlás modellezésére szolgáló kopulák (copula) inspiráltak. A COPOD először empirikus kopulát készít, majd ennek segítségével előre jelzi az egyes adatpontokhoz tartozó tail (farok) valószínűséget. Ez az érték határozza meg az adatpont kiugrásának a mértékét, amely minél magasabb, annál valószínűbb, hogy az adott minta anomália. Ezáltal a COPOD paramétermentes és számítási szempontból is hatékony [11].

OCSVM

Az egyosztályú tartóvektor-gép (OCSVM - One-class support vector machine) a lineáris és logisztikai regressziós modellek változatainak tekinthető, ahol a *határ* (*margin*) fogalma a túltanítás elkerülését segíti elő, csakúgy, mint a regularizáció a regressziós modellek esetében. Ezt a modellt két osztály szeparálására hozták létre, azonban anomália detektálás során a rendelkezésünkre álló címkézetlen adatok csupán a normál osztály példányait tartalmazzák. Ennek a problémának a kiküszöbölése érdekében feltételezzük,

hogy egy kernel-alapú transzformált reprezentációnak az origója az anomáliák osztályához tartozik [4]. Az SVM egyosztályos esetben egy olyan hipersíkot határoz meg, amely elválasztja az adatpontokat az origótól.

A két osztály közti döntési határ a 3.2 képlet segítségével írható le. Tekintsük a Φ ismeretlen függvényt, mely az $\bar{X}$ adatpontot $\Phi(\bar{X})$ -be transzformálja. A $\Phi(\bar{X})$ egy nagy dimenziójú vektor a transzformált térben, a $\bar{W}$ vektor pedig a neki megfelelő együtthatókat tartalmazza, a b paraméter jelenti az eltolást.

$$\bar{W} \cdot \Phi(\bar{X}) - b = 0 \quad (3.2)$$

Ezt a transzformációt nem kell explicit módon végrehajtani, mivel egy optimalizálási problémán belül implicit módon végrehajtható. Az optimalizálási probléma megfogalmazása: *A $\bar{W} \cdot \Phi(\bar{X}) - b$ értéke pozitív kell legyen a lehető legtöbb tanítási mintára, mivel az összes tanítási mintáról feltételezzük, hogy a normál (pozitív) osztályhoz tartozik.*

Miután betanítottuk a modellt, azaz megtanulta a döntési határt, utána használhatjuk ezt új minták besorolására. A besorolás a mintákhoz egy pontszámot (outlier score) értéket rendel, amely lehet negatív vagy pozitív. A negatív azt jelenti, hogy anomália az adott minta, míg a pozitív érték a normális kategóriába való sorolást jelenti.

IForest

Az izolációs erdő (IForest - Isolation forest) egy olyan anomália detektáló algoritmus, mely felépít egy több random fából álló erdőt, majd ezt átlagolja. A random fák felépítése tulajdonképpen az adatpontok szétválasztását, egymástól való szeparálását jelenti. Kezdetben egyetlen csomópontból álló bináris fából indulunk ki, mely egyetlen csomópontja tartalmazza az összes adatpontot, majd a vágásoknak megfelelően mélyül a fa, egészen addig, ameddig minden adatpont egy külön csomópontban fog elhelyezkedni. Ez egy rekurzív folyamat, mely során egy random módon választott jellemzőnek megfelelő legkisebb és legnagyobb érték közé eső random pont mentén szigetelődnek el az adatpontok egymástól, azaz a tengelyekkel párhuzamos vágásoknak megfelelően. Mivel az anomáliák ritkább régiókban helyezkednek el, elkülönülnek a többi adatpont által alkotott klaszterektől és könnyebben izolálhatók. Ezáltal az anomáliákat tartalmazó faágak kevésbé lesznek mélyek, így a gyökértől számított távolságuk is kisebb. Ezekből a távolság értékekből számított átlag jelenti egy adott minta pontszámát (score) [4].

KNN

A k -legközelebbi szomszéd (KNN - k-Nearest Neighbors) algoritmus távolság alapú, mivel mindig az aktuális adatpont szomszédságában elhelyezkedő adatpontoktól mért távolságokat veszi figyelembe. Egy adott mintához társított pontszám (score) meghatározásának több változata is van. Az első, amikor a pontszám egyenlő az aktuális adatpont és a hozzá legközelebb eső k -adik szomszéd távolságával. Ebben az esetben a k értéke 1-nél nagyobb (különben $k = 1$ esetén minden pontszám 0 lenne) és n -nél kisebb, ahol n a minták száma. Ennek a módszernek a legkritikusabb része a megfelelő k szám meghatározása. Ennek a problémának megfelelően egy robusztusabb alternatíva, ha a k darab legközelebbi szomszéd távolságának átlagát tekintjük egy adott minta pontszámának. További módszerek a pontszám meghatározására a medián vagy a harmonikus közép kiszámítása.

Minden módszer esetében a magas pontszámmal rendelkező minták tekinthetők anomáliáknak [4].

LOF

Az LOF (Local Outlier Factor - lokális kiugró érték tényező) egy sűrűség alapú detektor. Egy mintának megfelelő lokális sűrűséget úgy határozzuk meg, hogy figyelembe vesszük a pontok számát annak egy specifikus lokális régiójában. Ez az érték arányos az izoláltság szintjével. Összehasonlítva egy minta lokális sűrűségét a szomszédai lokális sűrűségeivel, egy alacsonyabb értékkel rendelkező pont izolált pontként anomáliának minősül.

A 3.3 képlet alapján megfigyelhető, hogy minden $(\bar{X}, \bar{Y})$ adatpontpár esetén, számítunk egy elérhetőségi távolságot $(R_k(\bar{X}, \bar{Y}))$, mely a maximum érték az alábbiak közül: a két adatpont közti távolság vagy az $\bar{Y}$ távolsága a k -adik legközelebbi szomszédjától. Megállapítható, hogy ha az $\bar{Y}$ egy sűrű régióban van, akkor a maximum az előbb említett távolság lesz, viszont ha az $\bar{X}$ és $\bar{Y}$ közötti távolság kicsi, akkor ez az érték az utóbbival lesz egyenlő [4].

$$R_k(\bar{X}, \bar{Y}) = \max\{dist(\bar{X}, \bar{Y}), D_k(\bar{Y})\} \quad (3.3)$$

Az $\bar{X}$ adatpont átlagos elérhetőségi távolsága (lásd 3.4 képlet), a szomszédságában előforduló objektumoktól számított elérhetőségi távolságainak átlagával lesz egyenlő [4].

$$AR_k(\bar{X}) = MEAN_{\bar{Y} \in L_k(\bar{X})} R_k(\bar{X}, \bar{Y}) \quad (3.4)$$

Végül a mintákhoz társított pontszámokat úgy kapjuk meg, hogy az aktuális pont elérhetőségi távolságainak átlagát elosztjuk az összes szomszédos pontja elérhetőségi távolságainak harmonikus közeparányosával. Ez tulajdonképpen ekvivalens a minta elérhetőségi távolságának és az összes szomszédos pontja elérhetőségi távolságainak az átlagával.

$$LOF_k(\bar{X}) = \frac{AR_k(\bar{X})}{HMEAN_{\bar{Y} \in L_k(\bar{X})} AR_k(\bar{Y})} \quad (3.5)$$

Létezik az LOF-nek olyan változata is, amelyben az elérhetőségi távolságok helyett használhatjuk az euklideszi távolságokat, illetve a harmonikus átlag helyett az átlagot.

PCA

A főkomponens analízis (PCA - Principle component analysis) egy dimenzió csökkentő eljárás, mely segítségével megadható bármely d -dimenzió hipersíkjainak optimális reprezentációja. Ez azt jelenti, hogy előállíthatók azok a k dimenziós hipersíkok (bármilyen $k < d$ érték esetén), amelyek minimalizálják a négyzetes vetítési hibát a fennmaradó $(d - k)$ dimenziók felett, azonban az anomáliák esetében ez a hiba magas [4]. Főkomponens analízis során az adathalmaz kovariancia mátrixának k legnagyobb sajátértékének megfelelő sajátvektorainak a lineáris kombinációja olyan tengelyrendszert ad, amelyben az adatok nagyon kis veszteséggel reprezentálhatók. A transzformált adathalmaz varianciája az első sajátvektor (főkomponens) mentén a legmagasabb, ezt követően csökken. A kis sajátértékkel rendelkező sajátvektorok mentén tehát kicsi a variancia. Ez az anomália detektálás aspektusában egy releváns tényező, mivel a transzformált adathalmaz azon

elemei, melyek jelentősen eltérnek az átlagértékektől ezen irányok mentén, anomáliák lehetnek. Egy minta pontszámát (score) úgy kapjuk meg, hogy vesszük minden sajátvektorra (vagy a sajátvektorok egy részhalmazára) nézve a vetített euklideszi távolságok négyzetösszegét. Ezt az értéket súlyozni szokás a megfelelő sajátértékekkel. Ez az intuíció abból adódik, hogy mivel a sajátvektorok menti varianciákat a sajátértékek adják, az eltérés egy kisebb sajátértékkel rendelkező sajátvektor mentén nagyobb súllyal kell számítódjon, mint egy nagy sajátérték mentén.

3.5. Adatelemzés

Adatelemzés során hisztogramokat készítettem a dx , dy értékekre vonatkozóan, valamint megvizsgáltam a trajektóriákat a kezdő- és végpontok távolságának, illetve a köztes pontok számának függvényében.

A 3.5 ábra hisztogramjai az $1min$ és $3min$ adathalmazok alapján készültek. Az (a) ábrán látható a trajektóriák végpontjai között megtett utak hosszainak hisztogramja, a távolságok pixelben vannak megadva. A (b) ábra szemlélteti a trajektóriák köztes pontjai számának hisztogramját. Végül a (c) ábra mutatja a trajektóriák hossza és az annak megfelelő köztes pontok száma közötti összefüggést. Jellemzően a távolság növekedésével a köztes pontok száma is nő.

A hisztogramokra elsősorban a jellemzők előállításánál volt szükségem. A 3.6 (a) és (b) ábrákon láthatók az $1min$ és $3min$ adathalmazokból vett dx , dy értékek hisztogramjai. A sűrűségfüggvények (piros görbék) alapján látható, hogy a hisztogramok milyen mértékben követik a normál eloszlást. A nullához közeli értékek túlsúlyban vannak, így mindkét hisztogram kicsúcsosodik, ugyanakkor az is látszik, hogy a dx értékeknek nagyobb a szórása, mint a dy értékeknek.

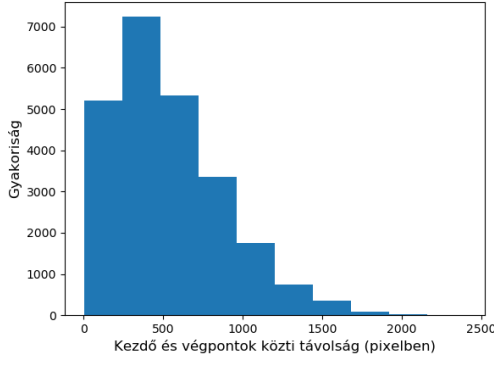
Ugyanezeket a jelenségeket figyeltem meg a *humanLike* szintetikus adathalmaz esetében is - 3.6 (e) és (f) ábra. Ezzel szemben a Bézier adathalmazok 3.6 (c) és (d) hisztogramjaira kevésbé jellemző a kicsúcsosodás, így jobban követik a normál eloszlást. Mivel minden hisztogram esetében megfigyelhető a szimmetria, abszolút értékekkel dolgoztam a hisztogram alapú jellemzőkinyerésnél.

3.6. Jellemzőkinyerés

A korábban említett dx , dy értékekből minden trajektória (minta) esetén összesen 29 jellemzőt állítottam elő. Ezek jellegüket tekintve az alábbiak: statisztikai, hisztogram-alapú, simasággal és a mozgás hatékonyságával kapcsolatos jellemzők. Ezeket részletesen az alábbi bekezdésekben ismertetem.

Az első csoporthoz olyan statisztikából ismert jellemzők tartoznak, melyek relevánsnak bizonyultak az egérmozgásokra vonatkozóan. Ilyenek a minimum, kvartilisek (első, harmadik), medián, maximum, módusz, ferdeség (skewness). Ezek mindegyikét kiszámítottam a dx -ek és dy -ok esetében is, így összesen 14 jellemzővel bővült a jellemzők listája.

A hisztogram alapú jellemzőket úgy állítottam elő, hogy minden minta esetén egy 5 bin-es felosztást használva kiszámítottam, hogy melyik intervallumba hány százaléka esik bele a dx , illetve dy értékeknek. Ez 10 jellemzőt eredményezett összesen (5 dx alapján,



(a) Trajektóriák hossza



(b) Trajektóriák köztes pontjainak száma



(c) Trajektóriák köztes pontjainak száma vs trajektóriák hossza

3.5. ábra. SapiMouse statisztikák

5 dy alapján). A bineket empirikusan határoztam meg, annak érdekében, hogy a lehető legjobban elkülöníthetők legyenek egymástól a különféle típusú mozgások.

A szakirodalomban [9] megemlítik a mozgás hatékonyságát (movement efficiency), mint egy diszkriminatív jellemzőt a botok és emberek egérmozgásaira vonatkozóan (a botok hatékonyabbak, mint az emberek). A hatékonyság a két pont közti távolság és a megtett útvonal hosszának aránya.

A szakirodalomban említett simaság (smoothness) is releváns jellemzőkkel bővítette a meglévőket. A [5] cikk alapján a dx , dy elsőrendű különbségekből számítottam további másodrendű különbségeket a 3.6 képletnek megfelelően, majd ezt továbbgondolva kiszámítottam a harmadrendű különbségek alapján vett simaságot is. A képletet alkalmaztam a dx és dy értékek esetében is külön-külön, kiszámítva egy vízszintes és egy függőleges irányú simaságot.

$$smoothness(z_1, z_2, \dots, z_n) = \frac{\sum_{i=2}^n |z_i - z_{i-1}|}{n - 1} \quad (3.6)$$

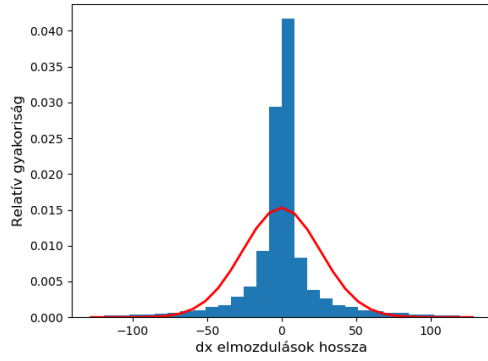
3.6.1. Szintetikus adatok minősége

Az *sdmetrics* Python csomag⁴ lehetővé teszi a generált (szintetikus) adatok minőségének mérését. Első lépésben elvégeztem a jellemzőkinyerést minden egyes adathalmazra, majd a minőség mérésénél a jellemzők eloszlásainak hasonlóságát mértem az eredeti és a generált adathalmaz esetében. Az *sdmetrics* csomag egy KSTest (Inverted Kolmogorov Smirnov D statistic) nevű metrikát ajánl a numerikus adatsorok összehasonlításához. A kapott érték egy 0 és 1 közötti érték, amely annál nagyobb minél hasonlóbb a két adathalmaz. A 3.2 táblázat szemlélteti a szintetikus adathalmazok minőségét. Látható, hogy a legnagyobb értéket a *humanLike* adathalmaz kapta, a legkisebbet pedig a *bezier* adathalmaz.

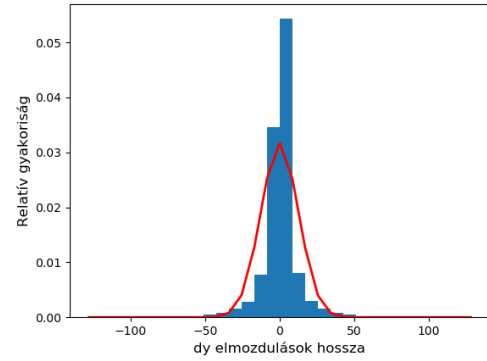
KSTest metrika - Single Table	
<i>bezier</i>	0.575
<i>randomBezier</i>	0.606
<i>humanLike</i>	0.809
<i>randomHumanLike</i>	0.774
<i>randomTimeGan</i>	0.752

3.2. táblázat. Szintetikus adathalmazok minősége. A nagyobb értékek magasabb minőséget (emberszerűbb) jelentenek.

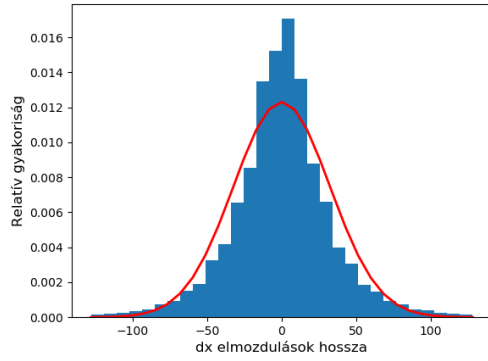
⁴<https://github.com/sdv-dev/SDMetrics>



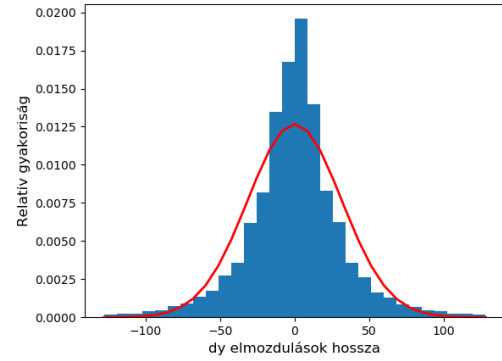
(a) Az emberi egérmozgások dx elmozdulásainak hisztogramja



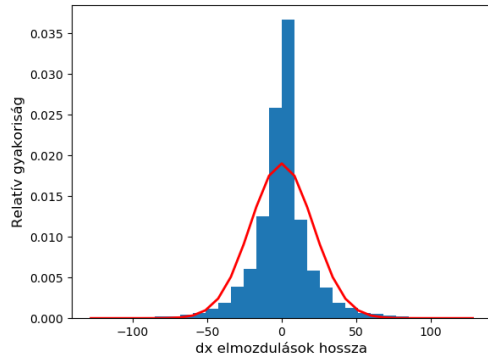
(b) Az emberi egérmozgások dy elmozdulásainak hisztogramja



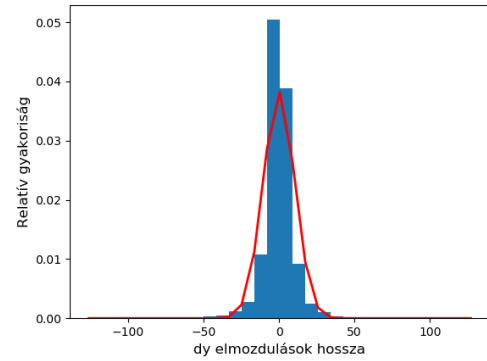
(c) A *bezier* adathalmaz dx elmozdulásainak hisztogramja



(d) A *bezier* adathalmaz dy elmozdulásainak hisztogramja



(e) A *humanLike* adathalmaz dx elmozdulásainak hisztogramja



(f) A *humanLike* adathalmaz dy elmozdulásainak hisztogramja

3.6. ábra. dx, dy hisztogramok különböző adathalmazok esetében

4. fejezet

Mérési eredmények

4.1. Mérési protokoll

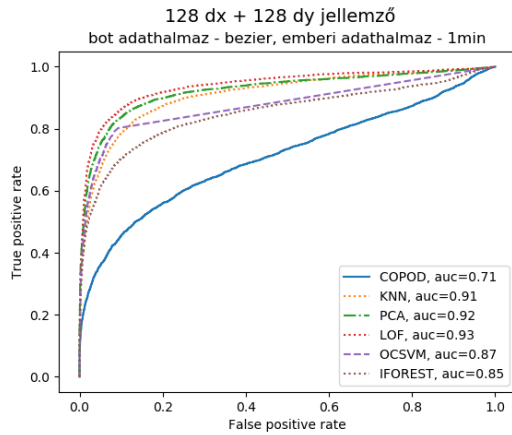
Anomália detektálás során az első lépésben a detektorok megtanulják a normális viselkedést, vagyis azt, hogy az emberre milyen egérmozgás jellemző. Ezt a tudást használják fel a következő lépésben arra, hogy különböző pontszámokat társítsanak a kapott mintákhoz. Mivel az emberi egérmozgást több megközelítés alapján is lehet utánozni és nehéz feltérképezni az összes ilyen módszert, a tanítási adathalmazom kizárólag emberi egérmozgásokat tartalmaz. A modellek kiértékelését (tesztelését) emberi és szintetikus mozgásokból előállított adathalmazokkal is végrehajtottam.

A tanításhoz minden esetben a 3.1 fejezetben említett 3 perces munkamenetekből álló SapiMouse adathalmazt használtam fel. A kiértékelés során az 1 perces munkamenetekből előállított adathalmazt használtam, mint humán adathalmaz, valamint az általam generált 4 adathalmazt felváltva, mint szintetikus adathalmaz. A méréseket kétszer végeztem el, először a nyers jellemzőkre (*rawfeatures*: (128*dx*, 128*dy*)), majd a 3.6 fejezetben említett 29 jellemzővel (*29features*).

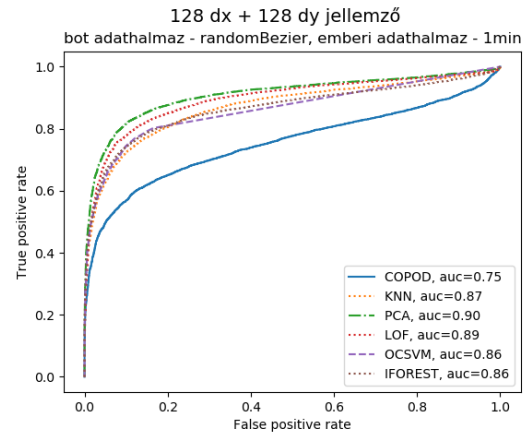
A tanítási adathalmaz (3*min*) 18308 mintát tartalmaz, a teszteléshez használt adathalmazok pedig 5919 mintát. A teszteléshez minden egyes mérés esetében a SapiMouse 1 perces adathalmaz mintáit használtam (5919 minta), mint emberi minták, és valamely szintetikus adathalmaz mintáit (5919 minta). Minden mérés eredménye 5919 pozitív és ugyanannyi negatív score érték lesz, amelyekből kiszámítjuk a ROC AUC értéket (lásd 4.2 alfejezet).

4.2. Metrikák

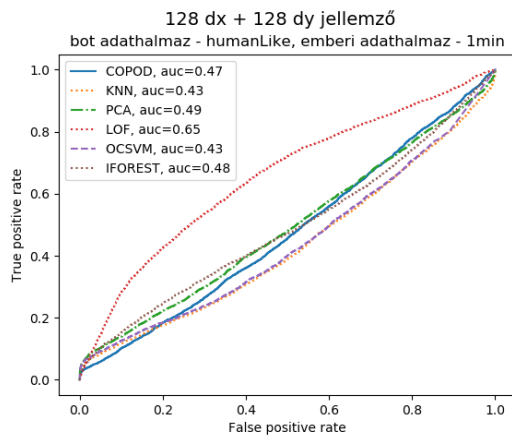
Az anomália detektorok teljesítményének kiértékelésénél a ROC (Receiver Operating Characteristics) görbét és az AUC (Area under the ROC curve) metrikát alkalmaztam [8]. A detektorok a rendelkezésünkre álló mintákhoz társítanak egy-egy pontszámot (score értéket), mely a pozitív osztályhoz (esetemben ez az emberi egérmozgás) tartozás mértékét mutatja. A negatív osztályhoz tartozó minták a szintetikus adathalmazok elemei. Definálunk különböző küszöbértékeket (thresholds) és minden küszöbérték esetében számítunk egy FPR (helytelenül osztályozott negatívak/összes negatívak száma - $FP/(FP+TN)$) és TPR (helytelenül osztályozott pozitívak/összes pozitívak száma - $TP/(TP+FN)$) értéket. Minden küszöbérték esetén a vele megegyező vagy nála nagyobb pontszámmal rendelkező



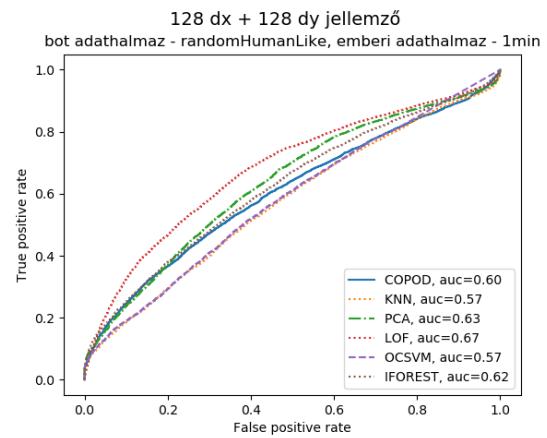
(a) *rawfeatures* jellemző alapú *bezier* adathalmazzal történő detektálás



(b) *rawfeatures* jellemző alapú *randomBezier* adathalmazzal történő detektálás



(c) *rawfeatures* jellemző alapú *humanLike* adathalmazzal történő detektálás



(d) *rawfeatures* jellemző alapú *randomHumanLike* adathalmazzal történő detektálás

4.1. ábra. ROC görbék 1 trajektória utáni döntésekkel *rawfeatures* típusú adathalmazok alapján

minták pozitívnak, a nála kisebb pontszámmal rendelkezők negatívnak minősülnek. Ennek alapján minden küszöbértéknek megfelel egy pont a ROC térben. Ezeket az értékeket ábrázolva kapjuk meg a ROC görbét [8]. Ilyen görbék láthatók a 4.1, 4.2, 4.3, 4.4 ábrákon.

Az AUC a ROC görbe alatti területet jelenti és azt fejezi ki, hogy mennyire soroljuk a pozitív mintákat a negatívak elé a tesztelés folyamán. Az AUC 0 és 1 közötti értékeket vehet fel. Minél közelebb van ez az érték az 1-hez, annál jobban teljesít a modell. Ha eléri a 1-et, akkor maximálisan teljesített, azaz maximális a szenzitivitása (sensitivity) és a specificitása (specificity). Maximális szenzitivitás esetén $FN=0$, azaz egyetlen pozitívát sem osztályozott negatívnak, maximális specificitás esetében pedig $FP=0$, azaz egyetlen negatívát sem osztályoz pozitívnak.

4.3. Eredmények

Amint a 4.1 fejezetben említettem, kétféle jellemzőhalmazzal dolgoztam:

- *rawfeatures*: 256 jellemző (128 dx és 128 dy).
- *29features*: 29 jellemző (lásd 3.6 fejezet)

Detector	<i>bezier</i>	<i>randomBezier</i>	<i>humanLike</i>	<i>randomHumanLike</i>
COPOD	0.71	0.75	0.47	0.60
KNN	0.91	0.87	0.43	0.57
PCA	0.92	0.90	0.49	0.63
LOF	0.93	0.89	0.65	0.67
OCSVM	0.87	0.86	0.43	0.57
IForest	0.85	0.86	0.48	0.62

4.1. táblázat. Detektorok összehasonlítása 1 trajektória utáni döntésekkel AUC metrika alapján *rawfeatures* típusú adathalmazok esetében

A detektorok 1 majd 10 trajektória után hozzák meg a döntést. Ennek azért van jelentősége, mert megfigyelhető, hogy a kötegek (jelen esetben 10 trajektória jelent egy köteget) alapján történő osztályozással jobb eredmény érhető el. Minden köteg esetében a pontszámot úgy határozzuk meg, hogy vesszük a köteghez tartozó trajektóriák pontszámát és átlagoljuk. Természetesen a kötegekhez tartozó trajektóriák azonos címkéjű minták.

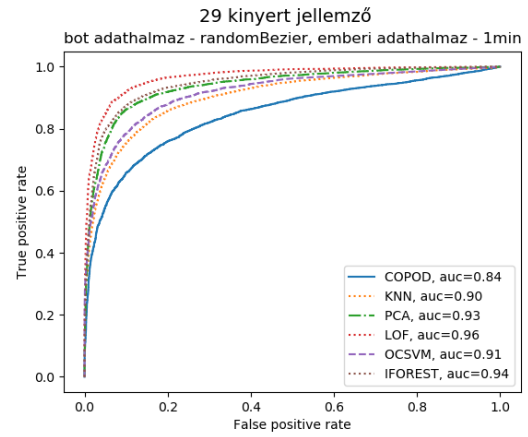
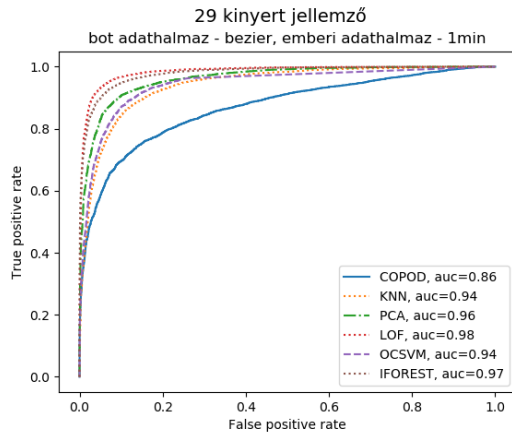
4.3.1. Egyetlen trajektória alapú eredmények

Rawfeatures alapú eredmények

Amint a 4.1 táblázatban és a 4.1 ábrán látható, a *rawfeatures* adathalmazok esetében a detektorok főként a *bezier* (a) és *randomBezier* (b) adathalmazok esetében teljesítenek jól. A *humanLike* (c) adathalmaz esetében, mely a legerősebb adathalmaz, az esetek többségében 0.5 alatti AUC értéket produkálnak. Ez gyengébb teljesítményt jelent, mint egy random osztályozás. A *randomHumanLike* adathalmaz esetében 0.6 körüli az AUC érték, ami nagyon gyengének minősül. Ez azt jelenti, hogy a detektorok emberinek minősítik ezeket a szintetikus mintákat.

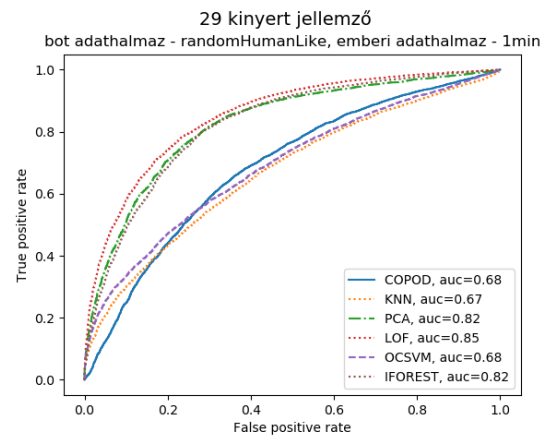
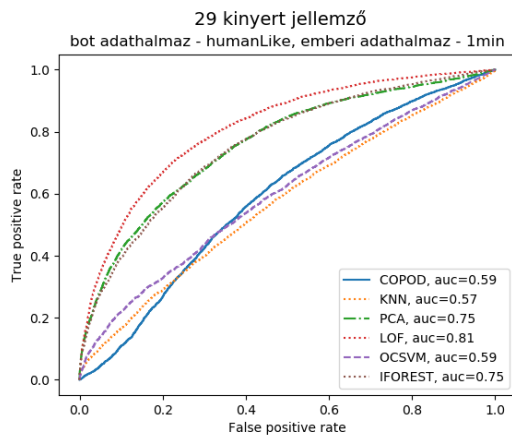
A *29features* alapján készült mérési eredmények

A *29features* típusú adathalmazok jellemzőkkel írják le a trajektóriákat (lásd 3.6 fejezet). Ez a megközelítés jelentős javulást hozott a detektorok teljesítményét illetően. Amint azt már láttuk a 4.1 ábrán, a *bezier* és *randomBezier* szintetikus adathalmazok elemei az idősorok alapján is elkülöníthetők voltak, a 4.2 ábrán az látható, hogy a jellemzőknek köszönhetően a görbék jobban közelítik az ábra bal felső csúcsát, azaz a (0,1) pontot a másik két adathalmaz esetében is. Ez azt jelenti, hogy az FPR csökken, tehát csökken a pozitívnak mondott negatívak száma, míg a TPR nő, vagyis növekszik a pozitívnak mondott pozitívak száma. A 4.2 táblázatból leolvasható, hogy a *humanLike* adathalmaz esetében a detektoroknak megfelelő AUC értékek meghaladják a 0.59-et, illetve az LOF kiemelkedően teljesít, 0.81-es AUC értékkel. A *randomHumanLike* adathalmaz esetében 3 detektor is meghaladja a 0.8-as AUC értéket, ami már jó értéknek számít.



(a) 29features jellemző alapú *bezier* adathalmazzal történő detektálás

(b) 29features jellemző alapú *randomBezier* adathalmazzal történő detektálás



(c) 29features jellemző alapú *humanLike* adathalmazzal történő detektálás

(d) 29features jellemző alapú *randomHumanLike* adathalmazzal történő detektálás

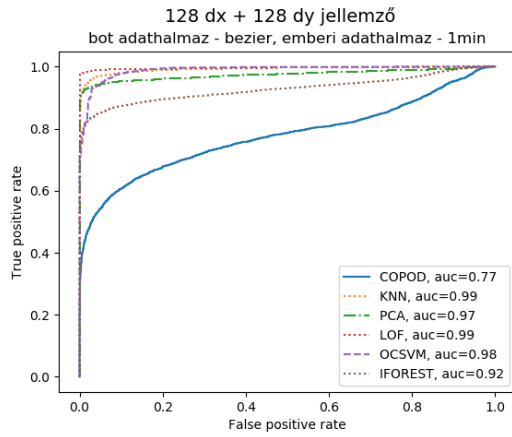
4.2. ábra. ROC görbék 1 trajektória utáni döntésekkel 29features típusú adathalmazok alapján

Detector	<i>bezier</i>	<i>randomBezier</i>	<i>humanLike</i>	<i>randomHumanLike</i>
COPOD	0.86	0.84	0.59	0.68
KNN	0.94	0.90	0.57	0.67
PCA	0.96	0.93	0.75	0.82
LOF	0.98	0.96	0.81	0.85
OCSVM	0.94	0.91	0.59	0.68
IForest	0.97	0.94	0.75	0.82

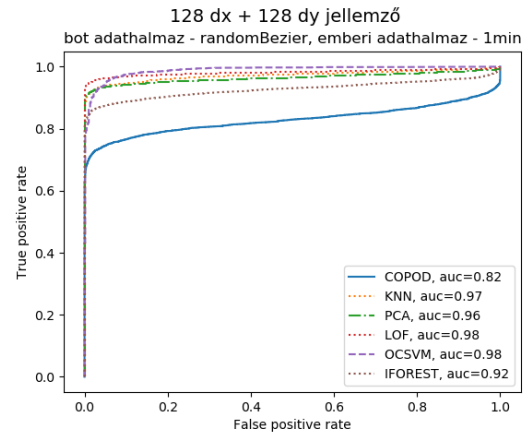
4.2. táblázat. Detektorok összehasonlítása 1 trajektória utáni döntésekkel AUC metrika alapján 29features típusú adathalmazok esetében

x és *y* irányú eredmények

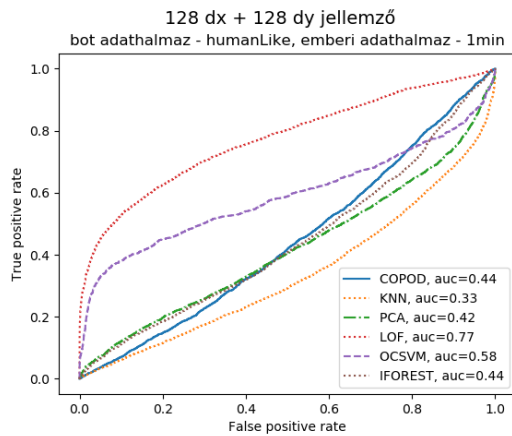
Mind a 4 szintetikus adathalmaz esetében külön választottam a dx és dy értékeket, annak érdekében, hogy összehasonlítsam a detektorok teljesítményét a két irány mentén.



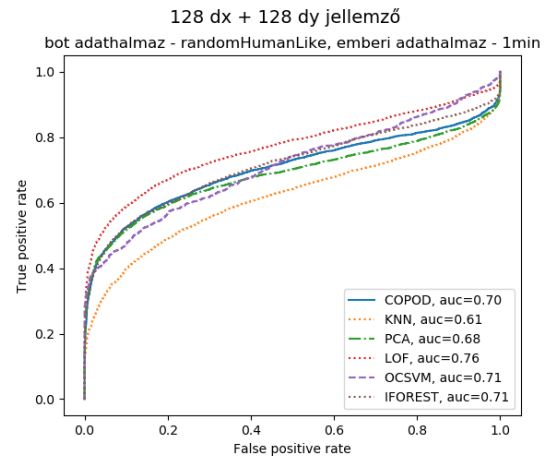
(a) *rawfeatures* jellemző alapú *bezier* adathalmazzal történő detektálás



(b) *rawfeatures* jellemző alapú *randomBezier* adathalmazzal történő detektálás



(c) *rawfeatures* jellemző alapú *humanLike* adathalmazzal történő detektálás



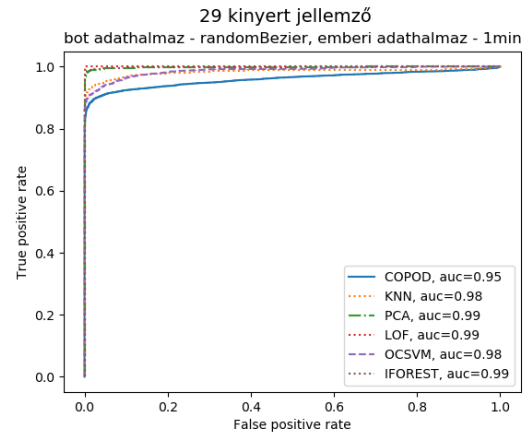
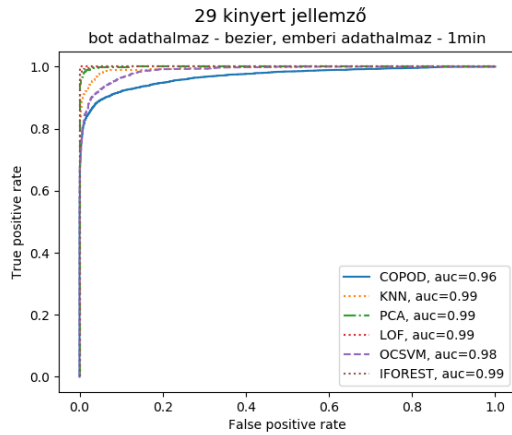
(d) *rawfeatures* jellemző alapú *randomHumanLike* adathalmazzal történő detektálás

4.3. ábra. ROC görbék 10 trajektória utáni döntésekkel *rawfeatures* típusú adathalmazok alapján

Dataset	dx	dy	$dx + dy$
<i>bezier</i>	0.79 (KNN, OCSVM)	0.93 (KNN, OCSVM)	0.93 (LOF)
<i>randomBezier</i>	0.84 (PCA), 0.8 (OCSVM)	0.88 (PCA, OCSVM)	0.9 (PCA)
<i>humanLike</i>	0.56 (LOF)	0.55 (LOF)	0.65 (LOF)
<i>randomHumanLike</i>	0.56 (IForest)	0.62 (IForest)	0.67 (LOF)

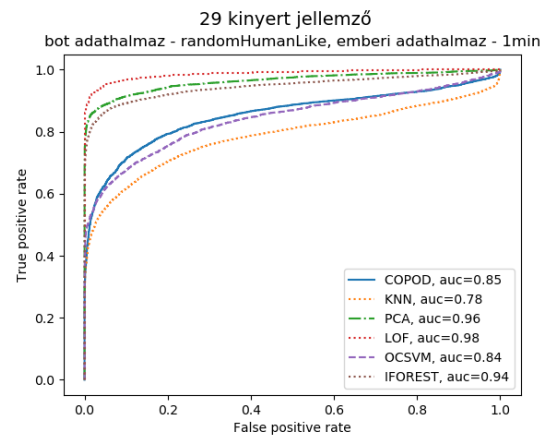
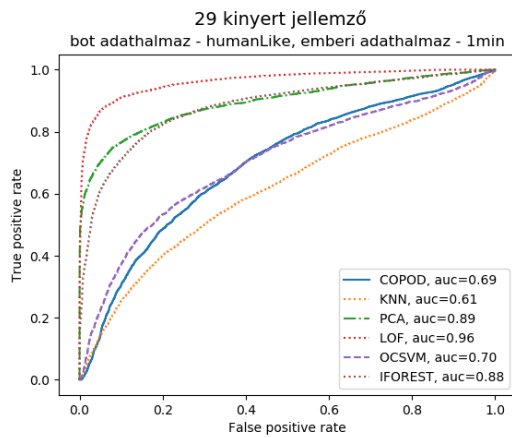
4.3. táblázat. AUC értékek összehasonlítása 1 trajektória utáni döntésekkel *rawfeatures* típusú adathalmazok esetében

A *rawfeatures* adathalmazok esetében azt tapasztaltam, hogy az y irányú elmozdulásokat tartalmazó adathalmazok esetében átlagosan 5 százalékkal jobban teljesítenek a detektorok, mint az x irányú elmozdulásokat tartalmazók esetében. Ez azt jelenti tehát, hogy az y irány diszkriminatívabb az x iránynál az egérmozgások elkülönítésének tekintetében,



(a) 29features jellemző alapú *bezier* adathalmazzal történő detektálás

(b) 29features jellemző alapú *randomBezier* adathalmazzal történő detektálás



(c) 29features jellemző alapú *humanLike* adathalmazzal történő detektálás

(d) 29features jellemző alapú *randomHumanLike* adathalmazzal történő detektálás

4.4. ábra. ROC görbék 10 trajektória utáni döntésekkel 29features típusú adathalmazok alapján

viszont nem tekinthető jobb megközelítésnek, mint a mindkét irányt tartalmazó változat. Habár egyes detektorok jobban teljesítenek 1-2 százalékkal, ha mindössze y irányú elmozdulások alapján mérnek, érdemesebb a mindkét irányt tartalmazó adathalmazokkal dolgozni, mivel az összes detektort figyelembe véve ez a megközelítés adja a legmagasabb AUC értéket. Ez látható a 4.3 táblázatban, ahol mind a 4 adathalmaz esetében fel van tüntetve a legmagasabb AUC érték és az eredményt produkáló detektor.

4.3.2. 10 trajektória alapú eredmények

A 10 trajektória alapján hozott döntések eredményei egy körülbelül 10 százalékos javulást mutatnak az 1 trajektóriás eredményekhez képest. Ez már elegendő ahhoz, hogy mind a 4 adathalmaz esetében kiválónak tekinthessük az osztályozást.

Detector	<i>bezier</i>	<i>randomBezier</i>	<i>humanLike</i>	<i>randomHumanLike</i>
COPOD	0.77	0.82	0.44	0.70
KNN	0.99	0.97	0.33	0.61
PCA	0.97	0.96	0.42	0.68
LOF	0.99	0.98	0.77	0.76
OCSVM	0.98	0.98	0.58	0.71
IForest	0.92	0.92	0.44	0.71

4.4. táblázat. Detektorok összehasonlítása 10 trajektória utáni döntésekkel AUC metrika alapján *rawfeatures* típusú adathalmazok esetében

***Rawfeatures* adathalmaz alapú eredmények**

A 4.3 ábra kapcsolatba hozható a 4.1 ábrával, mivel szintén *rawfeatures* típusú adathalmazzal történt a mérés. A kettő közti különbség abban áll, hogy míg az előbbi esetében a detektorok 1 trajektória után hozták meg a döntést, a 4.3 ábrán a kiértékelésnél 10 egymást követő trajektóriát vettem figyelembe. Látható, hogy a *rawfeatures* típusú adathalmazok esetében ez a fajta mérési protokoll is csak az (a) és (b) ábrák esetében elfogadható, az emberszerű egérmozgásokat (c és d ábrák) továbbra is kevés sikerrel különböztetik meg a detektorok. Ezt foglalja össze a 4.4 táblázat is.

A *29features* adathalmaz alapján készült mérési eredmények

A 4.4 ábra kapcsolatba hozható a 4.2 ábrával, mivel szintén *29features* típusú adathalmazzal történt a mérés, ezúttal azonban a detektorok 10 egymást követő trajektória után hozták meg a döntést. Az (a) és (b) ábrákon szinte maximális teljesítmények láthatók, azonban a (c) és (d) ábrákon is megjelennek 1-hez közeli értékek. Egyértelmű tehát, hogy a legjobb eredményeket minden adathalmaz esetében a jellemzők alapján érjük el 10 trajektória utáni döntésekkel. A 4.5 táblázatban alapján összehasonlítva a detektorokat, elmondható, hogy az LOF, PCA és IForest minden adathalmaz esetében kiemelkedő AUC értékeket szolgáltat.

Detektor	<i>bezier</i>	<i>randomBezier</i>	<i>humanLike</i>	<i>randomHumanLike</i>
COPOD	0.96	0.95	0.69	0.85
KNN	0.99	0.98	0.61	0.78
PCA	0.99	0.99	0.89	0.96
LOF	0.99	0.99	0.96	0.98
OCSVM	0.98	0.98	0.70	0.84
IForest	0.99	0.99	0.88	0.94

4.5. táblázat. Detektorok összehasonlítása 10 trajektória utáni döntésekkel AUC metrika alapján *29features* esetében

4.3.3. A *randomTimeGan* adathalmazzal végzett mérések eredményei

Mivel a korábbi adathalmazok alapján és a *randomTimeGan* adathalmazzal való mérések alapján is a legjobban teljesítő detektor az LOF, a 4.6-es táblázatban ennek a detektornak a teljesítményét tüntettem fel különböző szintetikus adathalmazok (a *randomTimeGan* adathalmaz változatai) alapján történő méréseknek megfelelően. Az adathalmazok közötti különbség az őket generáló neuronháló rejtett réteg-beli dimenziójának a változtatásából adódik. Ezt a paramétert változtatva megfigyelhető, hogy változik az adathalmaz minősége. A legjobb paraméter a 28-as lett, mivel az AUC érték itt éri el a minimumot. Ennek az értéknek megfelelő *randomTimeGan* adathalmaz alapján kapott AUC értékeket tartalmazza a 4.7 táblázat mind az 5 detektor esetében, valamint a 4.5 ábra szemlélteti a ROC görbéket. A *randomTimeGan 29features* típusú változatával dolgoztam minden esetben.

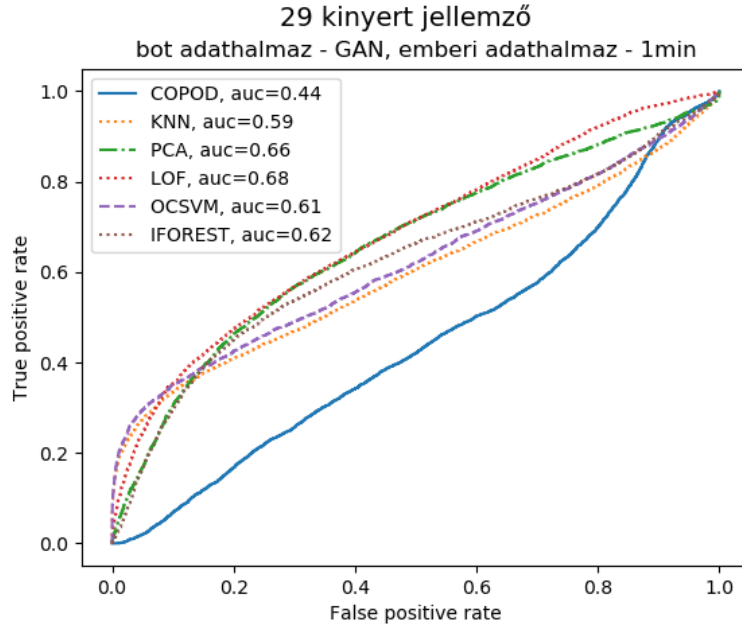
Rejtett réteg dimenziója	AUC
8	0.99
16	0.84
24	0.75
28	0.68
32	0.80
48	0.73
64	0.72

4.6. táblázat. LOF detektorok teljesítménye AUC metrika szerint a rejtett rétegek dimenziójának változása függvényében

Detektor	AUC
COPOD	0.44
KNN	0.59
PCA	0.66
LOF	0.68
OCSVM	0.61
IForest	0.62

4.7. táblázat. Detektorok teljesítménye 1 trajektória utáni döntésekkel *randomTimeGan* adathalmaz alapján AUC metrika szerint

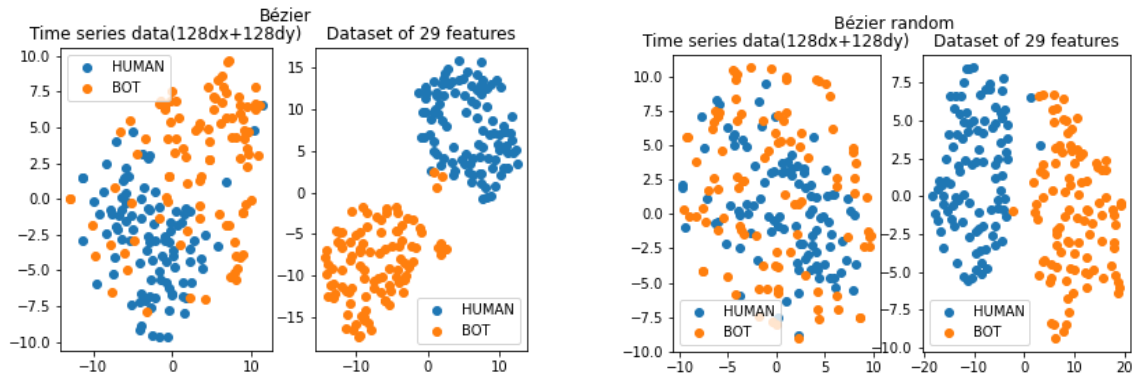
A korábbi 4.2 táblázatban szemléltettem 4 különböző *29features* típusú adathalmazra kapott AUC értékeket. Ezeket összehasonlítva a 4.7 táblázatban szereplő értékekkel, észrevehető, hogy az utóbbi esetében alacsonyabbak az AUC értékek, így a *randomTimeGan* tekinthető a legerősebb szintetikus adathalmaznak.



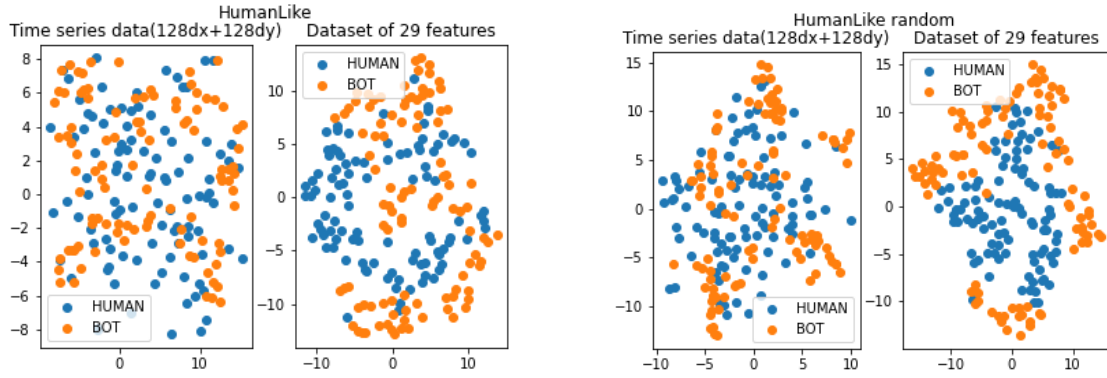
4.5. ábra. *29features* jellemző alapú *randomTimeGan* adathalmazzal történő detektálás

4.4. Vizualizáció

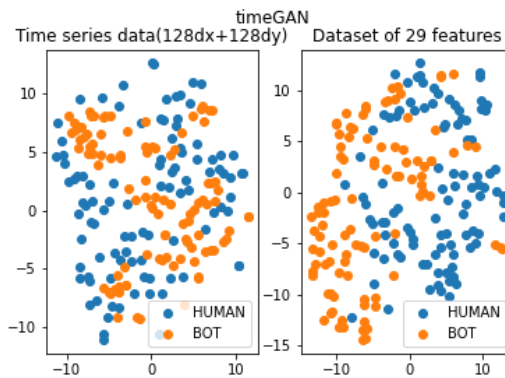
A t-SNE (t-distributed stochastic neighbourhood embedding) módszer nagy dimenziójú adatok megjelenítését teszi lehetővé. Ezt úgy valósítja meg, hogy levetíti az adatpontokat egy alacsonyabb dimenziójú térbe (ez lehet 2D vagy 3D), megőrizve azok eredeti klasztereződését. A 4.6 ábra szemlélteti az adathalmazok vizualizációját 2 dimenziós térben. Az (a), (b), (c), (d) és (e) ábrák mindegyikén a bal oldalon a *rawfeatures* adathalmazok, a jobb oldalon a *29features* adathalmazok láthatók. Megfigyelhető, hogy a *rawfeatures* ábrák esetében a két osztályhoz tartozó pontok nagyobb mértékben keverednek, míg a *29features* adathalmazok esetében a klasztereződés a jellemző. A különbség az adathalmazok reprezentációjából adódik. A klasztereződés mértéke jelzi az adathalmaz erősségét, melyre a detektorok teljesítményéből is következtethetünk. A *bezier* és *randomBezier* adathalmazoknál szinte teljesen elhatárolódnak egymástól a két különböző osztályhoz tartozó adatpontok, míg a *humanLike*, *randomHumanLike* és *randomTimeGAN* esetében nehezebb egy határt képezni az adatpontok elkülönítésére. Továbbá a (c) ábra is alátámasztja, hogy a Bézier alapú legerősebb adathalmaz a *humanLike*.



(a) *bezier* adathalmaz *rawfeatures* (balról) (b) *randomBezier* adathalmaz *rawfeatures* (balról) *29features* (jobbról)



(c) *humanLike* adathalmaz *rawfeatures* (balról) (d) *randomHumanLike* adathalmaz *rawfeatures* (balról) *29features* (jobbról)



(e) *randomTimeGAN* adathalmaz *rawfeatures* (balról) *29features* (jobbról)

4.6. ábra. Adathalmazok vizualizálása a t-SNE módszer használatával

5. fejezet

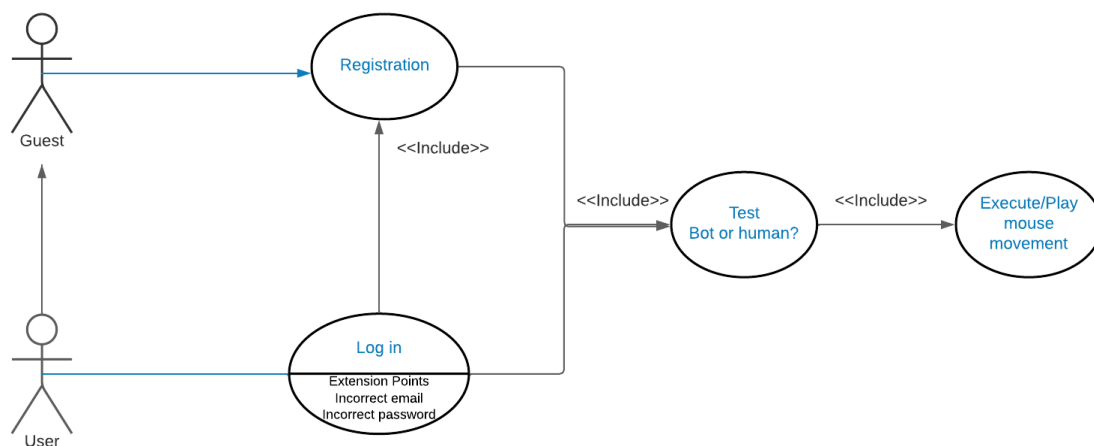
Bot detektáló szoftver

5.1. A szoftver áttekintése

Az általam elkészített szoftver egy webalkalmazás komponensnek tekinthető. Célja egy adott oldal felhasználóinak regisztrációját, illetve bejelentkezését megvalósítani, ezzel egyidejűleg pedig kiszűrni a webbotokat annak érdekében, hogy csak az emberek számára legyen elérhető az adott oldal. Ennek megvalósítása érdekében a rendszer minden felhasználótól begyűjt egy egérmozgást, majd ezt felhasználva a korábbi fejezetekben bemutatott anomália detektálási folyamatot hajtja végre. Ennek nyomán megállapítja a felhasználóról, hogy ember vagy bot.

5.1.1. Felhasználói követelmények

A szoftver funkciói a következők: teszt(mely során eldől a felhasználóról, hogy ember vagy bot), regisztráció és bejelentkezés. A teszt során a felhasználónak egy egérmozgást kell végrehajtania a képernyő egy random pontjából egy megadott célpontig (5.1. ábra), továbbá lehetőség van egy generált egérmozgás lejátszására.



5.1. ábra. Use-case diagram.

5.1.2. Rendszerkövetelmények

a) Funkcionális követelmények.

A szoftver komponens elsősorban megpróbálja megállapítani az oldalra ellátogatott felhasználóról, hogy ember vagy bot. Ennek érdekében a felhasználók a képernyő középső részén elhelyezett nyomógombon egy bal kattintást kell végrehajtsanak. Ehhez a szoftver első lépésben a felhasználó egerét a képernyő jobb vagy bal szélére pozicionálja, így a felhasználónak ebből a pontból kell mozgatnia a kurzort a képernyő közepén elhelyezkedő nyomógomb irányába. Ez lehetővé teszi a rendszer számára, hogy egy hosszabb egérmozgás koordinátáit naplózza. Amennyiben a felhasználó sikeresen teljesítette a tesztet, a szoftver komponens további regisztrációs lehetőséget biztosít. Érvényes email cím és egy jelszó megadása, majd megerősítése esetén a felhasználók sikeresen be tudnak regisztrálni, majd a későbbiekben a helyes email cím és jelszó páros megadásával bejelentkezni. Az alkalmazás minden hibát jelez a felhasználóknak (pl. nincs megadva a szükséges inputok valamelyike vagy helytelenül van megadva).

b) Nem funkcionális követelmények

- Személyi számítógép/laptop
- Internet hálózat: H+, 3G, 4G, 5G
- GitHub verziókezelő
- Visual Studio Code - Az alkalmazás megvalósítása Django keretrendszerben történt: python, html, css és javascript nyelvek használatával
- Firebase - jelszó hashing

Az alkalmazás jelenleg lokális hálózaton fut, azonban a regisztrációhoz, illetve a bejelentkezéshez szükséges internetkapcsolat. A rendszer a felhasználóról nem ment le további információkat, kizárólag az egérmozgás során begyűjtött adatokat.

5.2. Tervezés

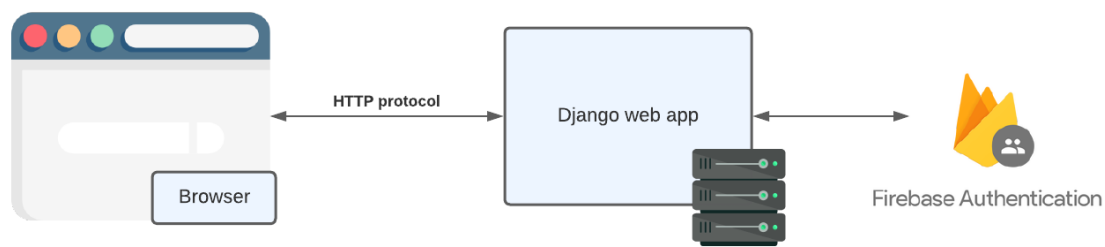
A szoftver célja a korábbi fejezetekben bemutatott kutatás szemléltetése, illetve gyakorlati felhasználhatóságának bemutatása. A tervezési fázis tartalmazta azt a lépést, amikor megterveztem egy prototípust, illetve leszögeztem a rendszer főbb funkcionálisait.

A szoftver legfontosabb részét a felhasználók egérmozgásainak begyűjtése képezi. Amint azt a 4.3 fejezetben is szemléltettem, egyetlen egérmozgás alapján jóval nehezebb eldönteni egy adott felhasználóról, hogy ember vagy bot, viszont a rendszer fontos szempontja volt, hogy minél egyszerűbb feladat végrehajtásával terhelje az oldalra ellátogatott személyt.

Mivel kutatásom során a Python programozási nyelvet használtam, illetve az ehhez tartozó könyvtárakat, egyértelmű volt, hogy egy Python alapú keretrendszerben fogom megvalósítani a szoftvert.

Az alkalmazás architektúráját a 5.2. diagram szemlélteti.

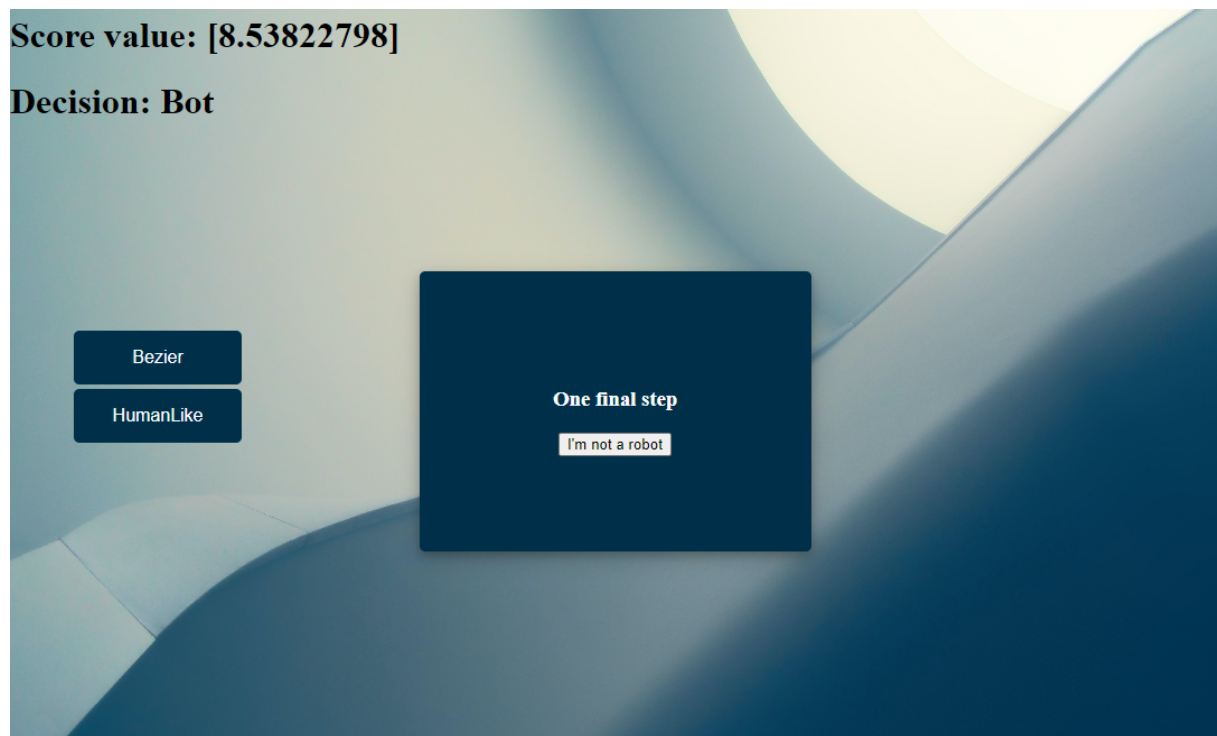
Amint az az architektúra diagramról is leolvasható, a felhasználók hitelesítését Firebase Authenticator segítségével valósítottam meg, az Email/Password opció segítségével.



5.2. ábra. Architektúra diagram.

5.3. Kivitelezés

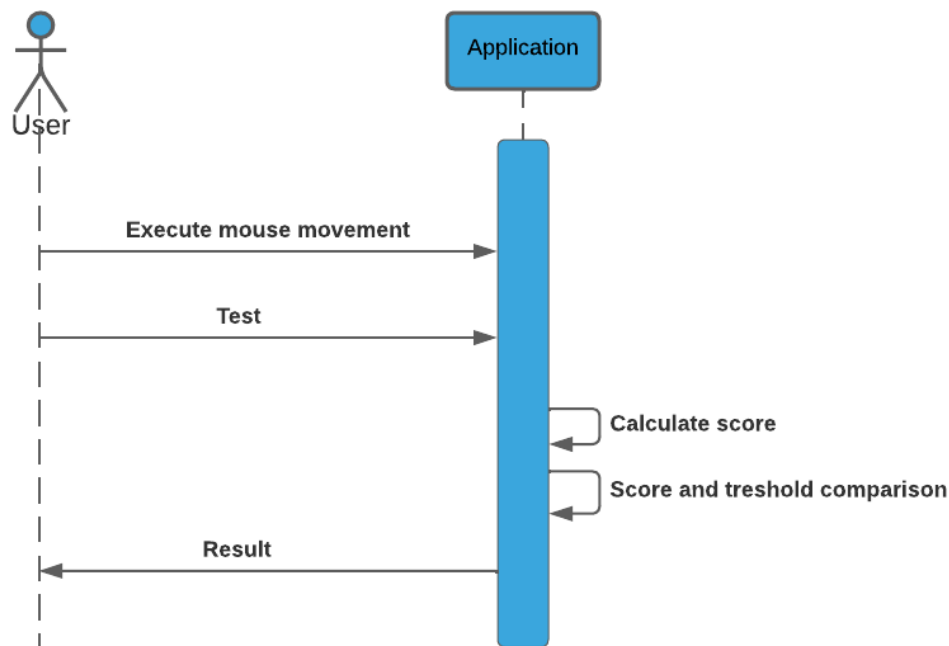
A weboldal indításakor a webes támadások kivédését szolgáló oldallal találja magát szemben a felhasználó (5.3. ábra). Az oldal betöltésekor a kurzor mindig a képernyő egy random pontjára helyeződik, azonban fontos, hogy ez mindig az adott felbontású képernyő peremén lévő területen helyezkedik el. Ebből a pontból a felhasználó a 5.3. ábrán látható *I'm not a robot* gomb felületéhez kell mozgassa a kurzort, eközben a szoftver végig naplózza a felhasználó egérmozgását. A gombhoz érve a felhasználó egy bal kattintást kell végrehajtson annak érdekében, hogy a teszt sikeres teljesítését követően tovább mehessen a bejelentkezés oldalra.



5.3. ábra. Detektálás oldal.

Tesztelés során az előre betanított modellt(LOF) használja a rendszer. A tanítást kezdetben a SapiMouse adathalmaz 3 perces munkamenetei alapján előállított 29featu-

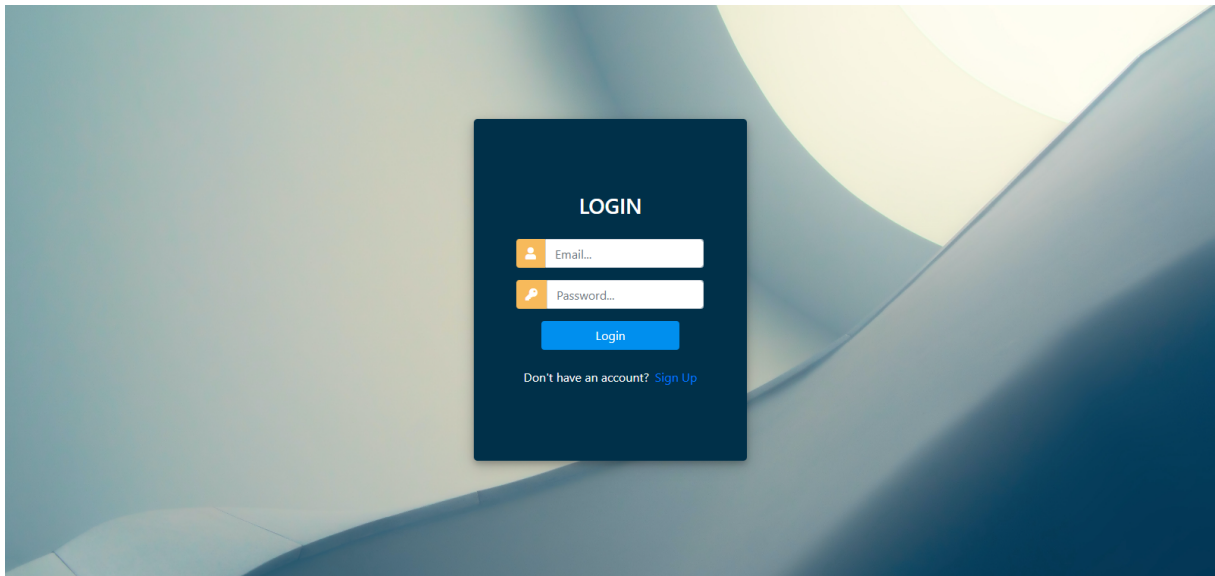
res típusú adathalmaz segítségével végeztem, azonban azt tapasztaltam, hogy ebben az esetben nehéz meghatározni egy megfelelő küszöbértéket az ember által végrehajtott, illetve generált egérmozgások elkülönítésére. Ennek következtében a felületről gyűjtöttem 4 személytől megközelítőleg 500 egérmozgást, majd ezek alapján előállítottam egy újabb *29features* típusú tanító adathalmazt. Ebben az esetben már meg tudtam határozni empirikusan egy küszöbértéket.



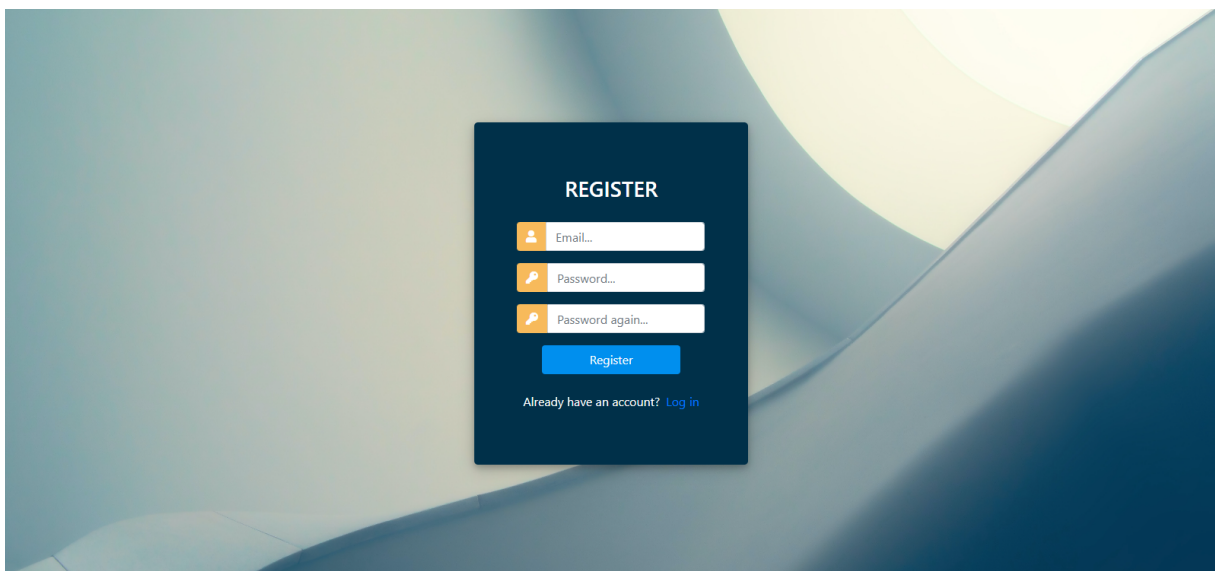
5.4. ábra. Teszt - szekvencia diagram.

A kezdőoldal továbbá tartalmaz két gombot (Bezier, HumanLike), melyekre kattintva egy-egy generált egérmozgást játszik le a szoftver egy szintén random pont és az *I'm not a robot* gomb felülete között, majd az emberhez hasonlóan egy bal kattintást hajt végre a gomb felületén. Ebben az esetben a teszt sikertelennek bizonyul az esetek többségében, hiszen a rendszer detektálja a mozgás generált jellegét.

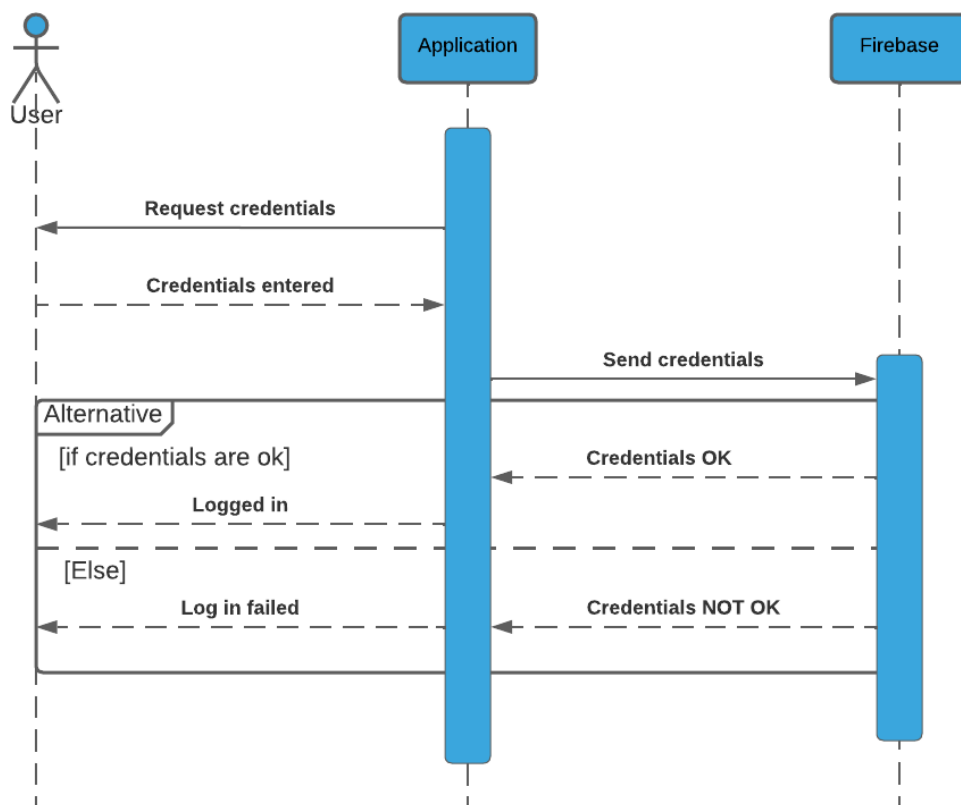
Amennyiben az egérmozgás alapján a felhasználó embernek minősül, a bejelentkezést (5.5. ábra) megvalósító ablak köszönti, innen lehetősége van átnavigálni a regisztrációs oldalra (5.6. ábra).



5.5. ábra. Bejelentkezés oldal.



5.6. ábra. Regisztrációs oldal.



5.7. ábra. Bejelentkezés - szekvencia diagram.

6. fejezet

Összefoglaló

Dolgozatomban különböző stratégiák alapján generáltam emberszerű egérmozgásokat, majd ezek emberszerűségét vizsgáltam különböző anomália-detektorok segítségével.

A szintetikus adatok generálása módosított Bézier görbék, illetve a TimeGAN neuronháló segítségével történt. A generált egérmozgások idősoroknak tekinthetők, melyek esetemben 129 pontpárból állnak (nyers adathalmaz). Ezeknek egy kisebb dimenziójú reprezentációját állítottam elő jellemzőkinyeréssel, így összesen 29 jellemzővel dolgoztam.

A valós emberi egérmozgásokat a Sapimouse adathalmaz szolgáltatta. Ennek egy részét a detektorok tanítására használtam, majd egy kisebb részét, illetve a különböző generált adathalmazokat a tesztelésükre. A detektorok teljesítményét AUC metrika alapján mértem. Az adathalmazok 29 jellemzőből álló reprezentációja növelte a detektorok teljesítményét a nyers adathalmazokkal szemben. A méréseket végrehajtottam 10 egymást követő egérmozgás esetében is, ez szintén pozitívan befolyásolta a szintetikus adatok detektálhatóságát.

Az AUC metrika alapján az LOF detektor tudta a legsikeresebben megállapítani az egérmozgásokról, hogy ember vagy bot állhat mögöttük. Az LOF 10 trajektória utáni döntésekkel a Bézier görbéken alapuló adathalmazok esetében 0.96-nál magasabb értékeket vesz fel, illetve 1 trajektóriás eredmények esetében 0.81 fölötti értékeket produkál. A TimeGAN esetében egy trajektória alapján az LOF-nek megfelelő AUC érték 0.68. Ez azt jelenti, hogy a legerősebb adathalmaz a TimeGAN-al generálható, hiszen a detektorok teljesítménye ekkor a leggyengébb. A Bézier görbéken alapuló adathalmazok közül a legerősebb a *humanLike* adathalmaz, mely módosított harmadfokú Bézier-görbét tartalmaz.

Míg a TimeGAN esetében random végpontok között generálódnak a mozgások, a Bézier görbék esetében lehetőség van megadott végpontok közti generálásra. Megfigyelhető, hogy amennyiben a rendelkezésünkre álló valós egérmozgások végpontjai között generáljuk a mozgásokat, illetve figyelembe vesszük a mozgások köztes pontjainak számát, a detektorok nehezebben tudják megkülönböztetni az emberi és szintetikus mozgásokat.

Az egérmozgások x és y koordinátáit külön véve megállapítottam, hogy az y irányú elmozdulások alapján jobban el lehet különíteni az emberi és szintetikus egérmozgásokat, mint az x irányú mozgások alapján.

Végezetül írtam egy szemléltető szoftvert is, mely egy weboldal komponensként a felhasználók hitelesítését végzi, továbbá megpróbálja kiszűrni a webbotokat. Ezt egyetlen egérmozgás alapján végzi, illetve a *29features* típusú reprezentációval dolgozik.

Továbbfejlesztési lehetőségként megemlíteném az egérmozgások további tanulmányozását újabb jellemzők előállítása érdekében, melyek még jobbra tennék a detektorok teljesítményét, illetve a TimeGAN neuronháló adaptálásának továbbfejlesztését egérdinamikai idősorok előállítása érdekében.

A kutatási részt megvalósító szoftver forráskódja elérhető a GitHub-on a következő linken: <https://github.com/kandi00/Allamvizsga>, a bot detektáló szoftver forráskódja ezen a linken: <https://github.com/kandi00/Bot-detection-software>.

Ábrák jegyzéke

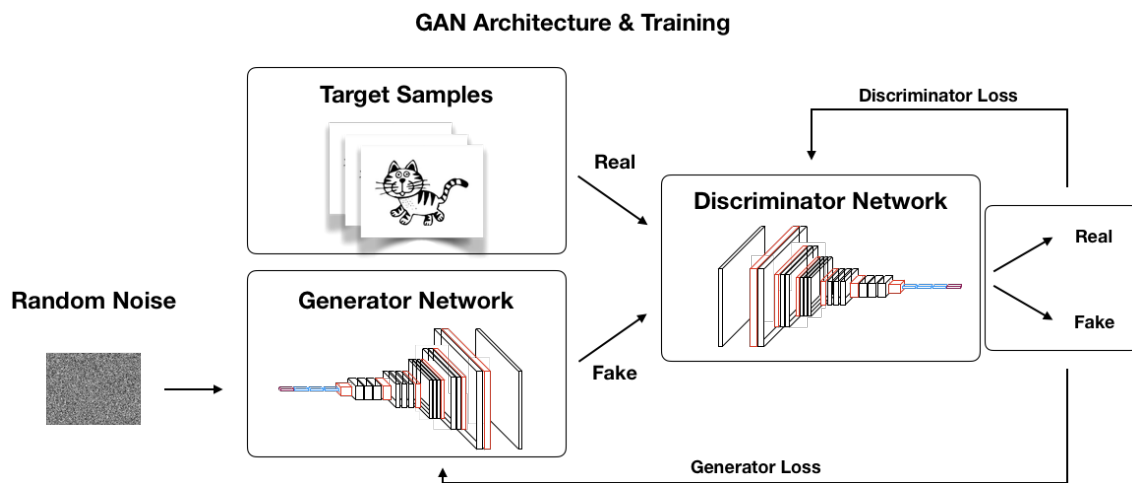
3.1. Egy emberi trajektória ábrázolása	17
3.2.	18
3.3. TimeGAN architektúra	20
3.4. TimeGAN-al generált egérmozgások	21
3.5. SapiMouse statisztikák	27
3.6. dx, dy hisztogramok különböző adathalmazok esetében	29
4.1. ROC görbék 1 trajektória utáni döntésekkel <i>rawfeatures</i> típusú adathalmazok alapján	31
4.2. ROC görbék 1 trajektória utáni döntésekkel <i>29features</i> típusú adathalmazok alapján	33
4.3. ROC görbék 10 trajektória utáni döntésekkel <i>rawfeatures</i> típusú adathalmazok alapján	34
4.4. ROC görbék 10 trajektória utáni döntésekkel <i>29features</i> típusú adathalmazok alapján	35
4.5. <i>29features</i> jellemző alapú <i>randomTimeGan</i> adathalmazzal történő detektálás	38
4.6. Adathalmazok vizualizálása a t-SNE módszer használatával	39
5.1. Use-case diagram.	40
5.2. Architektúra diagram.	42
5.3. Detektálás oldal.	42
5.4. Teszt - szekvencia diagram.	43
5.5. Bejelentkezés oldal.	44
5.6. Regisztrációs oldal.	44
5.7. Bejelentkezés - szekvencia diagram.	45
I.0.1. TimeGAN architektúra[2]	51
I.0.2. Autoencoder	51
I.0.3. Adversarial-supervised	52
I.0.4. Adversarial-embedding	52
I.0.5. Discriminator	53
I.0.6. Synthetic data	53

Irodalomjegyzék

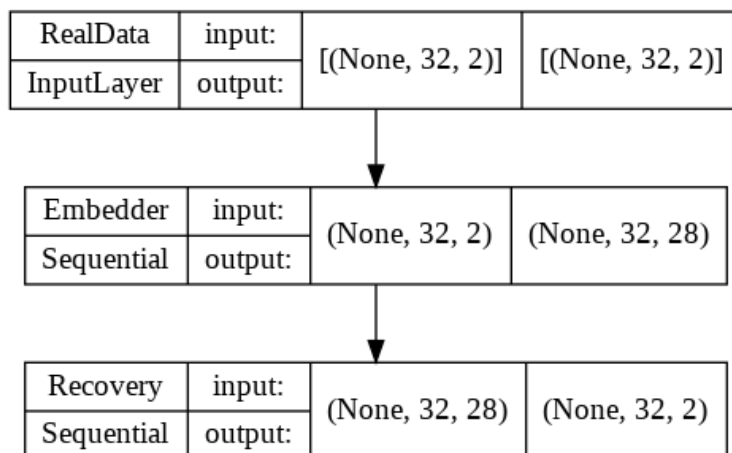
- [1] Bézier görbe - wikipédia. https://en.wikipedia.org/wiki/B%C3%A9zier_curve. (elérés dátuma: 2022. ápr. 11.).
- [2] Timegan architektúra. https://github.com/stefan-jansen/machine-learning-for-trading/tree/main/21_gans_for_synthetic_time_series. (elérés dátuma: 2022. ápr. 11.).
- [3] ACIEN, A., MORALES, A., FIERREZ, J., AND VERA-RODRIGUEZ, R. Bcaptcha-mouse: Synthetic mouse trajectories and improved bot detection. *Pattern Recognition* 127 (2022), 108643.
- [4] AGGARWAL, C. C. *Outlier Analysis*. Springer International Publishing, 2017.
- [5] ANTAL, M., BUZA, K., AND FEJER, N. Sapiagent: A bot based on deep learning to generate human-like mouse trajectories. *IEEEAccess* 9, 21099141 (2021), 124396 – 124408.
- [6] CHANDOLA, V., BANERJEE, A., AND KUMAR, V. Anomaly detection : A survey. *ACM Computing Survey* 41 (2009), 1557–7341.
- [7] CHU, Z., GIANVECCHIO, S., AND WANG, H. Bot or human? a behavior-based online bot detection system. In *From Database to Cyber Security* (2018).
- [8] FAWCETT, T. An introduction to roc analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.
- [9] GIANVECCHIO, S., WU, Z., XIE, M., AND WANG, H. Battle of botcraft: Fighting bots in online games with human observational proofs. *Proc. 16th ACM Conf. Comput. Commun. Secur* (2009), 256 – 268.
- [10] ILIOU, C., KOSTOULAS, T., TSIKRIKA, T., KATOS, C. V., VROCHIDIS, S., AND KOMPATSIARIS, I. Detection of advanced web bots by combining web logs with mouse behavioural biometrics. *ACM Journals* 2, 24 (2021), 1 – 26.
- [11] LI, Z., ZHAO, Y., BOTTA, N., IONESCU, C., AND HU, X. Copod: Copula-based outlier detection. *IEEEAccess*, 20424134 (2021).
- [12] YOON, J., JARRETT, D., AND VAN DER SCHAAR, M. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems* (2019), H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc.

- [13] ZHAO, Y., NASRULLAH, Z., AND LI, Z. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research* 20, 96 (2019), 1–7.

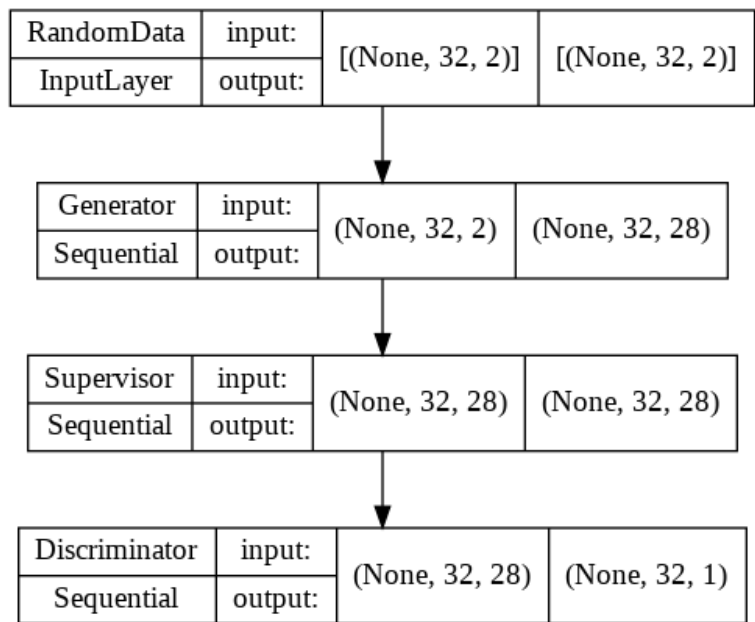
Függelék



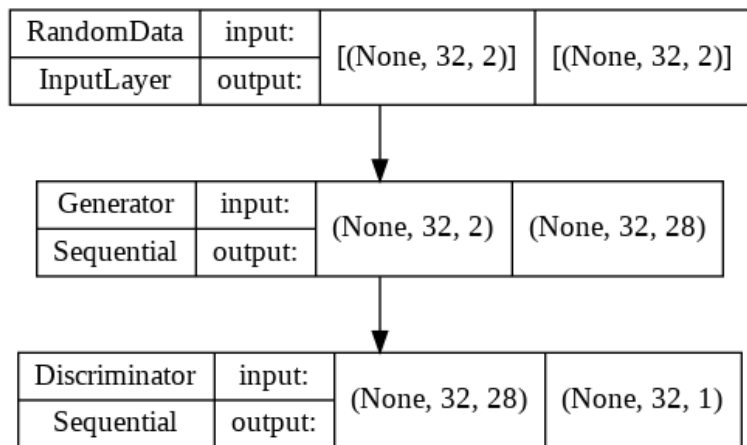
I.0.1. ábra. TimeGAN architektúra[2]



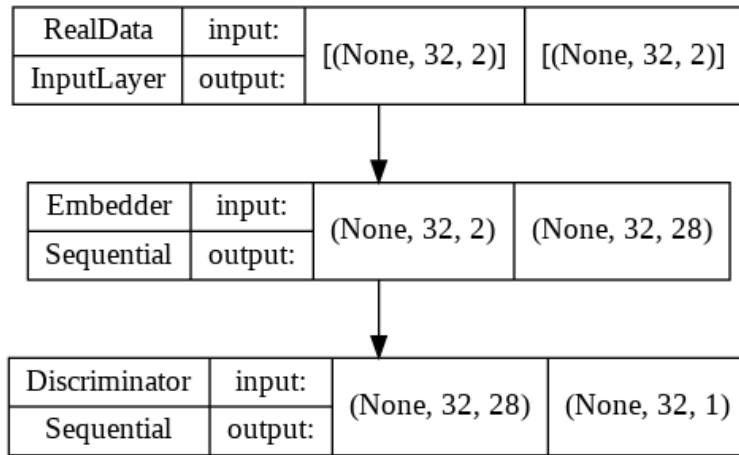
I.0.2. ábra. Autoencoder



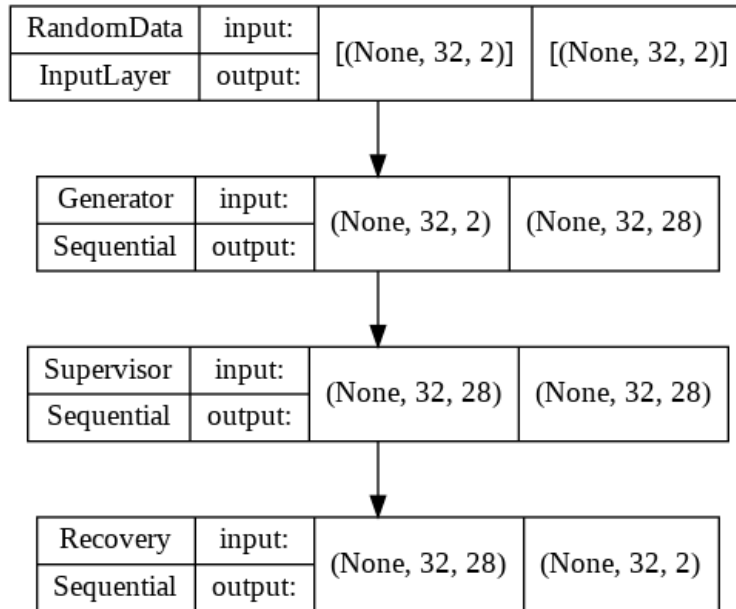
I.0.3. ábra. Adversarial-supervised



I.0.4. ábra. Adversarial-embedding



I.0.5. ábra. Discriminator



I.0.6. ábra. Synthetic data