Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Novák Gergely

CSALÁSDETEKTÁLÁS MOBIL OKTATÓJÁTÉKOKBAN

KONZULENS

Dr. Forstner Bertalan

BUDAPEST, 2015

Tartalomjegyzék

Összefoglaló

A tanszéki AdaptEd projekt keretein belül elkészült – és továbbra is folyamatos fejlesztés alatt áll – egy innovatív mobil oktatójátékokat támogató keretrendszer, amelynek célja (elsősorban) tanulási zavarokkal küzdő gyerekek tanulási folyamatának élvezetesebbé és hatékonyabbá tétele. Ezt úgy éri el, hogy a számos rendelkezésre álló szenzor (pl. EEG, EKG, szemmozgás követő), valamint a tanuló teljesítménye alapján úgy kalibrálja a hozzá illesztett oktatójátékot, hogy az a gyerek számára optimális beállításokkal (nehézség, feladattípus, stb.) fusson.

A keretrendszer működése során keletkező adatok (szenzoradatok, a játék futása során keletkező események, eredmények, képernyőképek, stb.) több szempontból is értékesek: egyrészt megfelelően vizualizálva alkalmasak arra, hogy a szakpedagógus távolról vagy utólag átfogó képet kapjon a játék lefolyásáról, a gyerek teljesítményéről és állapotáról, másrészt az elemzésükkel vizsgálható a tanulás folyamata és hatékonysága. Mindkét szempontból fontos azonban, hogy helyes, torzítatlan adatokkal rendelkezzünk. Ki kell tudnunk szűrni azokat a valós alkalmazási környezetben előforduló eseteket, amikor a gyerekek csalnak, pl. a barátaikkal vagy szüleikkel végeztetik el a feladatokat.

Munkám során megterveztem és megvalósítottam az adatok összegyűjtését, kliens valamint szerver oldali tárolását. Készítettem a pedagógusok, valamint a keretrendszer fejlesztői számára egy könnyen kezelhető webes felületet, amelyen táblázatos és grafikus formában is megtekinthetőek a játékmenetek, illetve kidolgoztam egy osztályozó algoritmust, amely képes detektálni a csalás gyanús játékmeneteket. Az ilyen módon összegyűjtött és megtisztított adatok számos későbbi analízis alapjául szolgálhatnak.

Abstract

A framework that supports innovative educational mobile games is being developed within the confines of the AdaptEd project at the Department of Automation and Applied Informatics. Its purpose is to make the process of learning for children (primarily with learning disabilities) more effective and enjoyable. To achieve this goal it uses several available sensors (e.g. EEG, EKG, eye tracking device) to calibrate the joint educational game to run with optimal settings (difficulty, task type, etc.).

The data from the operation of the framework (sensor data, events occurring during the game, results, screenshots, etc.) are valuable in multiple respects; firstly, if they are properly visualized then the special teacher can use them to get an overall picture of the game's course, the child's performance and state without close proximity, or later in time. Secondly, we can examine the process and the effectiveness of learning with their analysis. However, it is important in both aspects that we possess proper and distortion-free data. We need to be able to filter out those real life cases when the children cheat, e.g. they have their friends or parents do the exercises.

During my work, I planned and implemented the collection of data, and their storage both on the client and the server side. I created a user-friendly web interface for the teachers and the developers of the framework, which shows the gameplays in a tabular and a graphical way. In addition, I worked out a classification algorithm capable of detecting suspicious gameplays that may be frauds. The data collected and cleaned such way can be used for later analysis.

1 Bevezetés

A XXI. század elején megfigyelhető felgyorsult technológiai fejlődés, amely magával hordozta többek között az „okos” eszközök – okostelefonok, tabletek – globális elterjedését [1] az átlagosnál is nagyobb hatással van arra a legfiatalabb generációra, amely ezekkel a technológiai eszközökkel nőtt fel. Ezt a változást az oktatás fejlesztése során nem lehet figyelmen kívül hagyni [2]: jó példa erre a magyar közoktatásban is egyre népszerűbb interaktív tábla [3], valamint több olyan kutatás [4][5], melyek kimutatták, hogy digitalizált környezetben az ADHD-vel (figyelemhiányos hiperaktivitás-zavar, Attention Deficit Hyperactivity Disorder) diagnosztizált gyerekek mind reál, mind humán területen jobb eredményeket érnek el.

Egy finn felmérés szerint első osztályosok 84%-a játszik számítógépes játékokkal időnként, 31%-uk pedig minden nap, akár órákat [6]. Célravezető megközelítésnek tűnik tehát a gyerekek számítógépes játékok iránti érdeklődésének felhasználása az oktatás területén.

Az AdaptEd projekt (elsősorban) tanulási nehézségekkel küzdő – diszlexiás, diszkalkuliás – gyerekek számára készült, célja a tanulás hatékonyabbá és élvezetesebbé tétele. Ezt nem pusztán azzal kívánja elérni, hogy az oktatást áthelyezi az „új generáció” számára megszokott és ismert digitális környezetbe, hanem emellett egy adaptív tanítási modellt használ – ebből származik a projekt elnevezése.

1.1 Az AdaptEd keretrendszer

A projekt központi eleme az AdaptEd keretrendszer, amely egy androidos környezetben futó „segédalkalmazás”, ehhez vannak egy interfészen keresztül illesztve az oktatójátékok – pl. a diszkalkuliás gyerekeknek készült Korongház [7] vagy a Meixner alapítvány [8] módszertanát alkalmazó Meixner játékok. Fontos, hogy ezek valóban csak példák: a keretrendszerhez lényegében bármilyen fejlesztő (vagy fejlesztéssel foglalkozó cég) készíthet alkalmazásokat, amelyek a programozó számára szinte transzparens módon képesek kihasználni a rendszer nyújtotta lehetőségeket. Már most 10-nél több ilyen játék létezik: kottaolvasást és gitározást tanító, adaptív intelligenciát mérő, a felhasználó érdeklődését feltérképező, stb.

Az illesztett játékok és a keretrendszer kapcsolata kettős: egyrészt a játékok publikálják, majd elküldik a hozzájuk tartozó eseményeket és jutalmakat a keretrendszer felé, másrészt a keretrendszer a játékokból származó teljesítmény adatok, valamint a keretrendszerhez illesztett biofeedback eszközök (pl. EEG, EKG, szemmozgáskövető) alapján ajánlásokat tesz a játék finomhangolására, hogy az pont a gyermek aktuális állapotának megfelelő beállításokkal (nehézség, játéktípus) működjön.

1.2 Adatgyűjtés

A keretrendszer – még mindig folyó – fejlesztése során többek között az én feladatom volt a rendszer adatorientált aspektusainak megtervezése és megvalósítása. A rendszer és az oktatójátékok futtatása során számos olyan adat keletkezik, melyek a későbbiekben hasznosak lehetnek. Manapság az ilyen adatok tárolásának költsége elenyésző a potenciális hasznosságukhoz képest, ezért a tervezés során úgy döntöttünk, hogy minden keletkező adatot (ésszerű keretek között – pl. a játékok képernyőképeit tömörítve) tárolni fogunk. Ilyen adatok a játékok által definiált események, jutalmak, a csatolt szenzorokból érkező jelek, a játék menetéről készült képernyőképek, a felhasználói interakciói (érintések), stb.

Az összegyűjtött adatokat jelenleg két fő célra kívánjuk felhasználni (azonban a későbbiekben természetesen más szempontból is hasznosak lehetnek):

- Az adatokat felhasználóbarát módon megjelenítve a tanárok (vagy esetleg a szülők) később áttekintést nyerhetnek a játékok lefolyásáról, a tanuló teljesítményéről és állapotáról.
- Adatbányászati és adatelemzési módszereket felhasználva összefüggéseket kereshetünk az adatok között, hipotéziseket állíthatunk fel illetve vizsgálhatunk a segítségükkel. Jó alapot szolgálhatnak például olyan kutatásokhoz, melyek a tanulás hatékonyságát, eredményességét vizsgálják.

1.3 Csalásdetektálás

Az adatgyűjtés során a kooperációban szereplő pedagógusok felhívták a figyelmünket egy gyakori – és később általunk is tapasztalt – problémára: az általános iskolás (vagy fiatalabb) gyerekeknél megfigyelhető, hogy a táblagépet egymásnak (a barátaiknak, esetleg szüleiknek) adogatják. Az ilyen játékokról összegyűjtött adatok

nyilván nem reprezentálják hűen a regisztrált tanuló szokásait, és ezek elemzése problémákhoz, hamis eredményekhez vezethet, következésképpen az ilyen adatokat mérési hibának kell tekintenünk, és ki kell tudnunk őket szűrni.

Ennek érdekében kidolgoztam és validáltam néhány algoritmust, melyek a bemenetüket képező játékmenetről jó hatékonysággal el tudják dönteni, hogy az valóban a hozzá tartozó tanuló eredménye-e.

Természetesen a probléma nem csak szándékos csalás útján fordulhat elő, lehet, hogy csak egy kíváncsi barát, vagy egy jó szándékú szülő kezébe kerül a készülék annak realizálása nélkül, hogy ezzel mérési eredmények kompromittálódnak.

1.4 A dolgozat felépítése

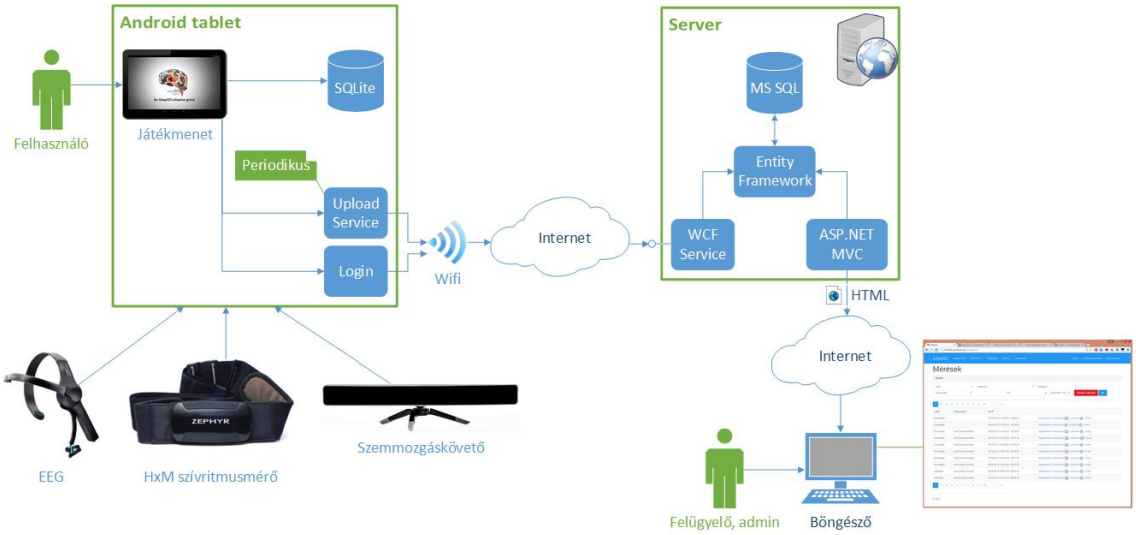
A dolgozat 2. fejezetében bemutatom a megtervezett rendszer architektúráját, az adatgyűjtés folyamatát, valamint röviden a tanárok számára elkészített webalkalmazást, amely – többek között – az adatok megjelenítését végzi.

Ezután rátérek a csalásdetektálás problémájára, a 3. fejezetben pontosan definiálom azt és felsorolok néhány megoldási lehetőséget – köztük az általunk választott adatelemzési megközelítést. A 4. fejezetben bemutatom az adatelemzési probléma megoldását egy általánosan elismert folyamatmodellen keresztül. Végül az 5. fejezetben ismertetem és értékelem az elért eredményeket.

2 Architektúra

Ebben a fejezetben röviden vázolom az AdaptEd tanszéki projekt keretében kialakított rendszer struktúráját és részletesen bemutatom ezen belül az általam fejlesztett részek felépítését.

2.1 Rendszer architektúra



2.1. ábra A rendszer architekturális vázlata

A 2.1. ábra vázlatosan bemutatja a rendszer – a szerver komponens szempontjából lényeges – részeit. Nem tartalmazza például a felügyelő alkalmazást, mellyel a pedagógusok valós időben felügyelhetik – megfigyelhetik és irányíthatják – a tanulók által használt táblagépeken futó alkalmazásokat: nehézséget állíthatnak, feladattípust módosíthatnak, szünetet rendelhetnek el, stb.

A rendszer logikailag és fizikailag is jól elkülönülő komponensei a kliens és a szerver.

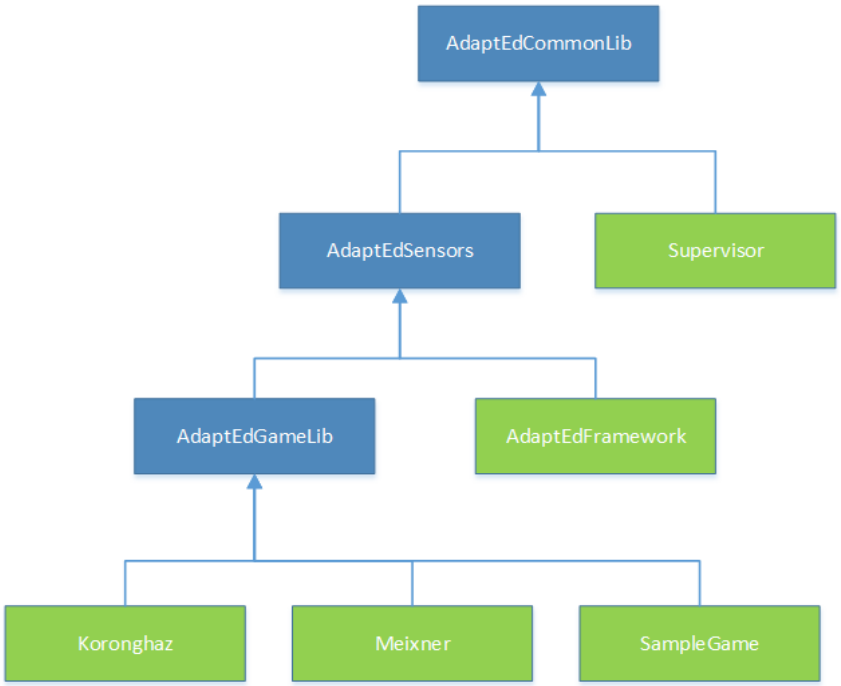
A kliens egy Android alapú tablet: ez futtatja a keretrendszert és az ahhoz kapcsolódó oktatójátékokat. Több biofeedback eszköz kapcsolódhat hozzá, melyek a felhasználót monitorozzák (az élettani jeleit mérik) és a mért adatokat folyamatosan elküldik a keretrendszernek. A keretrendszer ezeket, valamint a futó játékokból származó adatokat átmenetileg egy helyi adatbázisban tárolja, majd periodikusan feltölti a szerverre.

A szerver egy webszolgáltatást publikál: ezen keresztül történik a felhasználó autentikálása, valamint az adatok fogadása és adatbázisba mentése. Emellett hosztol egy webalkalmazást: ez felhasználói felületet biztosít a tanulók menedzselésére (felvétel, alapadatok megadása, jelszó beállítása, törlés, stb.) és a tableten zajlott játékmenetek utólagos megtekintésére a felügyelő oktatóknak, emellett játékok regisztrálására a fejlesztőknek, valamint az intézmények és az azokhoz tartozó tanárok kezelésére az intézményi adminisztrátoroknak.

2.2 Kliens

Az androidos tablet két egymástól jól elkülöníthető alkalmazástípust futtat: a keretrendszert (amely valójában egy háttérszolgáltatás), valamint az ahhoz illesztett oktatójátékokat (ezeknek van felhasználói felületük). A keretrendszerből mindig csak egy példány fut, ez szolgálja ki az összes játékpéldányt.

A keretrendszer több éves fejlesztés eredménye, számos projektből épül fel, ezeket és a köztük levő függőségeket a 2.2. ábra mutatja be.



2.2. ábra A keretrendszert alkotó projektek

A kék színnel ábrázolt projektek kódkönyvtárak – olyan közös kódbázisok, amelyeket több másik projekt is használ: a `CommonLib`-et mindenki, a `Sensors`-t – amely a szenzorok (EEG, szívritmusmérő, szemmozgáskövető, stb.) illesztéséhez szükséges osztályokat tartalmazza – a keretrendszer és a `GameLib`, a `GameLib`-et pedig a

keretrendszerhez illesztett játékok. Az `AdaptEdFramework` maga a keretrendszer, a projekt által nyújtott szolgáltatások használatához ezt kell futtatni a háttérben.

2.2.1 Keretrendszer szolgáltatások

A keretrendszer számos szolgáltatást nyújt a játék számára, ezek közül csak a legfontosabbakat sorolom fel:

- autentikáció: egy java annotációval beállítható és paraméterezhető, hogy a játék igényel-e felhasználói autentikációt. Ha igen, akkor a játék indításakor a keretrendszer megjelenít egy bejelentkező képernyőt (2.3. ábra), majd a bejelentkezés eredményét átadja a játék alkalmazásnak.

2.3. ábra Bejelentkező képernyő

- játékesemények regisztrálása és küldése: szintén annotációkkal definiálhatók játékesemények. Ezek a játék során elsüthetők: ekkor a keretrendszer továbbküldi az eseményt mindenkinek, aki feliratkozott rá (a felügyelő alkalmazásnak, a szervernek, stb.) – ezáltal távolról monitorozható a játék aktuális állapota.
- képernyőképek küldése: bizonyos időközönként a keretrendszer (transzparens módon) képernyőképeket készít a játékról, és azokat broadcastolja.

- jutalmak: a keretrendszer automatikusan jutalmazhatja a játékost pl. egy-egy szint végén a teljesítményének és a fiziológiai jeleinek függvényében.
- javasolt nehézség: a keretrendszer a rendelkezésére álló adatok (biofeedback információk, elért eredmény) javasolhatja a nehézség módosítását.
- játék állapotváltozások kezelése: a felügyelő alkalmazás segítségével az oktató különféle műveleteket végezhet a felügyelt játékokkal, pl. megállíthatja, újraindíthatja őket, értékelheti a tanulókat (ezzel tanítóhalmazt adva a keretrendszer osztályozó algoritmusainak). Ezekről a keretrendszer értesíti a játékalkalmazást.
- játékmenetek kezelése: játékmenetek indítása és megállítása.

2.2.2 Események feltöltése

A játék során keletkező eseményeket (dinamikusan regisztrált játékesemények, biofeedback eszközökből származó események, képernyőkép események, stb.) a keretrendszer először egy lokális SQLite adatbázisba perzisztálja. Erre azért van szükség, mert nem megengedhető, hogy egy esetleges hálózati probléma (nem elérhető WiFi vagy szerver) esetén a játékmenetek elvesszenek: ilyenkor megmaradnak a kliens adatbázisban és később kerülnek feltöltésre.

Az adatok feltöltését egy Android service végzi, mely periodikusan – percenként – fut le. Ez lekérdezi az adatbázisból az összes játékmenetet és az azokhoz tartozó eseményeket, azokból egy JSON üzenetet állít elő és azt küldi el – egyszerű UTF-8 kódolású szövegként – a szervernek. A JSON üzenet maximális mérete korlátozott, ezért a szolgáltatás szükség esetén több darabban küldi el az eseményeket.

Amennyiben a feltöltés sikeres – erről a szerver válaszából lehet meggyőződni, amely minden sikeresen feltöltött esemény azonosítóját tartalmazza – a szolgáltatás törli a lokális adatbázisból ezeket az eseményeket.

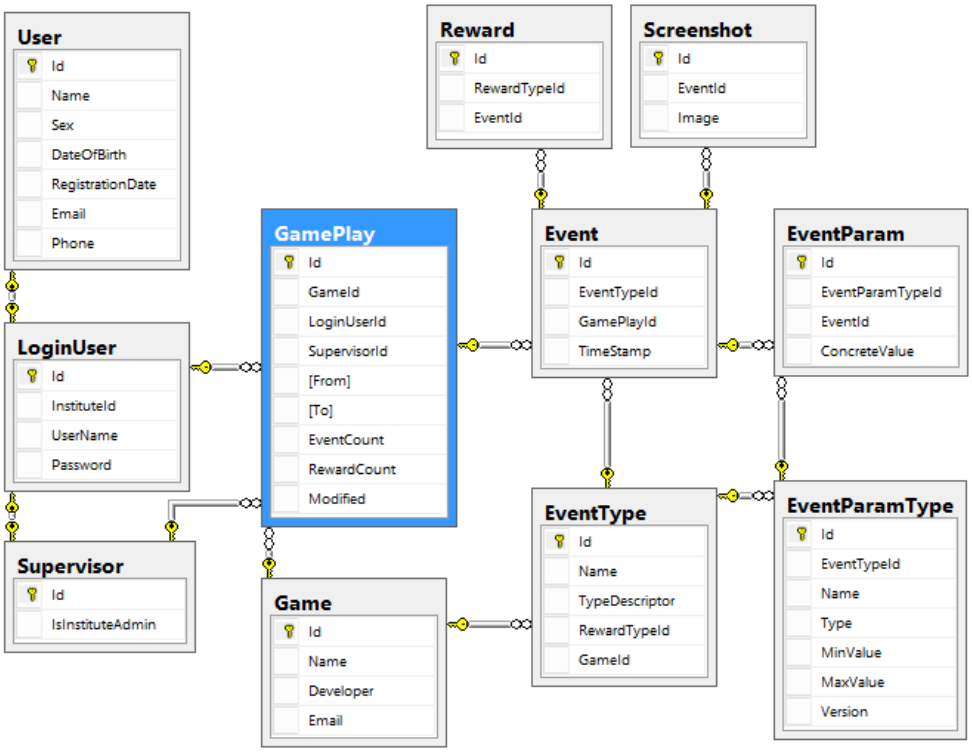
2.3 Szerver

A szerver szerepét egy Windows Server 2012 R2 operációs rendszeren futó IIS (Internet Information Services [9]) webszerver és MS SQL [10] adatbázisszerver tölti be. A szerver a tanszéki hálózaton egy virtuális gépen fut.

2.3.1 Adatbázis

Az adatbázis séma tervezése során maximális normalizáltságra törekedtem, azonban a későbbiek során – teljesítmény okokból – erről le kellett mondanom és néhány cache jellegű duplikátum attribútummal bővítettem a sémát.

A 2.4. ábra mutatja be a séma azon részét, mely a játékmenetekkel kapcsolatos táblákat tartalmazza (az áttekinthetőség kedvéért csak a). A játékmenet (`GamePlay`) az események (`Event`) tárolásának legmagasabb szintű eleme: egy tanuló (`LoginUser`) egy játékkal (`Game`) való foglalkozásának egy darabja. A játékmenet indítását a keretrendszeren keresztül maga az oktatójáték végzi: a játék fejlesztője határozza meg, hogy mikor kezdődik el egy játékmenet (pl. közvetlenül bejelentkezés után, vagy a menüben történő navigációt követően), és hogy mikor ér véget. A játékmenethez tartozhat egy felügyelő tanár (`Supervisor`), aki a felügyelő alkalmazást kezelte a játék során.



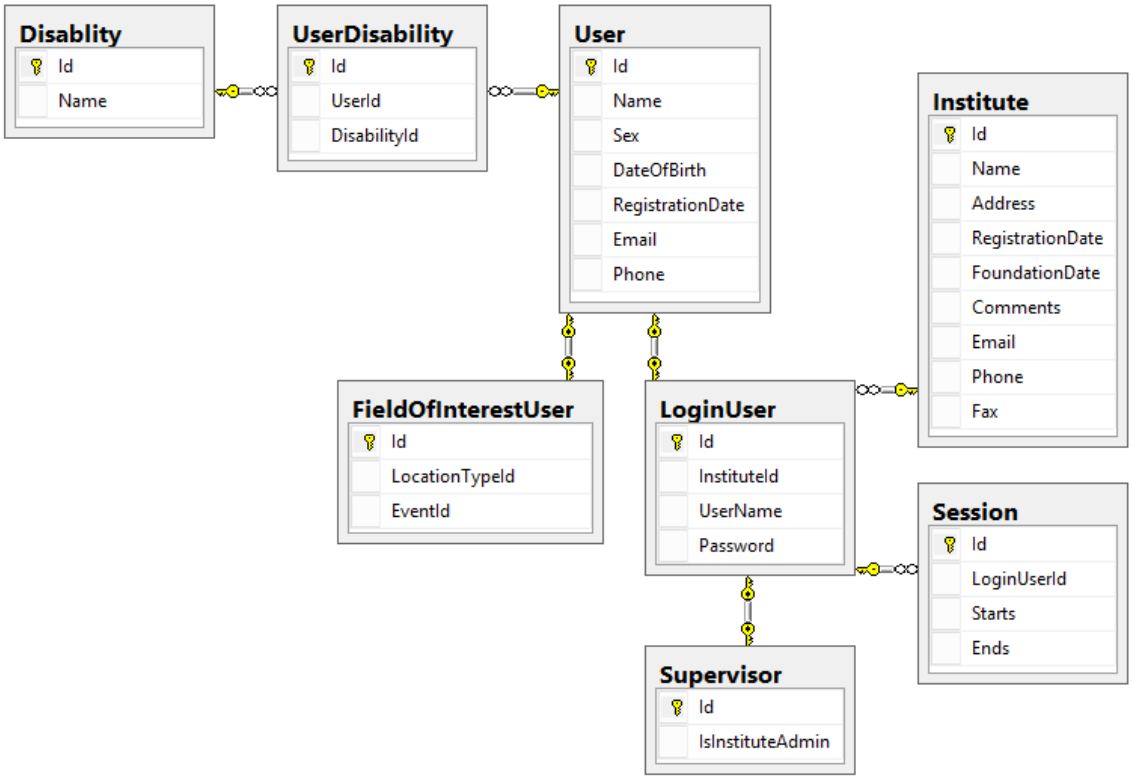
2.4. ábra Az adatbázisséma részlete: a játékmenethez kapcsolódó táblák

A játékmenet eseményeket tartalmaz, melyeknek két attribútuma az esemény típusát illetve idejét tárolja. Példa a cachelésre, hogy az `EventCount` tárolja ezen események számát. Erre azért van szükség, mert a játékmenetek áttekintő oldalának generálásakor – ahol akár 100 játékmenet is megjelenhet egyszerre – nem hatékony minden alkalommal illeszteni az Event táblát. Az eseménytípusokból (`EventType`) kétféle van: statikus és dinamikus. A statikus típusok általánosak és csak név (`Name`) attribútummal rendelkeznek (pl. „`ScreenshotEvent`”), míg a dinamikus típusok egy-egy játék által lettek regisztrálva, ezek neve „`GameDynamicEvent`” és a `TypeDescriptor` attribútum írja le a játékban megadott nevüket (pl. „`hu.bme.aut.adapted.koronghaz.events.NumbercardMovedEvent`”).

Az eseményekhez paraméterek (`EventParam`) tartozhatnak, melyeknek szintén definiált típusai vannak. E típus (`EventParamType`) attribútumai: eseménytípus – idegen kulcs, amely megadja, hogy milyen típusú eseményhez tartozik az paramétertípus, név (pl. „`accuracy`”, „`isCorrect`”), típus (`int`, `double`, `string`), határértékek (`MinValue` és `MaxValue`) és verzió (`Version`). A `Version` attribútum azért szükséges, mert elképzelhető, hogy a játék fejlesztője idővel megváltoztatja egy esemény paramétereit. Ekkor a régi paramétereket is meg kell őriznie az adatbázisnak és az újak fogadására és képes kell, hogy legyen. Ennek lehetőségét biztosítja a verziószám: ha megváltoznak egy dinamikus esemény paraméterei, akkor az új értékek eggyel növelt verziószámmal fognak rendelkezni. Az esemény paraméternek a típusa mellett értéke (`ConcreteValue`) van, amely a típustól függetlenül mindig szöveg – `nvarchar(255)` – típusú.

Az eseményhez tartozhat jutalom (pl. pálya teljesítésekor). A jutalmat leíró táblák struktúrája tökéletesen megegyezik az eseményeket leírókéval (`Reward`, `RewardType`, `RewardParam` és `RewardParamType`), ezért ezeket nem részletezem.

Jól látható, hogy a tervezés során törekedtem arra, hogy az események és jutalmak definiálása dinamikus legyen, az mind a játékok fejlesztői, mind a szerver üzemeltetői számára transzparens módon történik.



2.5. ábra Az adatbázisséma részlete: a felhasználókhoz kapcsolódó táblák

A 2.5. ábra a felhasználók tárolásához kapcsolódó táblákat mutatja be. A felhasználó (`User`) tábla a felhasználók tulajdonságait tárolja, valamint a tanulási zavarokat (`Disability`) egy kapcsolótáblán keresztül. Az objektum-relációs leképezés után ennek leszármazottai a `FieldOfInterestUser` (az „Érdeklődési térképet” – az Érdeklődési Térkép nevű illesztett alkalmazás által generált kérdőívet – kitöltő felhasználó) és a `LoginUser` (aki felhasználónévvel és jelszóval rendelkezik, melyekkel bejelentkezhet a rendszerbe – a jelszónak természetesen csak a hash-ét tárolja az adatbázis), valamint a `LoginUser` leszármazottja a `Supervisor`. Látható, hogy az öröklés relációs leképezéséhez Table-Per-Type stratégiát követtem.

Minden regisztrált felhasználó (`LoginUser`) egy intézményhez (`Institute`) tartozik, amelynek szintén megadhatók a tulajdonságai. A felhasználó belépésekor a rendszer egy Session ID-t generál a számára és a játékmenet feltöltésekor ezzel autentikálja magát. Ezen ID érvényességi ideje a `Starts` időponttól az `Ends` időpontig tart.

Az adatbázisséma a fentiekben bemutatottakon felül számos egyéb táblát tartalmaz (pl. az Érdeklődési térkép, illetve az Adaptive Raven alkalmazáshoz tartozó táblákat), összesen 31 táblából áll.

2.3.2 Webszerver

A következőkben a webszerver projekt felépítését mutatom be.

2.3.2.1 Core

Az `AdaptEd.Core` egy osztálykönyvtár, amely azokat a kódrészleteket tartalmazza, melyeket a többi projekt használ. Ezek közé tartoznak az adatbázis objektum-relációs leképezésével generált osztályok (az ORM-hez Entity Framework-öt [11] használok), valamint néhány segédosztály.

2.3.2.2 Webszolgáltatás

WCF [12] szolgáltatás, amely a kliens-szerver kommunikációhoz biztosít végpontot. Ez a következő szolgáltatásokat nyújtja:

- autentikáció: felhasználó és felügyelő be- és kijelentkezés
- játékmenetek feltöltése
- entitásinformációk (pl. felhasználónév, játéknév azonosító alapján) lekérdezése

A játékmenetek adatbázisba mentését egy feldolgozókból (`Processor`) álló lánc valósítja meg. A lánc lényege, hogy szabadon konfigurálható: az egyes játékmenetekre illetve eseményekre különböző feldolgozók definiálhatók, melyek az osztályukban definiált paraméterek (név, típus, játék, stb.) teljesülése esetén feldolgozzák azokat. Az alapértelmezett feldolgozó a `SaveGamePlayProcessor` végigiterál az eseményeken és azokat továbbadja az eseményfeldolgozóknak: a `SaveEventProcessor` és a `SaveEventParamProcessor` a korábban bemutatott adatbázistáblákba írja az eseményt illetve az eseményhez tartozó paramétereket, a `RewardUpdatedEventProcessor` az `eseményhez` készít egy jutalom objektumot a megfelelő típussal és paraméterekkel, a ScreenshotEventProcessor és a `HeatMapEventProcessor` pedig a különböző képernyőkép eseményeket helyezi el külön táblákba.

Az alapértelmezett mellett azonban definiálható bármilyen tetszőleges feldolgozó. Például az Érdeklődési Térkép nevű játékot teljesen különálló táblákba menti a `FieldOfInterestProcessor`, amely ehhez a teljesen alkalmazás specifikus esemény feldolgozókat használja (`PersonInfoAddedEventProcessor`,

`CardAddedEventProcessor`, `CardRemovedEventProcessor` és `EndGameEventProcessor`).

2.3.2.3 Webalkalmazás

ASP.NET MVC [13] alapú webalkalmazás, amely a felügyelő tanárok és az intézményi adminisztrátorok számára biztosít felhasználóbarát webes felületet.

A technológiának a nevében is benne van, hogy a Model-View-Controller tervezési mintát használja, amelynek a lényege, hogy elkülöníti az alkalmazás által kezelt adatot annak megjelenítésétől, illetve a felhasználói interakcióktól. A minta a következő három komponensből áll:

- A modell tartalmazza az adatokat és az üzleti logikát, jelen esetben az adatbázis objektum-relációs leképezését és néhány kiegészítő (pl. metaadatokat tartalmazó) osztályt.
- A nézet a modell valamilyen grafikus reprezentációja.

  Egy modellhez több megjelenítő objektum is tartozhat – klasszikus példája ennek, hogy a táblázatkezelő programok képesek a táblázatban tárolt adatokat különféle grafikonokon megjeleníteni, esetünkben pedig egy entitáshoz (pl. tanuló vagy csoport) több nézettípus (létrehozás, szerkesztés, részletek, törlés) tartozhat.

  Webes környezetben a nézetek valamilyen transzformációs nyelven megfogalmazott HTML sablonok, amelyek a modellt mint bemenetet HTML-lé konvertálják (ezt illusztrálja a 2.6. ábra). Az ASP.NET MVC több transzformációs nyelvet is támogat, a projekt ezek közül a Razor [14] View Engine-t használja.
- A vezérlő kezeli a felhasználói interakciókat és annak megfelelően módosítja a modellt. Webes környezetben a HTTP (GET és POST) kérésekre adnak választ – jellemzően egy-egy nézet (View) formájában.

```
<div class="form-group">
        @Html.Label("Felhasználónév",
                htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
                @Html.EditorFor(model => model.UserName,
                        new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.UserName, "",
                        new { @class = "text-danger" })
        </div>
</div>
```

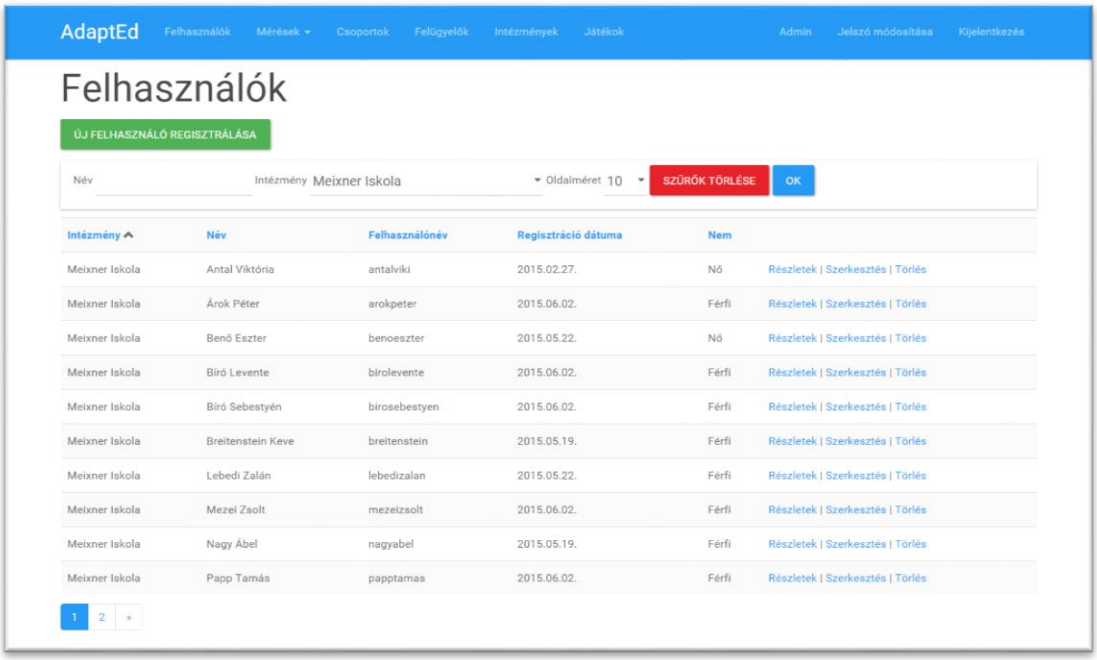
```
<div class="form-group">
        <label class="control-label col-md-2" for="N_v">Név</label>
        <div class="col-md-10">
                <input class="form-control text-box single-line valid"
                 id="Name" name="Name" type="text" value="Novák Gergely"/>
                <span class="field-validation-valid text-danger"
                        data-valmsg-for="Name" data-valmsg-replace="true"/>
        </div>
</div>
```

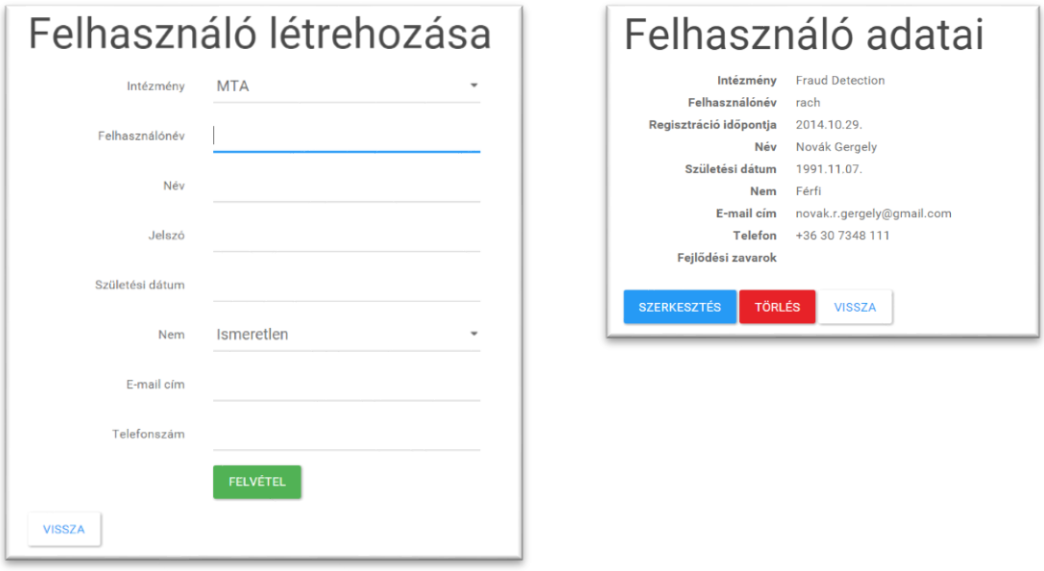
Név Novák Gergely

2.6. ábra A Razor transzformációs nyelv illusztrálása

2.3.2.3.1 Funkcionalitás

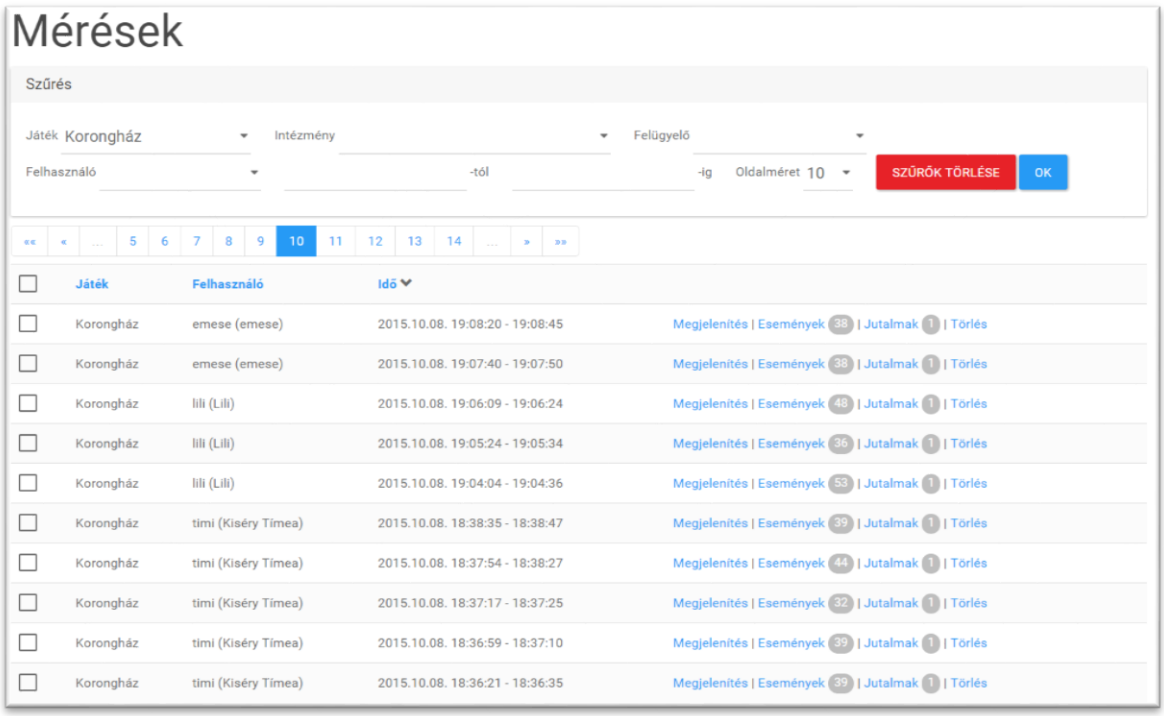
A weboldalnak két jól elkülönülő szerepe van: egyrészt menedzsment felületet biztosít a keretrendszer szereplőinek (pedagógusok, tanulók, osztályok, intézmények, játékok) kezelésére, másrészt felhasználóbarát módon megjeleníti az adatbázisban tárolt játékmenetre vonatkozó adatokat.

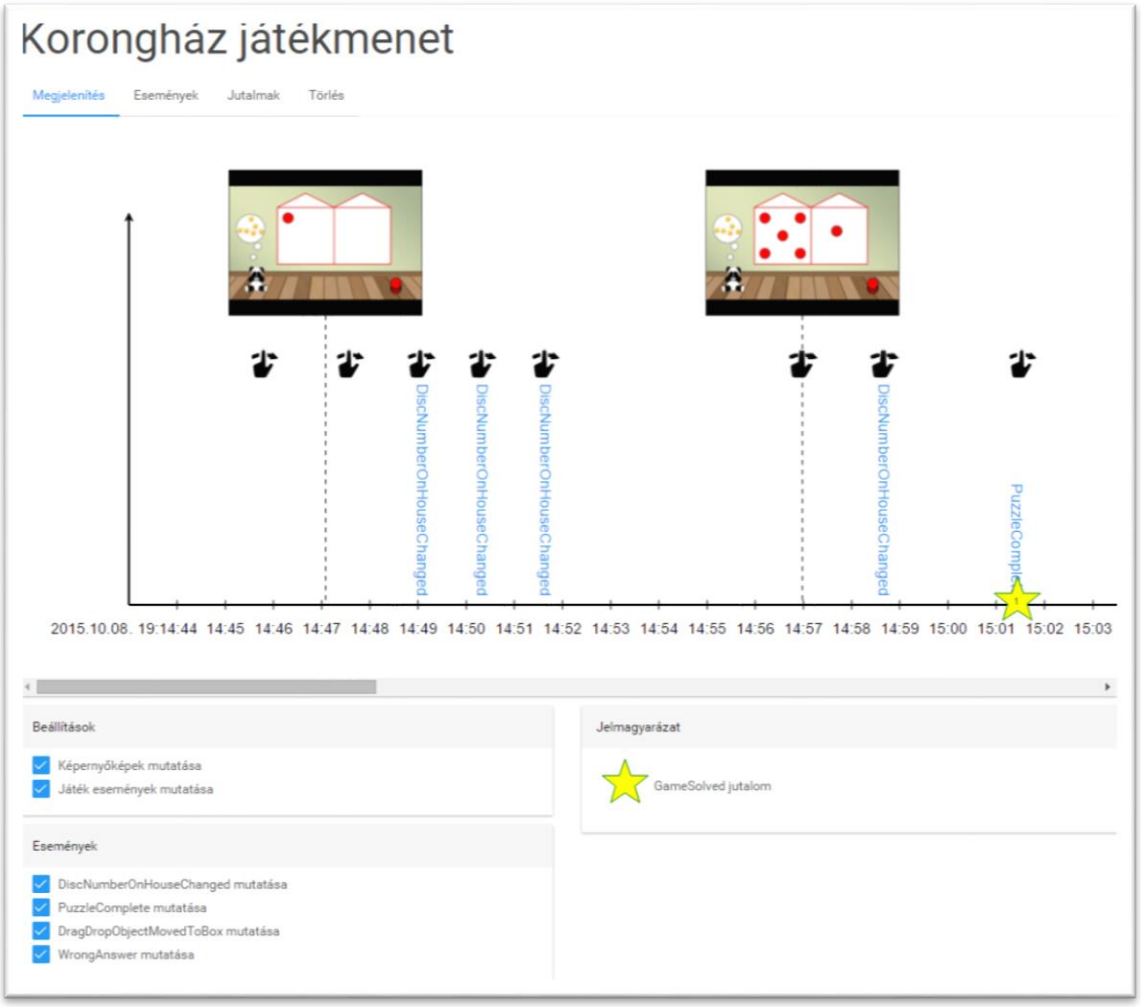




2.7. ábra A tanulók kezeléséért felelős oldalak

A 2.7. ábra példa a szereplők menedzselését, a felhasználók kezelését (listázás, szerkesztés, részletek mutatása, törlés) biztosító oldalakat mutatja be, a 2.8. ábra pedig a játékmenetek listázását, valamint egy játékmenet grafikus megjelenítését. A weblapok számos a modern weboldalaknál megszokott kényelmi szolgáltatást tartalmaznak: szűrési feltételek széles választéka, oszlopok szerinti rendezés, automatikus kiegészítés, állítható oldalméret, stb.





2.8. ábra Játékmeneteket bemutató oldalak

A játékmenetek kétféleképpen jeleníthetők meg: listázó táblázatokhoz (pl. Felhasználók) hasonló módon (az „Események”, illetve „Jutalmak” fülekre kattintva), valamint diagramszerűen (lásd: a 2.8. ábra alsó része). Ez utóbbi nézet biztosítja a szakpedagógusok számára a tanulók munkájának gyors áttekintését. A következő keretrendszerből érkező adattípusokat tartalmazza:

- a játékspecifikus eseményeket (kék színű szöveggel)
- képernyőképeket a játék aktuális állásáról
- jutalmakat (csillagokkal)
- az általános eseményeket (piktogrammokkal)
  - felügyelő tanár megjegyzése
  - touch és swipe események

- biofeedback eszközökből érkező adatokat (lásd: 2.9. ábra)
    - EEG
        - figyelem (kék görbe)
        - nyugodtság (zöld görbe)
    - EKG
        - pulzusszám változása
        - szívritmus variancia (HPV) (piros görbe)
    - szemmozgáskövető
        - pislogás
        - hőtérkép (a képernyőképre rajzolva jelenik meg)

2.9. ábra Biofeedback jelek

Az áttekinthetőség érdekében a megjelenített elemek jelölőnégyzetekkel ki és bekapcsolhatók.

Bizonyos speciális játékokhoz külön nézet tartozik, például a már említett Érdeklődési térképhez generált felhasználói oldalakon megtekinthetők a kitöltők válaszai, illetve a teszt eredménye:

2.10. ábra Az Érdeklődési térkép nézet

3 Csalásdetektálás

Minden adatelemzési projekt sikerességének egyik alapfeltétele, hogy „tiszta” adatokkal dolgozzunk [15]. Az adatgyűjtés során – amelyet az előző fejezetben bemutatott rendszer végez – óhatatlanul hibás adatokhoz juthatunk, amelyek egy későbbi elemzés során téves eredményt okozhatnak. Ilyen hibák előfordulhatnak például a kliens-szerver kommunikáció során: ha az adatokat feltöltő Android szolgáltatás a feltöltés során leáll – okozhatja ezt többek között a készülék kikapcsolása, akkor egy hiányos játékmenet adatait fogja a szerver tárolni.

A továbbiakban egy speciális „hibalehetőséggel” fogunk foglalkozni. Ahogy a 2.4. ábra tábláin is látható, a feltöltött játékmenetek egy-egy felhasználóhoz (`LoginUser`) tartoznak – ahhoz, akinek a felhasználónevével és jelszavával a játék megkezdése előtt a tanuló bejelentkezett. Ez az attribútum kulcsfontosságú lehet olyan adatelemzési feladatoknál, amelyek egy-egy gyermekre (a tanulás hatékonyságának, fejlődésének vizsgálata), vagy gyerekek összehasonlítására vonatkoznak.

Pedagógusok által gyakran tapasztalt jelenség, hogy a tanulók megpróbálják kijátszani a rendszer szabályait – klasszikus példa erre a „puskázás”. Esetünkben általunk is megfigyelt „csalás”, hogy a diákok nem saját maguk végzik el a kapott feladatot, hanem megkérik egy-egy társukat, vagy rosszabb esetben a szüleiket. Az olyan játékmeneteket tehát, amelyeket nem az a tanuló játszott végig, aki a rendszerbe bejelentkezett, ki kell tudnunk szűrni.

3.1 Megoldási lehetőségek

3.1.1 Autentikáció

A keretrendszer használatához a tanuló felhasználónév / jelszó párossal azonosítja magát (lásd 2.3. ábra). Megfelelően erős jelszó esetén így elérhető, hogy ne használhassa akárki a felhasználó nevében a játékokat.

Kézenfekvő megközelítés lehet a fent vázolt probléma megoldására az autentikációs folyamat átdolgozása, vagyis annak a megelőzése, hogy bárki más megtehesse ezt.

3.1.1.1 Ujjlenyomat-olvasó

Az utóbbi években kezdenek elterjedni a mobil eszközökön is (mind iOS, mind android platformon [17]) az ujjlenyomat-olvasók. Ezek használatával megakadályozható a bejelentkezés egy idegen számára – ha eltekintünk magának az ujjlenyomat-olvasó eszköznek az esetleges hiányosságaitól [18].

Esetünkben azonban nem elég, hogy megakadályozzuk, hogy egy idegen be tudjon jelentkezni a keretrendszerbe, hiszen az elsődleges use-case-ben éppen a jogos felhasználó szeretné, hogy az idegen használja a táblagépet, vagyis a bejelentkezést elvégezhetné helyette.

Az, hogy a tanuló távollétében ne legyen használható az eszköz, elérhető lehetne ujjlenyomat-olvasó használatával (többszöri ellenőrzéssel), de ezzel nem érnénk célt, hiszen nem azt kell biztosítanunk, hogy a tanuló jelen legyen a játék során, hanem hogy ő végezze el a feladatokat.

3.1.2 Tanári felügyelet

Megoldja a problémát, ha a játék pedagógusi felügyelet mellett zajlik, amit a keretrendszer maximálisan támogat a felügyelő alkalmazás segítségével. A tanár könnyen észreveszi, ha nem a tanuló játszik az alkalmazással.

Mindazonáltal előfordulnak olyan esetek, amikor nem kivitelezhető a tanári jelenlét – tipikus példája ennek a házi feladat, amelyet a tanuló otthon végez el, vagy a távoli megfigyelés.

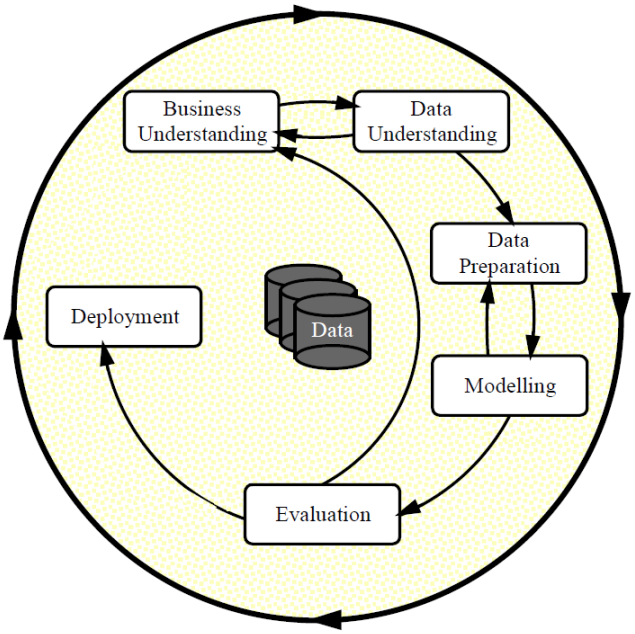
3.1.3 Utólagos adatelemzés

Az általam kidolgozott algoritmus a játék során generált adatok utólagos elemzésével határozza meg, hogy az egyes játékmeneteket valóban a bejelentkezett felhasználó játszotta-e végig.

Ez a feladat önmagában is tekinthető egy teljes adatelemzési problémának, amelynek az általam kidolgozott megoldását a következő fejezetben mutatom be.

4 Adatelemzés

Az előző fejezetben bemutatott probléma megoldására a CRISP-DM (CRoss Industry Standard Process for Data Mining) folyamatmodellt alkalmaztam [19]. A CRISP-DM egy iparág- és technológiafüggetlen folyamatmodell, melynek célja, hogy a különböző adatbányászati feladatok jól kezelhetően, gyorsan, megbízhatóan, megismételhető módon és költséghatékonyan elvégezhetők legyenek.



4.1. ábra A CRISP-DM folyamatmodell fázisai [19]

A CRISP-DM referenciamodell egy projekt életciklust definiál, amely hat iteratív fázisból áll – ezt mutatja be a 4.1. ábra. A dolgozat felépítése is ezeket a fázisokat fogja követni.

4.1 Business Understanding

A projekt célját és körülményeit már az előző fejezetben vázoltam – itt azok strukturált felsorolását végzem el.

4.1.1 Célkitűzés

Az adatelemzési feladat elsődleges célkitűzése az olyan játékmenetek kiszűrése, amelyet nem a bejelentkezett felhasználó játszott végig. A szűrést nem valós időben (tehát a játék alatt), hanem utólag kívánjuk elvégezni.

4.1.2 Rendelkezésre álló eszközök

A „hamis" játékmenetek meghatározásához az AdaptEd keretrendszerből származó, a rendszer szerver komponensén tárolt adatokat fogom felhasználni. Létfontosságú, hogy ezek az adatok a cél szempontjából releváns információkat hordozzanak – erre egyrészt törekedtem a rendszer kialakításakor, másrészt lehetőségem volt a folyamat előrehaladtával bővíteni a gyűjtött adattípusok halmazát.

4.1.3 Az adatelemzés célja

A célkitűzést egy osztályozó algoritmussal kívánom elérni. Az adatelemzés célja, hogy az osztályozás hatékonysága jó legyen, vagyis azok a játékmenetek, amiket az algoritmus csalásnak minősít, nagy százalékban valóban azok legyenek, míg a többi játékmenet tényleg a regisztrált felhasználóhoz tartozzon.

A hatékonyság megállapításához az 5.1.1. pontban definiált mérőszámokat fogom használni, elsősorban az AUC értéket. Egy osztályozó algoritmus teljesítménye a következő hozzávetőleges akadémia pontrendszerrel értékelhető [20]:

AUC	Minősítés
0,9 - 1	kiváló (A)
0,8 - 0,9	jó (B)
0,7 - 0,8	megfelelő (C)
0,6 - 0,7	gyenge (D)
0,5 – 0,6	elégtelen (F)

4.1. táblázat Osztályozók tradicionális minősítése az AUC érték alapján [20]

Az adatelemzés célja, hogy a létrehozott algoritmus ez alapján legalább jó minősítésű legyen.

4.2 Data Understanding

A keretrendszerből származó adatok gyűjtésének másik – jelen feladat szempontjából irreleváns – funkciója, hogy a pedagógusok utólag áttekinthessék a tanulók munkáját. Az adat egy része hasznos lehet ebből a szempontból, viszont feltételezhetően nem alkalmas az adatelemzéshez. Tipikusan ilyenek a játék során készített képernyőképek, melyek a tanár számára értékes információval szolgálnak a játék aktuális állapotáról, azonban a gyermekek megkülönböztetésére nem alkalmasak.

4.2.1 Hasznos adatok

A következőkben megvizsgálom azokat az adattípusokat, melyek ígéretesek a feladat szempontjából.

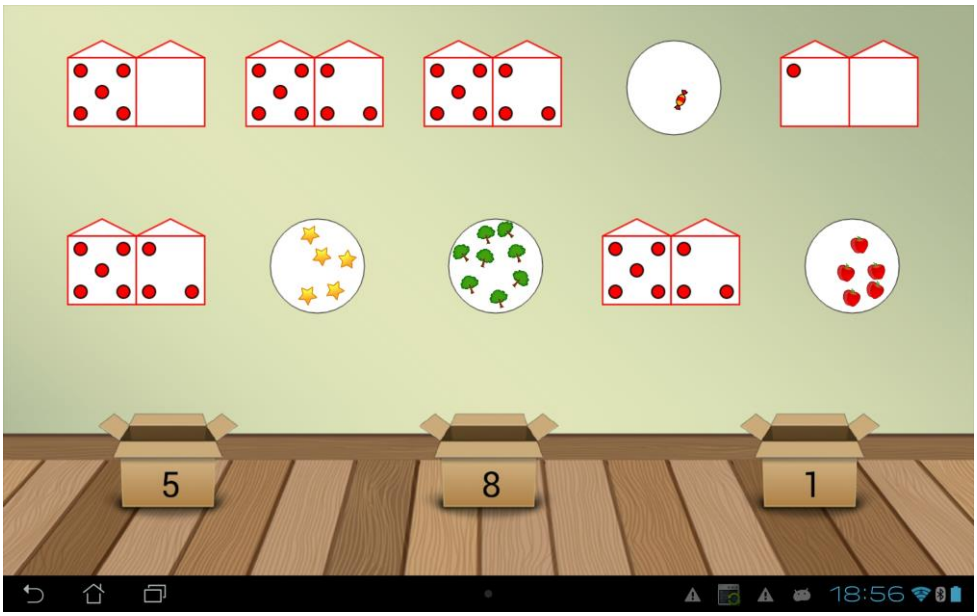
4.2.1.1 A játék és játékmód

Nyilvánvaló, hogy eltérő játékokkal illetve játékmódokkal ugyanaz a tanuló eltérő módon fog játszani. Tipikusan rendkívül különböznek a gyors reakcióidőt és a hosszabb gondolkodást igénylő játékok, valamint a mobil eszköz más-más beviteli lehetőségeit használók (érintés, húzás, giroszkóp, stb.).

Ezért a drasztikusan különböző játékokból (vagy játékmódokból) származó játékmenetek összehasonlítása nem célravezető. Szükséges tehát, hogy az elemzés során ezeket külön tudjuk kezelni. Ehhez felhasználható, hogy a vizsgált játékmenetek mely játékokhoz tartoznak – ezt az információt hordozza a `GamePlay` tábla `GameId` külső kulcsa (2.4. ábra).

Első iterációban az elemzéshez a Korongház [7] alkalmazás egy kiválasztott (Dobozolás nevű, lásd 4.2. ábra) játékmódját használtam, viszont szem előtt tartottam, hogy a kialakított modell tetszőleges másik, a keretrendszer illesztett játékkal kompatibilis legyen.

A Dobozolás játékmódnak két elsődleges feladata van: egyrészt a számokat (számjegyeket) mint absztrakciót kapcsolja össze a tanuló fejében a számnak megfelelő mennyiségekkel, másrészt a számolást mint tevékenységet gyakoroltatja. A diszkalkuliás gyerekeknek ez a két feladat gyakran nehézséget jelent [16], és mivel a hagyományos matematika oktatásban nem kapnak (számukra) kellően nagy hangsúlyt, a későbbiekben egyre súlyosabb problémákat és lemaradást okozhatnak.



4.2. ábra A Korongház alkalmazás "Dobozolás" játékmódja

A játékban 10 ábrán (korongházban vagy körben) levő alakzatokat (körök, csillagok, fák, gyümölcsök, stb.) kell megszámolni és összehúzni az alul található 3 doboz közül azzal, amelyiken az alakzatok mennyiségének megfelelő szám van. Helyes párosítás esetén az ábra hangeffekt kíséretében eltűnik, helytelen esetben pedig lassan visszacsúszik a helyére.

4.2.1.2 Teljesítmény

Az egyes oktatójátékok fejlesztői számára a keretrendszer lehetővé teszi, hogy saját eseményeket és jutalmakat definiáljanak. A Dobozolás példában esemény az objektumok mozgatása (`DragDropObjectMovedToBoxEvent`), valamint a rossz párosítás (`WrongAnswer`), jutalmat pedig a feladat elvégzésekor – tehát 10 sikeres párosítás után – kap a tanuló (`GameSolved`). Kézenfekvőnek tűnhet ezek felhasználása, azonban így nem lenne általános a modell. A játékspecifikus események helyett tehát olyan ezekből számolt derivátumokra van szükség, melyek univerzálisak.

Ilyen derivátumok lehetnek a tanuló teljesítményére vonatkozó mérőszámok, például a helyes illetve helytelen válaszok, megoldások száma – ez a Korongház minden játékmódjára alkalmazható. Előfordulhatnak azonban olyan alkalmazások, amelyekben „válaszok” sincsenek – a „hagyományos” játékok között számos ilyen van (Tetris, autós játékok, stb.), ezért még általánosabb mérőszámra van szükség.

Több iteráció után a választott derivátum a hatékonyság (accuracy) lett, melyet a következőképpen definiáltunk:

$$a = \frac{pontszám}{elérhető\ maximális\ pontszám} = \frac{helyes\ válaszok}{válaszok\ száma}$$

Megjegyzések ehhez a mérőszámhoz:

- 0 és 1 közötti érték.
- A tanuló teljesítményét reprezentálja.
- A definíció kompatibilis az összes jelenlegi oktatójátékkal és játékmóddal. Mindazonáltal ez sem tökéletesen általános: közvetlenül nem használható olyan alkalmazásokra, amelyeknél nincs maximálisan elérhető pontszám (ahol elméletileg akármilyen magas érték elérhető, például az eredeti Tetris játékban, vagy a 2013-as Flappy Bird-ben). Ilyen játékokhoz meghatározhatunk egy olyan magas m pontszámot, amelyet elérve a teljesítményt már 100%-nak tekintjük:

  $$a = \min\left\{\frac{pontszám}{m}, 1\right\}$$

- A hatékonyság kiszámítása természetesen játék specifikus – a definiált események és/vagy jutalmak alapján történik.
- A Dobozolás játékmódban a következőképpen számolható:

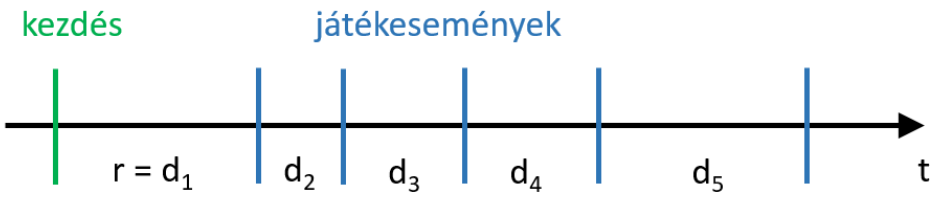
  $$a = 1 - \frac{w}{10 + w},$$

  ahol w a helytelen válaszok – vagyis a `WrongAnswer` események – száma.

4.2.1.3 Idő alapú értékek

A teljesítményt, valamint a tanuló szokásait is reprezentálhatják a játékmenet során történő események között eltelt időintervallumok. Ezek közül jelenleg kettőt használunk.

- „Reakcióidő” (reaction time, r): nem a hagyományos értelemben vett reakcióidőről van szó, hanem a játék kezdete és az első esemény között eltelt idő.
- Késleltetések (delay, d_i): az egyes események (feladatrészek) között eltelt idő. Értelemszerűen ennek a számítása is játékspecifikus.



4.3. ábra Idő alapú értékek

Ezekről a paraméterekről úgy gondoljuk, hogy hordozhatnak tanulóra jellemző információt. Egyes tanulók alapvetően lassabban dolgoznak, jobban áttekintik a helyzetet egy-egy lépés előtt, míg mások rutinból, gyorsan végzik a feladatokat.

4.2.1.4 Érintés értékek

Több kutatás is foglalkozott azzal, hogy a mobiltelefon aktuális felhasználóját annak az érintőképernyő használatára vonatkozó biometrikus viselkedése alapján azonosítsa [21][22][23]. Ehhez a keretrendszernek gyűjtenie kell az érintésekből származó adatokat.

A felhasználó egy-egy érintéséről az Android API-n keresztül lekérdezhetők a következő attribútumok [24]:

- Pozíció: az érintés X, Y koordinátája a képernyőn
- Méret: az érintés felületének mérete, pontosabban az érintőképernyő által érzékelt pixelek száma eszközspecifikus határértékkel normalizálva, 0 és 1 közé skálázva
- Nyomáserősség: az érintés erőssége, szintén 0 és 1 közé skálázva
- Időbélyeg: az érintés ideje

Az érintések csoportosíthatók a hosszuk szerint. Az „egyszerű” érintés (touch) egyetlen érintési pontot tartalmaz, míg a swipe (húzás) gesztus érintések sorozata. Touch eseményre jó példa egy gomb megnyomása, míg swipe-ra a görgetés (scrollozás) vagy két elem összehúzása. A swipe-ot alkotó érintések attribútumai aggregálhatók, kiszámolható pl. a swipe hossza vagy sebessége.

Az érintések gyűjtését végző komponenshez először készítettem egy prototípust: egy különálló Android alkalmazást, amely egy rajzolófelülettel rendelkezik, és a képernyő bal felső sarkában folyamatosan megjeleníti az aktuális érintés attribútumait – ez látható az alábbi ábrán.

4.4. ábra Érintés eseményeket megjelenítő prototípus

A prototípusnak több előnye is volt: egyrészt a segítségével könnyen és kényelmesen vizsgálhatóak voltak a felsorolt attribútumok, másrészt a kódja később kis módosításokkal beépíthető volt a keretrendszerbe.

A vizsgálat során a következő eredményre jutottam: a méret attribútum különösen hasznos lehet gyermek és felnőtt felhasználók szeparálására, mert bár a hivatalos dokumentáció hangsúlyozza, hogy mind a méret, mind a nyomás esetén csak approximációkat tud a rendszer szolgáltatni, a tesztek során szépen elkülönültek a különböző ujjméretek – mind egy ember ujjai, mind a felnőtt/gyerek párok (lásd 4.5. ábra).

Kor	Ujj	Méret
Felnőtt	kisujj	0,27
	gyűrűs ujj	0,4
	középső ujj	0,4
	mutatóujj	0,33
	hüvelykujj hegye	0,53
	teljes hüvelykujj	0,67
Gyerek	kis	0,2
	mutató	0,27
	hüvelyk	0,4

4.5. ábra Érintés méret értékek

4.2.1.5 A mobil eszköz egyéb szenzoraiból származó adatok

A SilentSense nevű 2013-as projekt az érintés eseményekből nem csak az előző pontban bemutatott értékeket használta fel a felhasználó azonosításához, hanem ezek mellett az „eszköz reakcióját” is egy-egy érintés esemény (touch, swipe) után [23]. A reakciót a telefon szenzoraival: a gyorsulásmérővel és a giroszkóppal mérték. Arra jutottak, hogy a telefon reakciói összefüggésben állnak a felhasználóval, elsősorban abból adódóan, hogy a különböző emberek különböző módon tartják és használják a telefonjukat.

Mi úgy döntöttünk, hogy ezeket az adatokat egyelőre mégsem alkalmazzuk, mert azt figyeltük meg, hogy a gyerekek a játékokat futtató 10”-os táblagépeket általában egyáltalán nem szokták tartani, hanem az asztalra helyezik és úgy játszanak velük.

4.2.1.6 Biofeedback értékek

A biofeedback eszközökből származó jelek (pl. pulzus) feltételezhetően szintén jól használhatók lennének az elemzéshez, mert – bár nagymértékben függhetnek az adott helyzettől (milyen lelkiállapotban van a tanuló, végzett-e fizikai aktivitást a mérés előtt, stb.) – valószínűleg számos egyénre jellemző információt hordoznak.

A probléma az alkalmazásukkal az, hogy ilyen eszközök jellemzően csak iskolai környezetben állnak rendelkezésre, amikor – többnyire – úgyis van jelen felügyelő tanár. A házi feladatok elvégzésekor – otthon, amikor feltehetően a legnagyobb a csalás veszélye – nem használnak a tanulók EEG, EKG készülékeket, ezért jelenleg az ezekből származó jeleket nem használjuk fel.

4.3 Data Preparation

Az adatelőkészítési fázis feladata, hogy a kezdeti nyers adatból előállítsuk a modellező eszköz bemenetéül szolgáló adathalmazt. Ezen adathalmaz struktúrája függ a modellezés típusától – ilyen szempontból e két fázis között szoros a kapcsolat, ezért röviden vázolom, hogy milyen modellezési megközelítéseket alkalmaztam a problémára.

4.3.1 Modellezési megközelítések

Az érintés alapú felhasználó autentikációt megvalósító projektek számos különböző módszert dolgoztak ki. Ezek közül két gyakori megközelítést tanulmányoztam részletesen, majd mindkettőt alkalmaztam a vizsgált problémára. Az egyik megközelítés az egyes érintések hasonlóságának (távolságának) meghatározásán alapszik [22][25], a másik pedig osztályozás alapú [23]. Ezeket részletesen a 4.4 pontban fogom bemutatni, itt csak az adatelőkészítés szempontjából lényeges különbségekkel foglalkozom.

4.3.2 Kiválasztás (tisztítás)

A Data Understanding fázisban megvizsgáltam az összegyűjtött adatok potenciális felhasználhatóságát. Az adatelőkészítési fázisban kiválasztom ezek közül azokat, amelyek a modellezés bemenetét fogják képezni.

4.3.2.1 Sorok

Minden tipikus adatelemzési feladatban az adatok egy táblázatban helyezkednek el, amelynek az oszlopai a változókat (attribútumokat), a sorai pedig a megfigyeléseket (rekordokat) reprezentálják. A jelen adatelemzési feladat célja, hogy az egyes játékmenetekről határozzuk meg, hogy csalás-e, tehát a legmagasabb absztrakciós szinten a sorok (megfigyelések) maguk a vizsgálandó játékmenetek.

Az osztályozás alapú modellezésnél (továbbiakban „*C* módszer” mint „classification”) valóban erről van szó, az érintések összehasonlításánál (továbbiakban „*D* módszer” mint DTW) azonban egy-egy érintés is tekinthető egy sornak – ezekről határozzuk meg, hogy az adott felhasználóhoz tartoznak-e, majd a játékmenetről e predikátumok aggregálásával döntünk. Végső sorban azonban mindkét esetben azt kell eldöntenünk, hogy mely játékmeneteket vegyük be az elemzésbe.

A legfontosabb, hogy a tanító halmaz valós, pedagógus által felügyelt mérések adatait tartalmazza. Ki kell szűrni a keretrendszer és az oktatójátékok fejlesztése során végzett teszt méréseket, amelyek során lényegében semmilyen a játékmenetre vonatkozó ellenőrzés nem volt. Emellett figyelembe kell venni, hogy a kiválasztott játékmeneteken minden kívánt attribútum számolható legyen (például nem elfogadhatók azok a játékmenetek, amelyek rögzítésekor az érintési események még nem voltak implementálva).

A másik oldalról viszont elengedhetetlen, hogy a kijelölt játékmenetek kellően „sokszínűek” legyenek – azaz több és többféle (korban, nemben különböző) felhasználótól származzanak, és egy-egy felhasználóhoz elegendően sok rekord tartozzon.

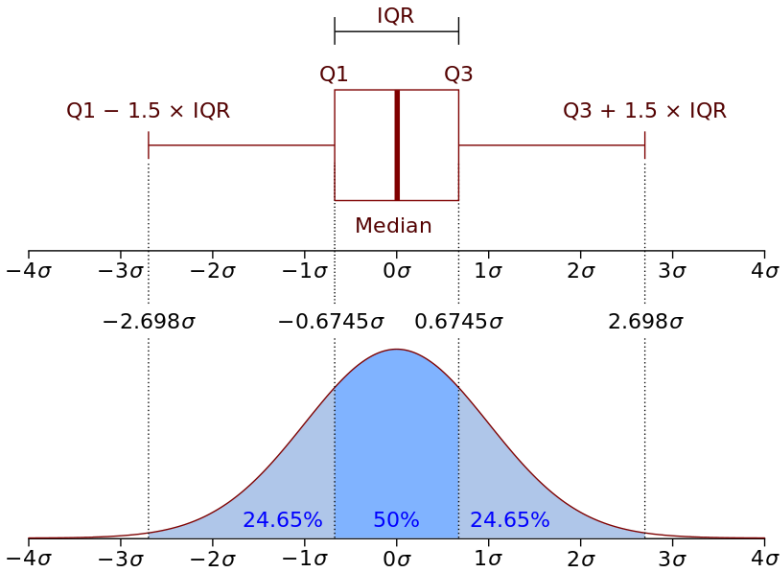
Mindezeket a szempontokat figyelembe véve úgy döntöttem, hogy a sorok kiválasztását (legalábbis az első iterációban) manuálisan végzem el, így az adatbázis kizárólag az összes fenti kritériumnak megfelelő játékmenetet fogja tartalmazni.

4.3.2.2 Oszlopok

Az oszlopok kiválasztásánál szintén nem mindegy a modellezési módszer: míg a *D* módszernél közvetlenül alkalmazhatók az érintések attribútumai (koordináta, méret, nyomás), addig a *C* módszernél csak ezek aggregátumai használhatók fel, mert az előbbi érintéssorozatokat, míg az utóbbi játékmeneteket hasonlít össze. És fordítva, a játékmenetre vonatkozó aggregátumok (játék, teljesítmény, idő alapú értékek) egy „játékmenet-sorra” értelmesek, de egy-egy érintésre nem.

Ennek megfelelően a *D* módszerhez az érintés paraméterei lesznek az oszlopok, a *C* módszernél pedig az összes többi attribútum, valamint az érintés paramétereiből származtatott értékek. Nem triviális, hogy ez utóbbiak pontosan mik legyenek: a koordináták összevonása önmagában nem feltétlenül hordoz információt a felhasználóról – pl. az, hogy honnan indulnak és hova tartanak a swipe-ok vagy hogy milyen hosszúak inkább a játék jellemzői. Ezért a koordinátákból (és az időbélyegekből) először sebességet számítottam – amely intuitíve már inkább a tanuló sajátossága. Ez (valamint a méret és a nyomás is) azonban még mindig touch-okra és swipe-okra vonatkozik, nem játékmenetekre.

A végső megoldást a vizsgált adatok leíró statisztikájának használata jelentette. A leíró statisztika folytonos megfigyelések valamilyen erősen absztrahált jellemzése: ilyen például az átlag, medián, módusz, minimum és maximum értékek. Első iterációban a minimumot, átlagot és mediánt használtam, de ezeknél jobb eredményt értem el a percentilisek (pontosabban a kvartilisek) bevezetésével.



4.6. ábra Kvartilisek bemutatása a normál eloszláson

Az *n. percentilis* definíciója, hogy a (sorbarendezett) adatok azon *n.* eleme, melynél az adatok *n*%-a kisebb. A kvartilisek pedig a Q_1=25., Q_2=50. és Q_3=75. percentilis. A kvartilisek használatának előnye, hogy a minimum, átlag és maximum értékeket „elronthatják” a kiugró – esetleg mérési hibából származó – értékek.

4.3.3 Implementáció

Az adatgyűjtés eredménye (kimenete) a 2.3.1. pontban bemutatott adatbázis, ez szolgál az adatelőkészítési fázis bemeneteként. Kézenfekvő, hogy az adatelőkészítés kimenetét (az előző pontban leírt struktúrának megfelelő adathalmazt) szintén ebben az MS SQL adatbázisban tároljuk – természetesen külön táblában.

A végső megoldásban (amelyről az 5.5. pontban fogok írni) valóban így tettem, azonban a tervezési fázisban az elemzéshez több okból sem az adatbázist kívántam (közvetlenül) felhasználni: egyrészt az adatbázis (az egész tanszéki hálózatra vonatkozó biztonsági megfontolások miatt) csak lokálisan érhető el, az adatelemzési feladatok futtatásával pedig (legalábbis a tervezési fázisban) nem akartam a virtuális gépet terhelni. Másrészt az adatok exportálásának közbeiktatásával lehetőség nyílik lényegében tetszőleges technológia és platform kipróbálására az osztályozáshoz.

Ennek megfelelően létrehoztam szerver projektben egy `AdaptEd.FraudDetection` nevű komponenst, melynek egyik funkciója az adatelemzési tábla generálása. Ez a függvény bemenetként a manuálisan kiválasztott játékmenetek azonosítóit (`GamePlayId`) várja, az adatbázisból összegyűjti a szükséges adatokat, majd

a kész adattáblát CSV fájlba exportálja. A 4.2. táblázat az exportált CSV fájl néhány sorát és oszlopát mutatja.

gamePlayId	userId	reaction	delayQ1	delayMean	delayQ3	accuracy	speedQ1	speedMean
4278	3	2770	727	1339	944	1	770,34	1056,94
4282	3	1725	1026	1309,78	1614	0,8333	941,04	1212,08
4290	151	3555	1148	1441	1639	0,9091	790,6	976,55
4293	153	2059	1602,25	2068,83	2333,25	0,7692	575,64	729,77

4.2. táblázat Részlet az adattáblából

A tervezés és validáció után az adatbázissémát kiegészítettem egy új táblával (`PreparedGamePlay`), amely az exportált CSV-nek megfelelő attribútumokkal rendelkezik és az éles környezetben már ezt használtam.

4.3.4 Normalizálás

Az általam alkalmazott (a következő fejezetben bemutatott) modellezési eljárások hatékonyságát jelentősen növelte, ha standardizált adatokon futtattam őket.

A standardizálást a következő képlettel végeztem el:

$$x_{norm} = \frac{x - \bar{x}}{\sigma},$$

ahol $\bar{x}$ az attribútum elemeinek átlaga, σ pedig a szórása.

A 4.3. táblázat a 4.2. táblázat megfelelő sorait és oszlopait mutatja a normalizálás után. Magától értetődő, hogy csak a folytonos változókat normalizáltuk, a kategóriaváltozókat (játékmenet azonosítója és felhasználó azonosítója) nem.

gamePlayId	userId	reaction	delayQ1	delayMean	delayQ3	accuracy	speedQ1	speedMean
4278	3	0,1766	-0,9933	-0,1781	-0,8994	0,3631	-1,2703	-1,2752
4282	3	-0,1002	0,0053	-0,2325	0,0223	-1,1243	-1,2596	-1,2671
4290	151	0,3845	0,4127	0,0120	0,0567	-0,4480	-1,2690	-1,2794
4293	153	-0,0118	1,9297	1,1818	1,0117	-1,6962	-1,2825	-1,2922

4.3. táblázat Részlet a normalizált adattáblából

4.4 Modellezés

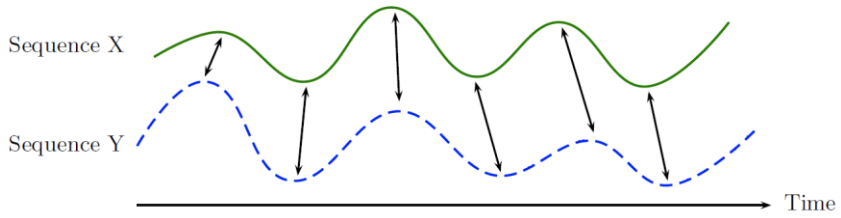
A modellezés során az előző fázis végén megkonstruált adathalmazon futtatunk különböző modellezési technikákat.

A 4.3.1. pontban felvezettem, hogy a szakirodalomban ismert két fő megközelítést vizsgáltam meg részletesen, a következőkben ezeket mutatom be.

4.4.1 Érintések hasonlósága

4.4.1.1 Dynamic Time Warping

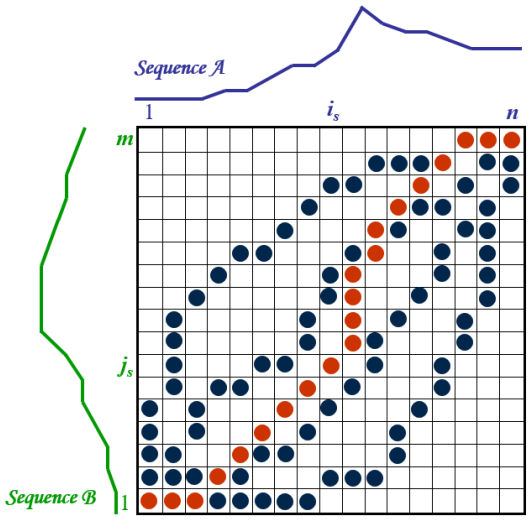
A Dynamic Time Warping (DTW) algoritmus két idősor hasonlóságát (távolságát) határozza meg hatékony módon. Intuitíve az algoritmus a két sorozatot nemlineáris módon egymáshoz illeszti [26], ahogy ezt a 4.7. ábra illusztrálja.



4.7. ábra A DTW algoritmus illusztrációja [26]

Az algoritmus a minimális illesztés (valamint annak költségének) megtalálásához dinamikus programozást használ. Kiszámolja az idősorok pontjainak páronkénti távolságát, majd egy olyan „párosítás-utat” (warping path) keres a sorok végpontjai között, melyek összköltsége minimális, és teljesíti az alábbi feltételeket:

- szigorúan monoton: vagyis a pontok időben követik egymást (ezáltal nem lehet ismétlődés sem)
- folytonos és csak egymást követő pontokat tartalmaz: nincs ugrás az illesztésben (ezáltal nem maradhatnak ki részletek az egyes sorokból)
- a határai a sorozatok végpontjai (nem érhet előbb véget)



4.8. ábra DTW - a "legolcsóbb" illesztés megtalálása [27]

A 4.8. ábra illusztrálja az algoritmus futását:

- A kezdeti feltétel az első két pont távolsága (a táblázat bal alsó sarka)
- A lépés – a felsorolt megkötések miatt – háromféle lehet: „jobbra”, „fel”, vagy „átlósan jobbra”. Az algoritmus ezek közül a legolcsóbbat választja.
- Megállási feltétel: a dinamikus programozás táblázatának a jobb felső elemének elérése.
- Eredmény: az algoritmus eredménye egyrészt az optimális illesztés, másrészt annak költsége (amit szokás idő szerint normalizálni – azaz elosztani az idősorok hosszának összegével)

Az algoritmus futásideje $O(N^2)$.

4.4.1.2 Multi-Dimensional Dynamic Time Warping

Az algoritmus lényegén nem változtat, ha az idősorok pontjai több dimenziósak. Ilyen esetben akkor érjük el a legpontosabb eredményt [28], ha a pontpárok többdimenziós távolságával számolunk – az így módosított algoritmust hívják Multi-Dimensional Dynamic Time Warping-nak (MD-DTW).

A K dimenziós X és Y idősorok i és j pontjának többdimenziós távolsága a következő módon számolható:

$$D(i,j) = \sum_{k=1}^{K} |X(i)_k - Y(i)_k|$$

Természetesen használható négyzetes távolságfüggvény is, de azt az algoritmus szempontjából nem találtuk hatékonyabbnak.

4.4.1.3 Az algoritmus alkalmazása

Az MD-DTW használható egy-egy érintéssorozat (swipe) távolságának meghatározására, hiszen ezek is tekinthetők 5 dimenziós idősoroknak a következő dimenziókkal:

- x koordináta
- y koordináta

- méret
- nyomás
- idő

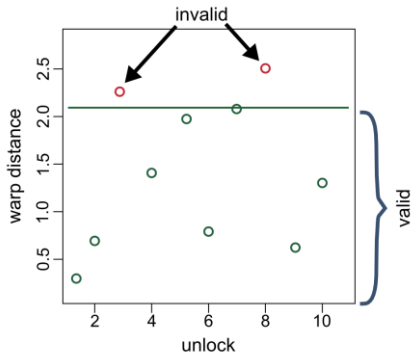
Ezek az összehasonlítások többféleképpen is alkalmazhatók a tárgyalt probléma megoldására.

4.4.1.3.1 A „leghasonlóbb” swipe alapján

A „TIPS” projekt [25] úgy dönti el egy swipe-ról, hogy az a mobiltelefon tulajdonosáé-e, hogy kiszámolja annak és az adatbázisban található összes többi érintéssorozatnak a távolságát a DTW algoritmussal – pontosabban nem az összessel, hanem csak azokkal, amelyek ugyanolyan alkalmazáskontextusban keletkeztek, és potenciálisan hasonlítanak, pl. az irányuk és hosszúságuk hasonló, és a legközelebbi swipe alapján dönt. Ha az a tulajdonosé, akkor elfogadja a bemenetet, ha nem, elutasítja.

4.4.1.3.2 Küszöb alapján

A „Touch me once and I know it’s you!” projekt [22] csak a saját swipe-okkal (a tulajdonoshoz regisztrált sorozatokkal) hasonlítja össze az inputot, ezek közül keresi meg a legközelebbit. Ezután akkor fogad el, ha a talált legközelebbi swipe távolsága a bemenettől egy előre definiált küszöbérték alatt van (lásd 4.9. ábra).



4.9. ábra Érvényes és érvénytelen minták szeparálása [22]

4.4.1.4 Implementáció

Az MD-DTW algoritmust C# nyelven implementáltam egy különálló `MdDtw` nevű libraryben. Két fő oka volt annak, hogy saját implementáció mellett döntöttem: egyrészt nem találtam olyan publikált megvalósítást, amely pontosan megfelelt volna a [28] cikkben leírtaknak, másrészt így jobban megismertem az algoritmust.

Az alábbi kódrészlet az algoritmus fő komponensét, a fentiekben illusztrált dinamikus programozási feladatot valósítja meg:

```
var dtw = new double[A.Length + 1, B.Length + 1];

for (int i = 0; i < A.Length + 1; i++)
{
    dtw[i, 0] = double.PositiveInfinity;
}
for (int i = 0; i < B.Length + 1; i++)
{
    dtw[0, i] = double.PositiveInfinity;
}
dtw[0, 0] = 0;

for (int i = 0; i < A.Length; i++)
{
     for (int j = 0; j < B.Length; j++)
     {
          double cost = 0;
          if (timeDimension != 0 && timeFrame != 0 &&
               Math.Abs(A.Measurements[i][timeDimension] -
                        B.Measurements[j][timeDimension]) > timeFrame)
          {
               cost = double.PositiveInfinity;
          }
          else
          {
               for (int k = 0; k < A.Dimension; k++)
               {
                    cost += Math.Abs(A.Measurements[i][k] -
                                     B.Measurements[j][k]);
               }
          }

          dtw[i + 1, j + 1] = cost + Min(dtw[i, j + 1],
                                         dtw[i + 1, j], dtw[i, j]);
     }
}
distance = dtw[A.Length, B.Length];
```

A távolságszámító algoritmus alkalmazását kipróbáltam mindkét az előzőekben bemutatott technikával – némileg átalakítva a konkrét feladatra. Mivel játékmenetekről kell eldönteni, hogy a felhasználóhoz tartoznak-e (nem egy-egy érintéssorozatról), ezért a swipe-ok besorolása után a játékmenetet a becslések átlaga alapján soroljuk be. A swipe-ok besorolását az előző pontban bemutatott technikák valamelyikével végzem. Ezeket a technikákat valósítják meg az alábbi pszeudokódok:

```
1. módszer (leghasonlóbb swipe alapján):
     input: egy swipe: s (x[1..n], y[1..n], size[1..n], pressure[1..n] és
                          time[1..n] paraméterekkel)
            a mintahalmaz összes olyan swipe-ja, amely nem az s-hez
               tartozó játékmenetben található: S[1..m]
     output: a játékmenet címkéje (0, 1): s.label
```

```
        for each s_i in S do:
             d_i = DTW(s, s_i)
        closest = s_i where d_i minimális
        return closest.label

2. módszer (küszöb alapján):
     input: egy swipe: s (x[1..n], y[1..n], size[1..n], pressure[1..n] és
                          time[1..n] paraméterekkel)
            a mintahalmaz összes olyan swipe-ja, amely nem az s-hez
               tartozó játékmenetben található: S[1..m]
            küszöbérték: k
     output: a játékmenet címkéje (0, 1): s.label

     S_saját = { s_i | s_i.felhasználó = s.felhasználó in S }
     for each s_i in S_saját do:
          d_i = DTW(s, s_i)
     closest_dist = min{d_i}
     if closest_dist > küszöb then
          return 0
     else
          return 1
```

Jól látható, hogy az 1. módszer komplexitása nagyobb, hiszen lényegesen több összehasonlítást kell végrehajtania (azaz a DTW algoritmus többször fut le). Az algoritmusok hatékonyságáról kapott eredményeket az 5.1.3. pontban fogom bemutatni.

4.4.2 Osztályozás

Mivel a probléma lényegében egy osztályozási feladat: játékmeneteket kell osztályozni „érvényes” és „csalás” címkékkel, kézenfekvő a feladat megközelítése osztályozó algoritmusokkal. Az osztályozás felügyelt (supervised) tanulási forma, vagyis már meglévő, megfelelően címkézett megfigyeléseket (tanító halmaz) használ a gépi tanuláshoz.

4.4.2.1 Címkézés

A továbbiakban pozitívnak (1) fogom nevezni azokat a játékmeneteket, amelyek „érvényesek” – vagyis valóban a regisztrált felhasználó játékai, és negatívnak (0) a nem érvényeseket, a „csalásokat”.

Bár a címkék definíciója így egyértelmű, nem triviális, hogy az egyes játékmenetek milyen címkét kapjanak. Egy lehetséges megközelítés, hogy a garantáltan érvényes (szakpedagógus által felügyelt) játékmenetek legyenek a pozitívak, míg a többi negatív. Ezzel az a probléma, hogy könnyen előfordulhat – sőt, gyakori, hogy a „csalás”-nak címkézett mérések valójában érvényesek, csak éppen nem volt jelen

felügyelő a rögzítésükkor. Alternatívaként kreálhatunk csalásméréseket, amikor garantáltan nem a bejelentkezett felhasználó játszik az alkalmazással.

Az általunk választott megoldás tulajdonképpen ezen alternatíva átdolgozása: a problémát úgy közelítjük meg, hogy minden vizsgálni kívánt játékmenethez tartozó felhasználóhoz egy-egy külön osztályozási feladatot definiálunk, azaz nem azt kívánjuk (közvetlenül) eldönteni, hogy csalás-e az adott játékmenet, hanem hogy az adott felhasználóhoz tartozik-e. Pozitívnak (1) címkézzük azokat, melyek a kiválasztott felhasználóhoz tartoznak és negatívnak (0) azokat, melyek nem. Ez a címkézés értelemszerűen megegyezik az eredeti definícióval.

4.4.2.2 Tanító halmaz kialakítása

Hagyományosan az osztályozási feladatoknál szokás az összegyűjtött megfigyelésekből kialakítani egy statikus tanítóhalmazt, ezen tanítani az osztályozót, majd azt futtatni egyrészt a maradék megfigyelésen (ellenőrzés és hatékonyságmérés céljából), másrészt az ismeretlen, nem címkézett adatokon. Ettől a gyakorlattól egyetlen pontban tértünk el: nem statikus tanítóhalmazt hozunk létre, hanem dinamikusat – vagyis minden vizsgálandó megfigyeléshez másikat. Ennek a megközelítésnek egyértelmű hátránya, hogy a több tanítóhalmazhoz több modellt kell építenünk, amely teljesítmény szempontból nem hatékony. A következő okokból döntöttünk mégis így:

- Egy t időpillanatban kialakított halmaz nem veheti figyelembe azokat a játékmeneteket, amelyek t után keletkeztek. Ez esetünkben azért probléma, mert egyrészt a vizsgált tanulók idővel változhatnak, fejlődhetnek, új szokásokat alakíthatnak ki, ezáltal a játékstílusuk is módosulhat – épp ezért az is fontos, hogy ne használjunk túlságosan régi méréseket a tanításhoz, másrészt akár új tanulók is kerülhetnek a rendszerbe – ez esetben a tanuló halmazban nem is lenne pozitív rekord.
- A predikció sebességének csökkenése nem jelent súlyos problémát, mert annak nem valós időben kell megtörténnie, hanem utólag. Egy olyan rendszernél, amelynek online kell működnie (pl. a már hivatkozott transzparens felhasználói autentikáció), nem feltétlenül engedhetők meg többletszámítások, de esetünkben nem erről van szó.
- Mivel a keretrendszer jelenleg még tesztfázisban van – most történik az iskolai bevezetése – nem áll rendelkezésünkre tetszőlegesen sok adat,

ezért fontos lehet minden elérhető információ (játékmenet) beépítése a tanítóhalmazba.

4.4.2.2.1 Összes többi

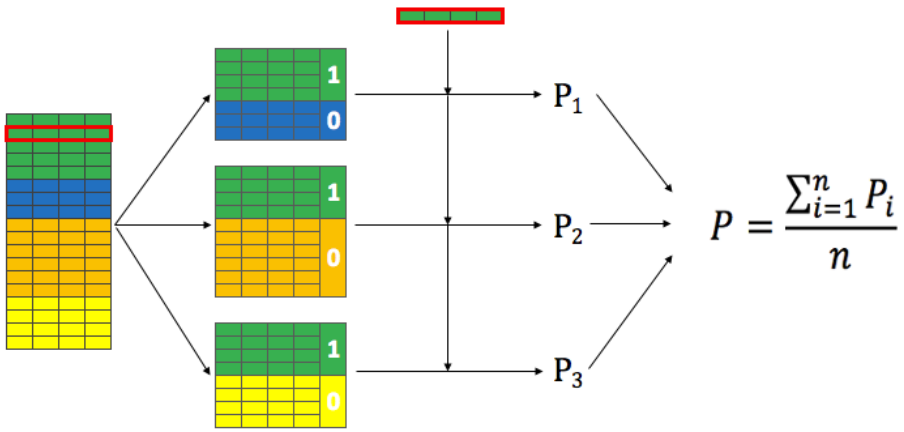
A dinamikus tanítóhalmaz kialakításának egyik lehetséges módja, hogy abba belevesszük az osztályozandó (validáló vagy ismeretlen) játékmenet kivételével az összes többit – a 4.4.2.1. pontban leírt módon: pozitív címkével az ugyanahhoz a felhasználóhoz tartozókat, negatív címkével a többit.

Ennek a módszernek két problémája van: egyrészt (a tanulók eloszlását körülbelül egyenletesnek feltételezve) sokkal több 0-ás érték lesz, mint 1-es, és ezt bizonyos osztályozók rosszul kezelik (nagyobb valószínűséggel prediktálnak 0-ást, mint 1-est), másrészt fennáll a túltanulás veszélye. Mégis – különösen kis mintaszám esetén – hatékonyan használható ez a módszer.

4.4.2.2.2 Páronként

Az előbb említett mindkét problémát kiküszöböli az, ha nem használjuk fel az összes többi játékmenetet, hanem valamilyen stratégiával kiválasztunk közülük néhányat.

Ilyen stratégia lehet a véletlen választás, azonban ekkor figyelni kell arra, hogy a tanítóhalmazban mindkét címke reprezentálva legyen, ráadásul viszonylag egyenletesen.



4.10. ábra Tanítás és predikció páronkénti tanító halmazzal

Az általunk választott stratégia az, hogy „felhasználópár-halmazokat” alakítunk ki, vagyis felhasználjuk az összes játékmenetet, de nem egyszerre, hanem tanulónként

csoportosítva: ezt illusztrálja a 4.10. ábra. Az ábrán színek jelölik a különböző felhasználókat. A bal oldali táblázat reprezentálja a teljes adathalmazt – a Data Preparation fázis kimenetét, a piros sor pedig a validálandó elemet (vagy a validálandó elemek halmazát). A teljes adathalmazból kivesszük ezt a sort (vagy ezeket a sorokat) és a maradékból a tanulók számánál eggyel kevesebb (n, az ábrán három) darab tanító halmazt képzünk, melyek mindegyike tartalmazni fogja a vizsgált tanuló játékmeneteit pozitív címkével, és pontosan egy másik tanuló játékmeneteit negatív címkével. Az osztályozó algoritmusokat ezeken a halmazokon tanítjuk, majd a validálandó/ismeretlen elemmel/elemekkel futtatjuk: a kapott predikátumokat jelöljük P_1, P_2, ... P_n-nel. Végül a döntésünk e predikátumok aritmetikai közepe lesz.

Ennek a stratégiának előnye, hogy a pozitív és negatív értékek eloszlása egyenletes lesz, hátránya viszont, hogy tovább növeli a komplexitást és a végrehajtási időt, de ezt az előzőekben leírtak alapján elfogadható trade-offnak tekintjük.

4.4.2.3 Osztályozó algoritmusok

A tanító halmaz kialakítása után a modellezés fázisban kipróbáltam számos bevett osztályozó algoritmust. Ezt elősegíti, hogy a legtöbb gépi tanulást támogató platformban – így az általam használt python alapú scikit-learnben [29], valamint a C# alapú Accord.NET-ben [30] – az algoritmusokat implementáló osztályok egy magas szintű interfészt valósítanak meg (kisebb-nagyobb eltérésekkel), így könnyen cserélhetők. Ez az interfész a következő elemeket tartalmazza:

- létrehozás és parametrizálás: a különböző paraméterek (pl. a kNN-nél a k, vagyis a vizsgált legközelebbi szomszédok száma) megadásával finomhangolható az osztályozó belső működése és ezáltal javítható a hatékonysága
- tanítás a címkézett tanító halmazon
- predikció az ismeretlen (vagy validálandó) halmazon: általában e művelet kimeneteként megadható, hogy a predikátumokra, vagy azok valószínűségére (konfidencia) vagyunk kíváncsiak – jelen esetben a pozitív/negatív (0/1) értékekre, vagy azok valószínűségére, pl. $P = 0$, vagy $P(0) = 0{,}73$, $P(1) = 0.27$

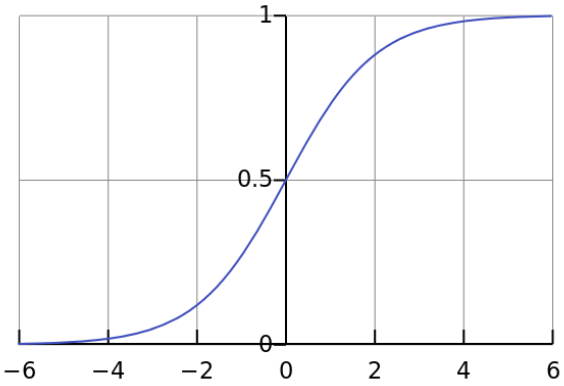
Ezen az absztrakciós szinten az összes alább ismertetett osztályozó ugyanúgy működik. A továbbiakban azok belső működését fogom röviden vázolni.

4.4.2.3.1 Logisztikus regresszió

A logisztikus regresszió – a nevével ellentétben – valójában egy osztályozó algoritmus (azaz a célváltozó nem numerikus, hanem kategorikus típusú), amely a fent leírtaknak megfelelő interfészt valósít meg.

A működése hasonló a lineáris regresszióhoz, amely a függő változót (célváltozót) a független változók lineáris kombinációjával közelíti: ez szemléletesen azt jelenti, hogy – kétdimenziós esetben – a lineáris regresszió egy egyenest illeszt a ponthalmazra. A különbség az, hogy a logisztikus regresszió nem értéket, hanem valószínűséget becsül, azaz nem tetszőleges valós, hanem 0 és 1 közötti értéket, és ehhez a logisztikus függvényt használja fel.

$$\sigma(x) = \frac{1}{1 - e^{-x}}$$



4.11. ábra A logisztikus függvény definíciója és alakja

További hasonlóság a lineáris és a logisztikus regresszió között, hogy matematikai értelemben mindkét esetben egy-egy hibafüggvény minimalizálásáról (vagyis optimalizációs problémáról) van szó: lineáris esetben ez a függvény a pontok egyenestől való távolságának az összege (vagy négyzetösszege), logisztikus esetben pedig egy – a logisztikus függvénnyel számolt – költségfüggvény. Az általam használt implementáció konkrétan a következő hibafüggvényekkel számol:

$$\min_{w} ||Xw - y||_2^2 \quad \left| \quad \min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

4.12. ábra Lineáris és logisztikus regresszió mint optimalizációs problémák [31]

4.4.2.3.2 k-Nearest Neighbour

A kNN (k-Nearest Neighbour) osztályozó működési mechanizmusa az, hogy megkeresi az osztályozandó megfigyeléshez „leghasonlóbb” *k* megfigyelést a tanító halmazból, és ezek címkéje alapján dönt. A „hasonlóság” mértékét valamilyen távolságfüggvénnyel konkretizáljuk – a legáltalánosabb választás erre az Euklideszi-távolság (ezért is fontos a normalizálás), a döntés pedig a legközelebbi rekordok szavaztatásával történik: bináris esetben lehet a címkék átlaga, vagy a távolságfüggvénnyel súlyozott átlaga.

A kNN algoritmus egyik ismert előnye, hogy kis mintaszámon is jó eredménnyel működhet.

4.4.2.3.3 Random Forest

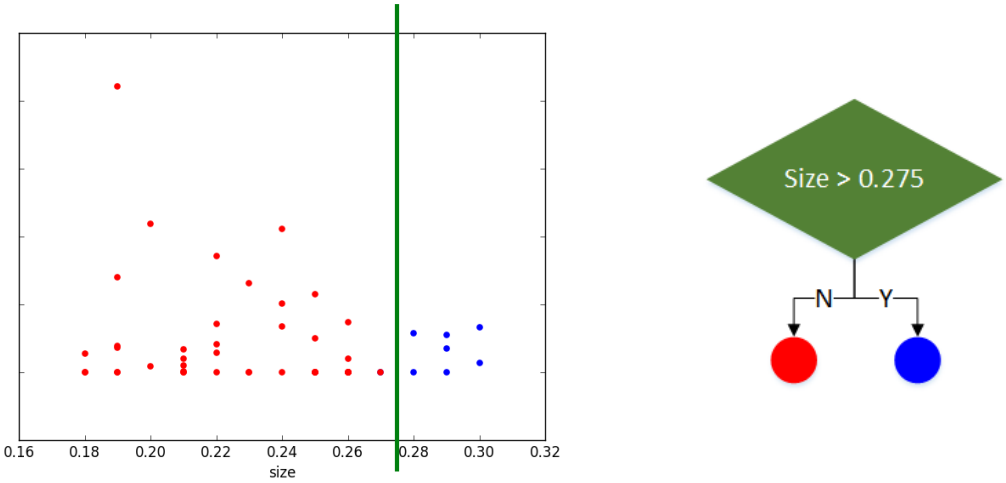
A Random Forest osztályozó bemutatásához először ismertetem a döntési fákat.

A döntési fa grafikusan ábrázolható egy olyan fagráffal, melynek csomópontjai feltételeket, levelei pedig döntéseket (címkéket) tartalmaznak. Klasszikus példája (4.13. ábra), amikor az osztályozónak az időjárási körülményeket leíró kategóriaváltozókból kell „eldöntenie”, hogy érdemes-e golfozni. Az ábra jobb oldalán levő fa kékkel jelöli a feltételeket (pl. Outlook = Sunny, Windy = FALSE) és narancssárgával a döntéseket.

4.13. ábra A döntési fa klasszikus példája [32]

A döntési fát egy generáló algoritmus (a legnépszerűbbek az ID3 [33] és a C4.5 [34]) a tanító halmaz alapján hozza létre – ez a tanuló fázis. A generáló algoritmus mögött húzódó gondolat az, hogy a (maradék) adathalmazt iteratívan úgy próbáljuk lineárisan szeparálni egy-egy feltétel mentén, hogy a szeparátumok minél homogénebbek legyenek.

Egy ilyen szeparálásra mutat példát a vizsgált probléma kontextusában a 4.14. ábra, amin jól látható, hogy a Size $> 0{,}275$ feltétel tökéletesen homogén – csak pozitív, illetve csak negatív mintákat tartalmazó – részhalmazokra bontja a megfigyeléseket.



4.14. ábra A megfigyelések szétválasztása méret szerint

A példa csak a szemléltetés célját szolgálja az alábbi megjegyzésekkel:

- Az esetek többségében a mintahalmaz nem bontható fel tökéletesen homogén részekre.
- Ahogy már említettem, a szeparáció a tanítás során algoritmikus módszerekkel történik, nem manuálisan.

Végül a fa kiértékelése – vagyis a bemenet osztályozása – a fa fentről (gyökértől) lefelé történő bejárását jelenti: a bejárás minden csomópont után a feltételnek megfelelő részfa felé folytatódik az első levél (címke) eléréséig – a levél értéke lesz a predikátum.

A Random Forest egy együttes osztályozó (vagy metatanuló). Az együttes osztályozók több osztályozó algoritmus együttes használatával kívánnak pontosabb eredményt elérni. A Random Forest az tanító halmazt oszlopok szerint véletlenszerű részhalmazokra bontja, majd e részhalmazokból döntési fákat generál. Az osztályozást ezek után úgy végzi, hogy a bemenetet – pontosabban annak megfelelő részhalmazát – minden generált döntési fán futtatja, a végső predikátum pedig a fák eredményeinek átlaga lesz.

4.4.2.3.4 AdaBoost

Az AdaBoost [35] szintén egy olyan együttes osztályozó, amely több (gyengébb) algoritmust használ. A Random Forest-től abban különbözik, hogy nem a tanítóhalmaz részhalmazian tanítja a gyengébb osztályozókat, hanem a teljes halmazon, iteratív módon, egy a rekordokhoz rendelt súlyváltozó bevezetésével:

- Az első iterációban a súlyváltozó eloszlása egyenletes, vagyis ez az iteráció tulajdonképpen egyszerűen lefuttatja a gyenge algoritmust, például (de nem feltétlenül) egy egyszerű döntési fát.
- Ezután – illetve minden iteráció után – megnöveljük a helytelenül becsült sorokhoz tartozó súlyokat, így az iterációk során a nehezen becsülhető rekordok egyre nagyobb súlyt kapnak.
- Végül – előre meghatározott számú iteráció után – „összefűzzük" az iterációk során létrehozott modelleket: a végső predikátum ezen modellek predikátumainak a súlyozott átlaga lesz.

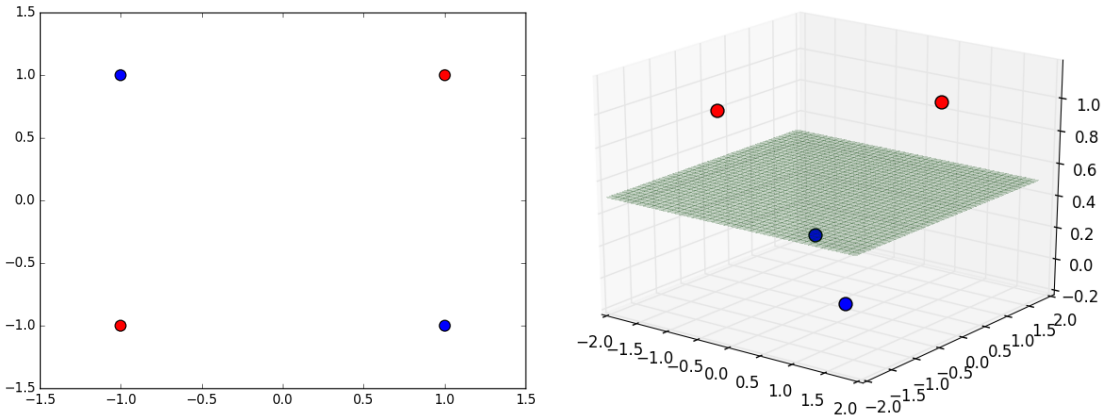
4.4.2.3.5 Support Vector Machine

Az SVM (Support Vector Machine) a tanítóhalmaz pontjait elhelyezi egy sokdimenziós térben, valamint definiál egy olyan hipersíkot, amely egyrészt a pontokat két halmazba osztja aszerint, hogy a hipersík által meghatározott félterek melyikében helyezkednek el, másrészt a „lehető legszélesebb űrt" (margót) hagyja a ponthalmazok között, azaz amelytől a két ponthalmaz távolsága maximális. Ezután az osztályozás triviális módon történik: a minta annak a ponthalmaznak a címkéjét kapja, amelyik félterébe esik.

Kiindulásképp a dimenziók száma megegyezik a független változók (attribútumok, oszlopok) számával. A keresett hipersík ebben az esetben pontosan a már a döntési fáknál tárgyalt lineáris szeparációt jelenti. Azonban – ahogy erre a döntési fáknál is felhívtam a figyelmet – gyakran előfordul, hogy a pontok ebben a térben nem választhatók szét lineárisan. Ekkor az SVM algoritmus addig növeli a dimenziószámot, amíg a két ponthalmaz szétválaszthatóvá válik.

Ezt az eljárást szemlélteti az alábbi egyszerű példa:

X	Y	Célváltozó
1	1	1
1	-1	0
-1	1	0
-1	-1	1

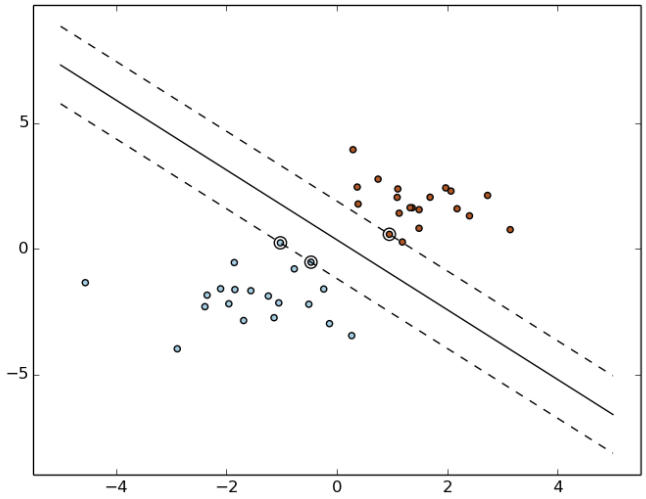


4.15. ábra Példa lineáris szeparálhatóságra a dimenziószám növelésével

Látható, hogy ez a négy pont nem szeparálható lineárisan: a síkba nem helyezhető el olyan egyenes, hogy a pozitív és negatív pontok halmaza az egyenes két oldalára essen. Eggyel (háromra) növelve a dimanziószámot azonban már megoldható a feladat, ahogy azt a 4.15. ábra jobb oldala mutatja.

A dimenziószám növelésével viszont problémát jelent a számításigény drasztikus növekedése. Ezt a problémát oldja meg a „kernel trükk”: ismeretes, hogy egy hipersík definiálható egy olyan ponthalmazként az n dimenziós térben, amely halmaz minden pontjára igaz, hogy egy vektorral képzett skaláris szorzata konstans, ez a vektor pedig képezhető a minták lineáris kombinációjaként. A trükk lényege, hogy ezek a skaláris szorzatok számolhatók kernel függvényekkel, így magukat a koordinátákat, valamint azok távolságát nem kell meghatároznunk.

A 4.16. ábra mutatja a szeparálás eredményét: a folytonos vonal jelöli a szétválasztó hipersíkot, a szaggatott vonalak pedig a margót – két olyan párhuzamos hipersíkot, melyeken már található egy-egy pont a két halmazból (ezeket a pontokat nevezzük support vektoroknak).



4.16. ábra Maximális margójú hipersík [36]

4.4.2.4 Implementáció

A 4.3.3. pontban már említettem, hogy az előkészített adatok exportálása lényegében bármilyen osztályozási keretrendszer használatát lehetővé teszi. Ezt kihasználva két egymástól független implementációt hoztam létre.

4.4.2.4.1 Scikit-learn, Python

Az offline (exportált adatokkal történő) tanulást és osztályozást az „scikit-learn" [29] nevű open source, python nyelvű gépi tanulás keretrendszerrel valósítottam meg. A keretrendszer előnye, hogy egyszerű – kényelmesen és gyorsan prototipizálható benne lényegében bármilyen feladat, de hatékony – mind a végrehajtási sebesség (az osztályozó algoritmusok egy része C nyelven van implementálva), mind az osztályozók teljesítménye szempontjából. Ezen felül szoros együttműködésre képes a „pandas" [37] nevű statisztikai csomaggal, mely az adatok áttekintésében, vizualizálásában és előkészítésében nyújt segítséget.

4.4.2.4.2 Accord.NET, C#

A 2.3. fejezetben bemutatott virtuális gép alkalmas python scriptek futtatására, mégis kipróbáltam egy közvetlenül hívható megoldást is: .NET környezetben az Accord.NET [30] keretrendszerrel valósítottam meg a feladatot.

A tapasztalataim alapján ez is egyszerű fejlesztést és hatékony végrehajtást biztosít, azonban – legalábbis jelen esetben – mindkét szempontból elmarad a python alapú megoldással szemben: a fejlesztés valamivel lassabb, a kód (leginkább a nyelvi különbségekből fakadóan) hosszabb és kevésbé jól olvasható, a futásidő kicsit hosszabb

és az – ugyanúgy parametrizált, ugyanazokon az adatokkal tesztelt – osztályozók hatékonysága is alacsonyabb.

létrehozás	sklearn	`svm.SVC(probability=True)`

	Accord.NET	`var svm =`
		`  new KernelSupportVectorMachine(`
		`  Gaussian.Estimate(teachSet),`
		`var svmModel =`
		`  new SequentialMinimalOptimization(`
		`  svm, inputs, svmOutputs)`
		`{`
		`  UseComplexityHeuristic = true`
		`};`

tanítás	sklearn	`model.fit(teach_set[x], teach_set.target)`
	Accord.NET	`svmModel.Run();`
osztályozás	sklearn	`m.predict_proba(validation[x])`
	Accord.NET	`var result = validation.Apply(svm.Compute)`

4.4. táblázat Kódrészletek az implementációkból

5 Értékelés

5.1 A modellezés hatékonysága

A modellek létrehozása után részletesen megvizsgáltam azok hatékonyságát. Természetesen a vizsgálatok eredményei alapján módosítottam, finomhangoltam, majd újravalidáltam őket. Ezt a lépéssorozatot – szükség esetén egészen az adatelőkészítési fázisig visszamenően – a CRISP-DM modellnek megfelelően többször elvégeztem.

A továbbiakban az elkészített osztályozók jelenlegi teljesítményét fogom különböző mérőszámokon keresztül bemutatni.

5.1.1 Mérőszámok

Bináris osztályozók teljesítményének mérésére számos mérce létezik.

5.1.1.1 Tévesztési mátrix

A tévesztési mátrix (lásd 5.1. ábra) táblázatos formában vizualizálja az osztályozó hatékonyságát: a találatait és tévesztéseit.

		A célváltozó valódi értéke	
		Pozitív (P)	Negatív (N)
A predikátum	Pozitív	True Positive (TP)	False Positive (FP, elsőfajú hiba)
	Negatív	False Negative (FN, másodfajú hiba)	True Negative (TN)

5.1. ábra A tévesztési mátrix általános definíciója

A tévesztési mátrix mezőiből a következő mérőszámok vezethetők le:

- TPR (True Positive Rate, sensitivity)

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- TNR (True Negative Rate, specificity)

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP}$$

- FPR (False Positive Rate, fall-out)

$$FPR = \frac{FP}{N} = \frac{FP}{TN + FP} = 1 - TNR$$

- FNR (False Negative Rate)

$$FNR = \frac{FN}{P} = \frac{FN}{TP + FN} = 1 - TNR$$

- pontosság (accuracy)

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + FN + TN + FP}$$

A jelen probléma kontextusában a tévesztési mátrix a következő értékeket tartalmazza:

		A játékmenet valódi minősítése	
		Érvényes (P)	Csalás (N)
A predikátum	Érvényes	Helyesen érvényesnek becsült játékmenet	Helytelenül érvényesnek becsült játékmenet
	Csalás	Helytelenül érvényesnek becsült játékmenet	Helyesen csalásnak becsült játékmenet

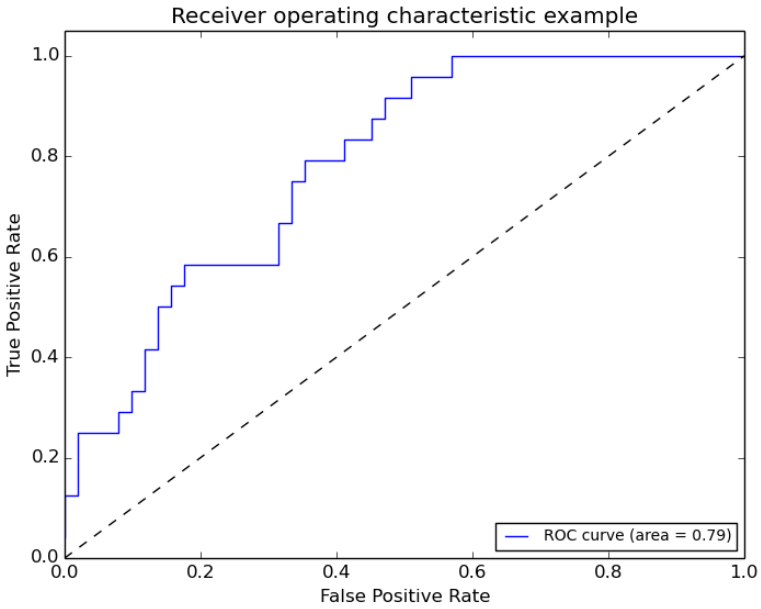
5.2. ábra A csalásdetektáló algoritmus általános tévesztési mátrixa

5.1.1.2 ROC görbe

Az imént felsorolt mérőszámok között minden teszt esetén egy trade-off figyelhető meg a bináris osztályozó döntési beállításaitól függően: ha például egy osztályozó minden mintát elfogad, akkor a TPR 100% lesz, a TNR viszont 0%, ha pedig mindent elutasít, akkor fordítva. A két szélsőséges eset között azt mondhatjuk, hogy akkor fogadunk el egy mintát, ha az osztályozó egy adott k konfidenciaszintnél nagyobb valószínűséggel prediktálja pozitívnak. k küszöbértéket növelve a TPR csökken, a pedig TNR nő – hiszen csak magasabb konfidencia mellett fogadunk el valamit, vagyis a pozitív predikátumok (így a TP-ok száma) csökken, csökkentve pedig fordítva.

Ezt a trade-offot jeleníti meg a ROC (Receiver Operating Characteristic) görbe, amelyre az 5.3. ábra mutat példát. A görbe a TPR-FPR pontpárokat jeleníti meg a k küszöb függvényében ($k \in [0, 1]$). A hatékony generálásához növekvő sorrendbe rendezzük a predikátum valószínűségeket, és k-val csak ezeken iterálunk végig a

generált pontok összekötésével (hiszen más k értékekre úgysem fog különböző értéket felvenni a függvény).



5.3. ábra ROC görbe [38]

5.1.1.3 AUC

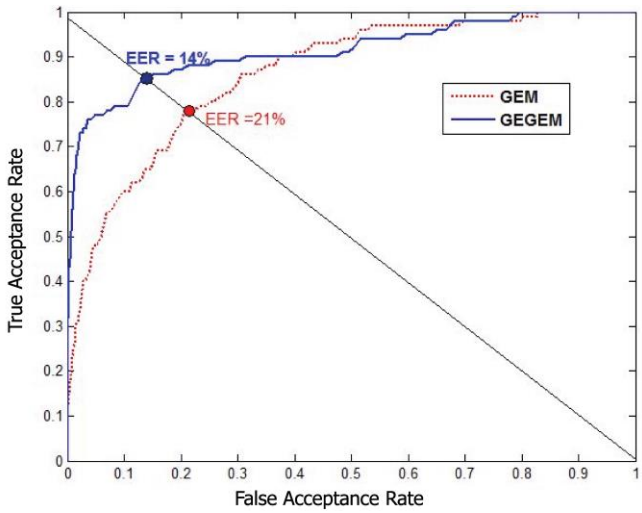
Az eddig leírtak alapján világos, hogy a fenti mérőszámok egyike sem tükrözi pontosan az osztályozó hatékonyságát. Szükséges tehát további mérőszámok bevezetése.

Az AUC (Area Under the Curve) a ROC-görbe alatti terület, vagyis a görbe határozott integrálja. Ezt a mérőszámot már alkalmasnak tekintjük az osztályozó értékelésére, valamint különböző osztályozók összehasonlítására [39].

5.1.1.4 EER

Az Equal Error Rate (vagy Crossover Error Rate) az az arány, amikor az első- és másodfajú hibaértékek megegyeznek, azaz $FPR = FNR$. Ez az érték is alkalmas lehet osztályozók összehasonlítására. A ROC görbéről könnyen leolvasható (lásd 5.4. ábra), mivel $FNR = 1 - TPR$.

Magától értetődő, hogy az AUC esetében a magasabb (maximálisan 1), EER esetében az alacsonyabb (ideális esetben 0) értékek a jobbak.



5.4. ábra Az EER a ROC-görbén [40]

5.1.2 Minta

Az osztályozást egy 54 pozitív (érvényes) és 270 negatív (csalás) megfigyelésből álló mintán végeztem el. A minta alapjául szolgáló játékmenetek mind pedagógus felügyeletével és a Korongház alkalmazás Dobozolás játékmódjával történtek. A mintahalmaz összesen 1145 swipe-ot tartalmazott.

Az osztályozás értékeléséhez keresztvalidációt használtam. A keresztvalidáció lényege, hogy a teljes mintahalmaz egy részéből képezzük a tanító halmazt, erre építünk modellt, majd a kapott modellt a mintahalmaz másik (diszjunkt) részén futtatjuk. Ez a validációs fázis: mivel a validációs halmaz rekordjainak is ismerjük a címkéjét, meg tudjuk állapítani, hogy az osztályozó milyen hatékonysággal működik – legalábbis azon a halmazon. Hogy az ebből adódó bizonytalansági tényezőt kiszűrjük, a keresztvalidációt többször végezzük el, és az összesített eredményt értékeljük.

5.1.3 DTW

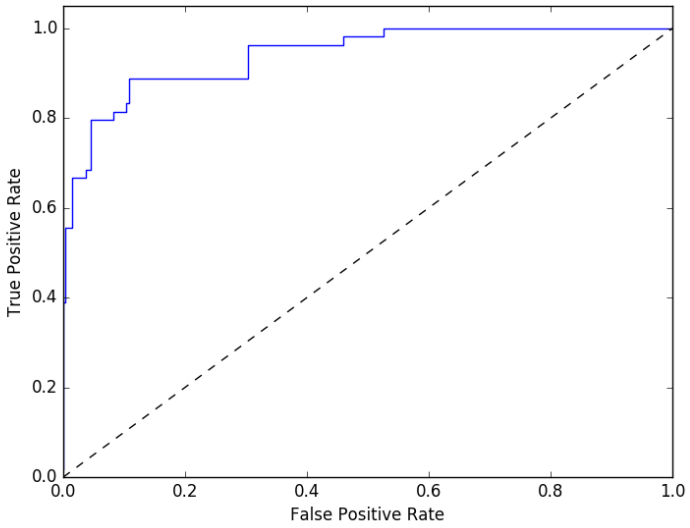
Itt a 4.4.1. pontban bemutatott megoldás hatékonyságát fogom ismertetni.

Az első módszer (4.4.1.3.1) – amikor minden swipe-ról a hozzá leghasonlóbb swipe címkéje alapján döntünk – tévesztési mátrixa (k = 0.21 esetén) a következő:

TP = 45	FN = 29
FP = 9	TN = 241
TPR = 83,33%	TNR: 89,26%
ACC = 88,27%	

5.1. táblázat Az érintéssorozatokat összehasonlító (1. típusú) algoritmus tévesztési mátrixa

Ez azt jelenti, hogy az algoritmus – ilyen paraméterekkel – az 54 pozitív mintából 45-öt azonosított helyesen (83,33%), míg a 270 negatívból 241-et (89.26%), azaz összesen 324-ből 286-ot (88,27%). Ezek az értékek éreztetik az így beállított bináris osztályozó hatékonyságát, ám pontos minősítésre és összehasonlításra (mint már kifejtettem) nem alkalmasak, ezért a vizsgálatot a ROC-görbe generálásával és az AUC (valamint EER) értékek kiszámításával folytattam.



5.5. ábra A leghasonlóbb swipe alapú algoritmus ROC-görbéje

Az 5.5. ábra ROC-görbe alakján „ránézésre” látszik, hogy az osztályozó hatékony, mert meredeken emelkedik. Ezt támasztja alá az AUC érték is:

$$AUC = 0{,}9317,$$

ami kiváló (A) minősítésnek felel meg.

Az Equal Error Rate – az a hibaarány, amikor a pozitív és negatív tévesztések aránya megegyezik – 0,114 (ez közel van a fenti tévesztési mátrix állapotához).

A második módszer vizsgálatához szükség van egy küszöbértékre: vagyis hogy milyen (legkisebb) távolság esetén fogadjunk, illetve utasítsunk el egy érintéssorozatot. A küszöbérték pontos meghatározása kulcsfontosságú az algoritmus optimális működéséhez, ezért nem érdemes véletlen, vagy intuíció alapján választani.

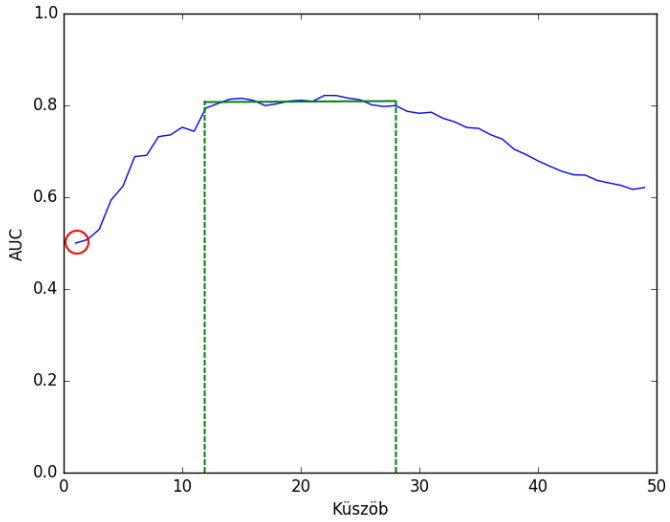
Első lépésként megvizsgáltam, hogy a DTW algoritmussal kiszámolt távolságokat. A következő táblázat ezek leíró statisztikáját tartalmazza:

Karakterisztika	Érték
átlag	36.820633
szórás	28.638870
minimum	1.071473
Q1 (első kvartilis)	20.314137
Q2 (medián)	29.844818
Q3	42.339862
maximum	310.506837

5.2. táblázat A legközelebbi távolságok leíró statisztikája

Ez alapján már nagyságrendileg elhelyezhetők a potenciális küszöbértékek: mindenképpen száznál (de valószínűleg Q3-nál, azaz ~42-nél) is érdemes kisebb értéket választani. Az is fontos észrevétel, hogy nem érdemes „túlságosan pontosan” (pl. 2 tizedes jegy pontossággal) meghatározni a küszöböt, hiszen más-más játékmeneteket tartalmazó adathalmazokon nyilvánvalóan valamennyire különböző érték az ideális.

Mindezen megfontolásokat figyelembe véve lefuttattam és keresztvalidáltam az algoritmust a mintán 1 és 50 közötti (egész) küszöbértékekkel. A hatékonyságot az AUC értékkel mérve az eredményt az 5.6. ábra vizualizálja:



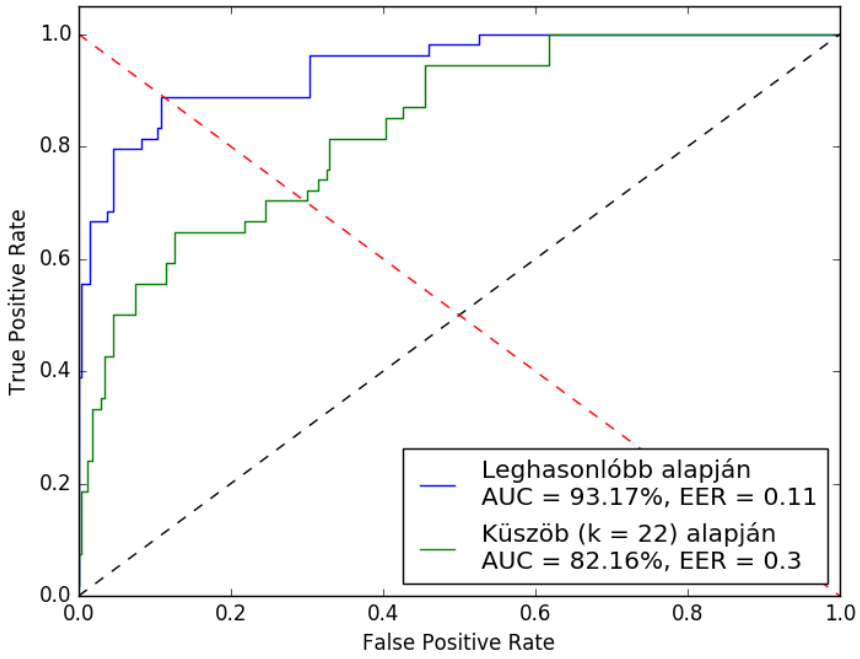
5.6. ábra A küszöb alapú algoritmus AUC értékei a küszöb függvényében

Az ábrán jól megfigyelhetők az alábbiak:

- A kék színnel jelölt a küszöb → AUC függvény a zölddel berajzolt szakasz elejéig növekvő, azon tendáló, majd csökkenő trendet mutat, vagyis az osztályozó hatékonysága a küszöb növelésével egy ideig nő, majd egy idő után csökken.

- A zöld szakasz tehát az az intervallum, amelyen belül választott küszöbértékekkel az algoritmus maximumközeli hatékonysággal működik (az adott adathalmazon). Látható, hogy ez az intervallum meglehetősen hosszú: ebből egyrészt az következik, hogy az osztályozó viszonylag jól tolerálja a küszöb kis változtatását, másrészt feltehetően más adathalmazon is hasonlóan fog működni.
- A pirossal kiemelt 1-es küszöbértékre az AUC értéke körülbelül 0,5, azaz az osztályozó hatékonysága kvázi a véletlen becslés (pénzfeldobás) hatékonyságával egyenlő – ezt közvetlenül magyarázza, az 5.2. táblázat minimum sora.

Végül összehasonlítottam a két implementált algoritmust: az 5.7. ábra egy diagramon mutatja a ROC-görbéiket (a küszöb alapú módszer esetében *k*-t 22-nek választottam).



5.7. ábra A DTW alapú algoritmusok ROC-görbéi

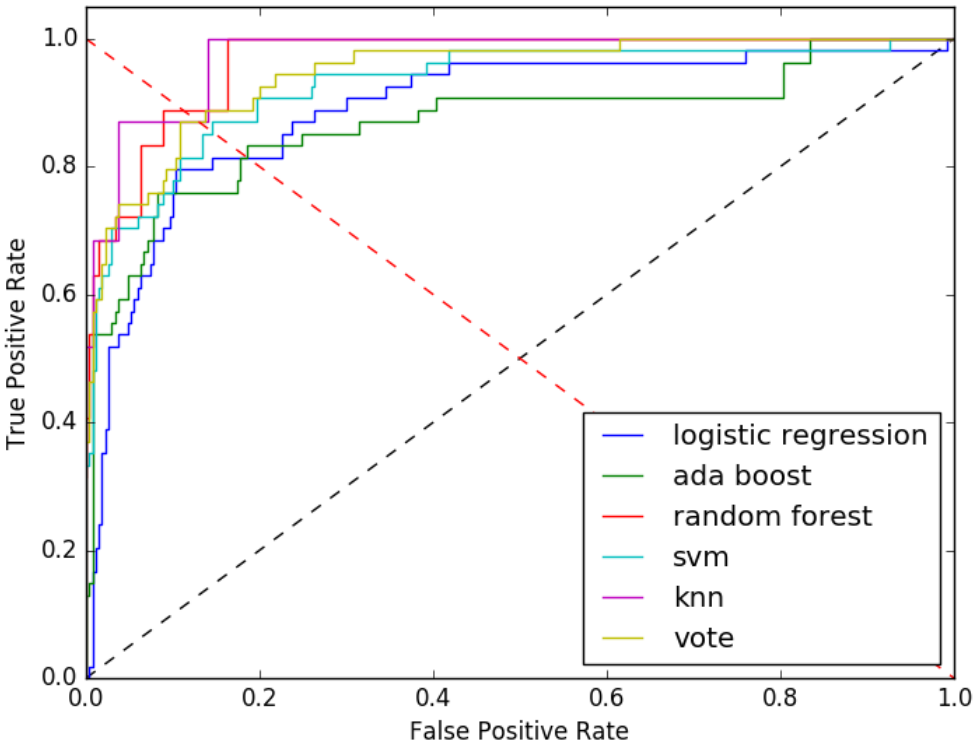
Jól látható, hogy az 1. módszer lényegesen hatékonyabb – a görbéje végig a 2. módszeré fölött van, azaz minden False Positive ráta mellett magasabb True Positive arányt tud elérni. Ezt természetesen az AUC és EER értékek is alátámasztják.

5.1.4 Osztályozók

Ebben az alfejezetben a probléma másik általam megvalósított megoldásának eredményeit, a különböző osztályozó algoritmusok hatékonyságának vizsgálatát fogom bemutatni.

A 4.4.2.2. pontban kifejtettem, hogy a tanítóhalmaz kialakításától függően két alapvetően különböző osztályozó algoritmust készítettem: a „teljes tanító halmaz” módszer a keresztvalidáció tanító halmazának minden elemére egyszerre épít modellt, míg a „páronkénti tanító halmaz” módszer felhasználóként felosztja a teljes halmazt, és mindegyikre külön-külön épít egy-egy modellt, végül az eredményeket aggregálja (4.10. ábra).

5.1.4.1 Teljes tanító halmaz



Osztályozó	AUC (%)	Minősítés	EER
logisztikus regresszió	89,12%	B	0,185
AdaBoost	86,68%	B	0,185
Random Forest	95,45%	A	0,111
SVM	92,82%	A	0,144
kNN	90,40%	A	0,129
szavazás	94,81%	A	0,129

5.8. ábra Különböző osztályozók ROC-görbéje, AUC és EER értékei a teljes tanítóhalmaz esetén

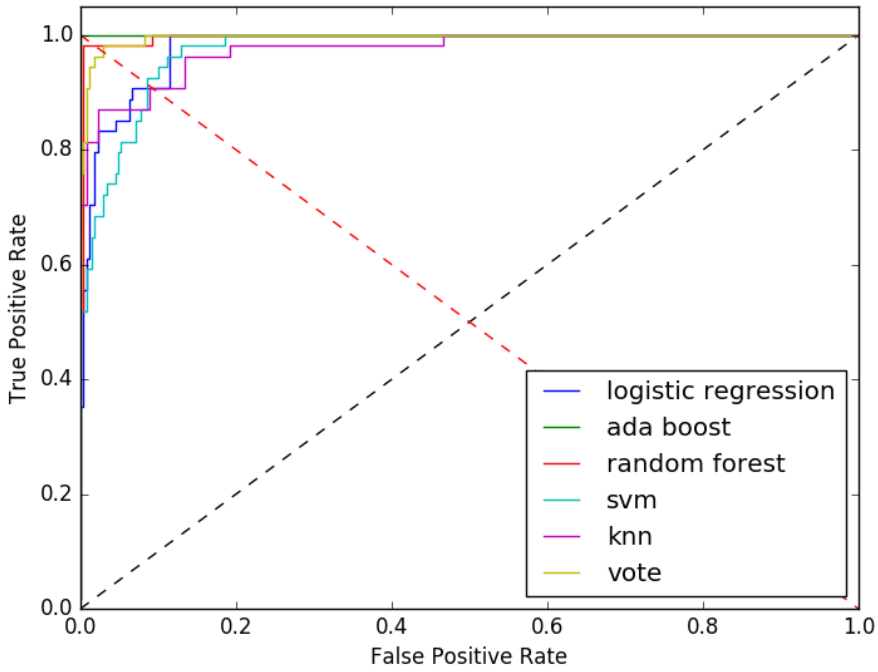
Az 5.8. ábra az osztályozók ROC-görbéit, AUC és EER értékeit mutatja abban az esetben, ha egy közös modellt építünk a teljes tanítóhalmazra. Mind az öt a 4.4.2.3. pontban bemutatott osztályozó algoritmust kipróbáltam (a táblázat első öt sora), valamint egy olyan együttes osztályozót („szavazás”), amely lefuttatja ezeket az osztályozókat, és a predikciók átlaga alapján dönt.

Jól látszik, hogy a két legjobb eredményt a Random Forest, és ez utóbbi együttes osztályozó érték el 94-95% körüli AUC és 0.11-0.13 körül EER értékekkel. Megjegyzem, hogy ezek a számok a mintahalmaz, valamint a keresztvalidáció, illetve az algoritmusok véletlen faktorai függvényében a mérések többszöri elvégzésekor némi szórást mutatnak – volt például olyan mérés, ahol az együttes osztályozó AUC értéke nagyobb volt a Random Foresténél.

Érdemes még meggondolni, hogy vajon a két „B” minősítésű osztályozó növeli, vagy csökkenti az együttes osztályozó („szavazás”) hatékonyságát. A méréseim azt mutatták, hogy – ebben a konkrét esetben, ezen a halmazon – csökkentik, a három legjobb osztályozóból (Random Forest, SVM, kNN) álló metatanuló mérőszámai jobbak:

Osztályozó	AUC (%)	Minősítés	EER
szavazás (3)	96,01%	A	0,11

5.1.4.2 Páronkénti tanító halmaz



Osztályozó	AUC (%)	Minősítés	EER
logisztikus regresszió	98,02%	A	0,092
AdaBoost	100%	A	0
Random Forest	99,57%	A	0,018
SVM	97,41%	A	0,084
kNN	96,81%	A	0,092
szavazás	99,63%	A	0,029

5.9. ábra Különböző osztályozók ROC-görbéje, AUC és EER értékei páronkénti tanítóhalmaz esetén

Az előző méréssorozatot megismételtem a második, páronkénti tanító halmazképző technikával, az eredmény a fenti ábrán látható. Világosan látszik, hogy ezt a módszert alkalmazva minden algoritmus lényegesen jobban teljesített. Viszont a jelentős hatékonyságjavulás mellett nem szabad megfeledkeznünk a számításigény növekedéséről sem: ebben az esetben az algoritmus 1 helyett a felhasználók számával arányos darab modellt készített (ezt részletesen és mérésekkel alátámasztva a következő pontban fogom kifejteni).

A (ROC-görbén zöld színnel jelölt) AdaBoost osztályozó ebben a konstellációban 100%-os AUC értéket ért el, azaz „tökéletes osztályozónak” tekinthető. Ez az érték azonban egyrészt a mérések során a már említett véletlen faktorokból kifolyólag szintén rezgett (pár ezreléknyit), másrészt korántsem jelenti azt, hogy az osztályozó a feladatot tökéletesen oldaná meg, azaz minden bemenetre helyes címkét prediktálna. Az 1-es AUC érték (az AUC és a ROC-görbe definíciója szerint) azt jelenti, hogy a validált rekordokat az AdaBoost konfidencia értékei (pontszámai) alapján sorba rendezve létezik egy olyan vágási pont (küszöb), amely fölött csak pozitív, alatt pedig csak negatív megfigyelések vannak. Azonban ez a küszöb – mind a létezése, mind a pontos értéke – minta specifikus, tehát amikor gyakorlatban használjuk az algoritmust (vagyis nem ismerjük a valódi címkét és a küszöböt manuálisan kell – pl. a tanítóhalmaz alapján – beállítanunk), valószínűleg nem fog hibátlanul működni: a küszöb környezetében néhány százaléknyi hiba előfordulhat.

Fontos még megjegyezni, hogy a bemutatott eredményt az osztályozók az 5.1.2. pontban bemutatott mintán produkálták. Ez a minta alkalmas az algoritmusok tesztelésére: mind sokszínűségéből, mind a méretéből adódóan, feltételezhető viszont, hogy lényegesen nagyobb minta esetén az osztályozók némileg – de valószínűleg nem nagymértékben – más hatékonysággal működnének. Ennek a vizsgálata a közeljövőre tervezett kutatás feladata: egy kellően nagy minta összegyűjtése már folyamatban van –

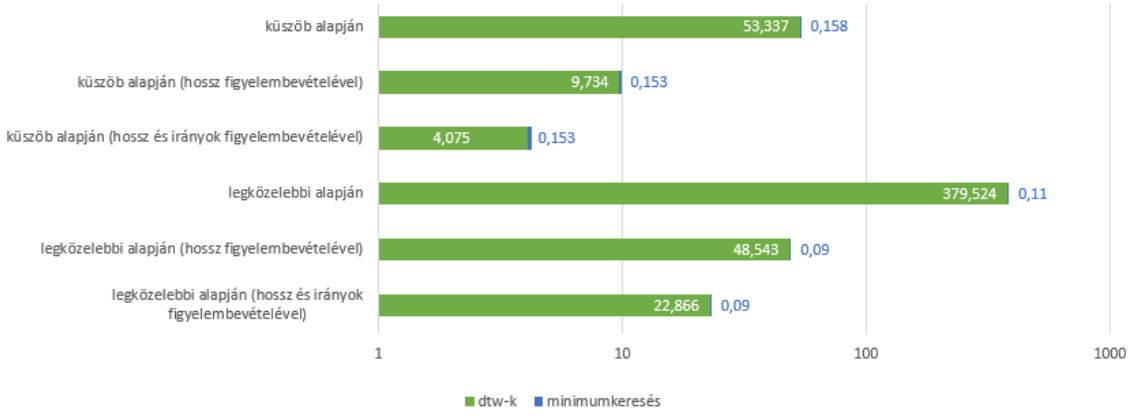
az oktatójátékok a tesztfázisból mostanában kerülnek graduálisan bevezetésre az iskolákban, valamint néhány már a Play Store-ban is elérhető [41].

Végül megemlítem, hogy az elért 99% fölötti eredmény a szakirodalomban nem egyedülálló: a SilentSense projekt szintén 1% alatti hibaaránnyal tudta a felhasználókat azonosítani [23].

5.2 Futásidő

A futásidők mérését a hatékonyságok vizsgálatának megfelelő csoportosítás szerint végeztem el. A méréseket igyekeztem az algoritmusok valós környezetéhez illeszteni, ezért nem a keresztvalidáció futásidejét mértem, hanem a mintából kiválasztott 20 elemű játékmenet halmaz osztályozásának végrehajtási idejét.

Az alábbi ábrán a DTW-alapú algoritmusokra vonatkozó mérések eredménye – a számértékek másodpercben (milliszekundum pontossággal) értendők.



5.10. ábra A swipe-távolság alapú algoritmusok összehasonlítása

Mindenekelőtt felhívom a figyelmet arra, hogy a sávdiagram (a dolgozatban egyedüliként) az x tengelyen logaritmikus skálát használ. Erre az eredmények közötti nagyságrendi különbségek miatt van szükség: lineáris skálán csak a legnagyobb érték látszódna.

Világosan látszik, hogy az algoritmus futásidejének nagy részét a távolságok kiszámítása alkotja: ezután a minimumkeresés lényegében elhanyagolható. A DTW-algoritmus futásideje a 10 és 100 milliszekundumos határok között mozgott, tehát a teljes végrehajtási időt elsősorban az határozta meg, hogy ennek az algoritmusnak hányszor kellett lefutnia. Egyértelmű, hogy ez az érték a küszöb alapú módszernél alacsonyabb, hiszen ott a bemeneti játékmenet érintéseit az adatbázisban levő érintések

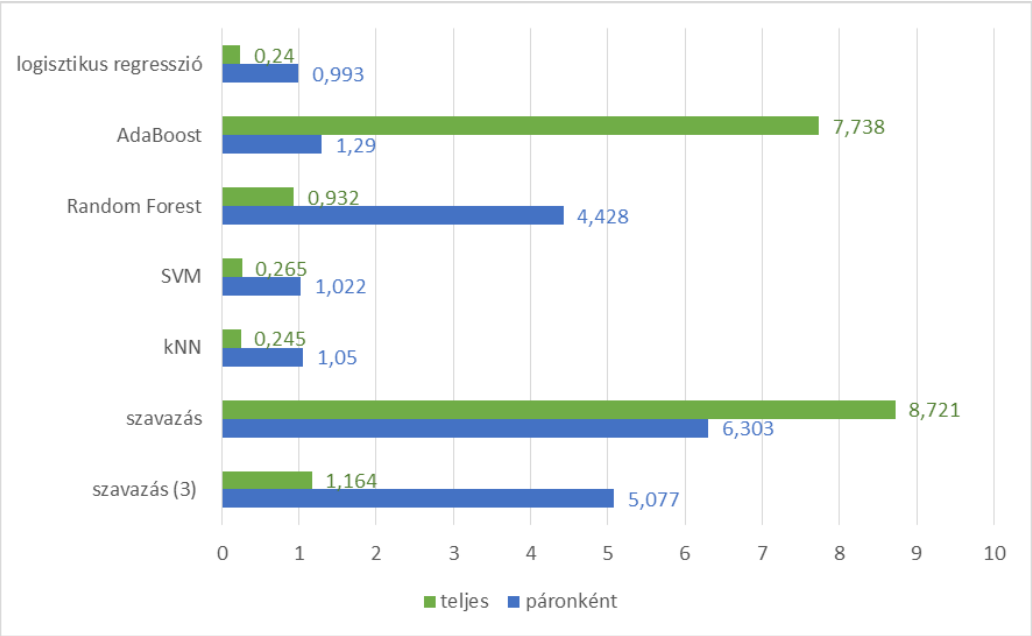
közül csak azokkal kell összehasonlítani, amelyek a bemenetnek megfelelő felhasználóhoz tartoznak.

Bár már többször hangsúlyoztam, hogy a projekt során nem prioritás az osztályozás sebessége, az algoritmusok optimalizáció nélküli változatával mért eredmények (~1 illetve 6 perc) már-már használhatatlanul lassúnak tűnnek. Hogy csökkentsem a futásidőt kipróbáltam a TIPS projektben már bevált módszert [25]: az adatbázisban levő érintéspárok előszűrését. Az optimalizálás lényege, hogy a DTW algoritmust csak olyan párokra futtassuk ténylegesen le, amelyek potenciálisan közel vannak egymáshoz. Ehhez én két faktort vettem figyelembe: a swipe hosszát (vagyis a sorozat hosszát) és irányát (balról jobbra, fentről le). Ezek az értékek nagyon gyorsan számolhatók, és az általuk történő előszűréssel láthatóan drasztikusan (töredékére) csökken a végrehajtási időt.

Felmerülhet a kérdés, hogy ezért cserébe nem gyengül-e az algoritmus, azaz nem szűrünk-e ki véletlenül olyan swipe-okat, amelyek valójában mégis a legközelebb lettek volna a bemenethez. Megvizsgáltam a kérdést, és azt az eredményt kaptam, hogy az algoritmusok hatékonysága lényegében nem változik. Ennek az az oka, hogy a legkisebb távolságok a vizsgált párok 91%-ában nem is változnak – tehát a legközelebbi érintést nem szűrjük ki, a maradék 9%-ban pedig csak minimálisan nő a legkisebb távolság: a vizsgált mintán átlagosan 2,34-et. (Ez az érték kellően kicsi, ennek a belátásához érdemes összevetni az 5.2. táblázattal, valamint az 5.6. ábra optimális küszöbét jelző zöld szakaszának hosszával).

Most – az előző pont struktúrájának megfelelően – áttérek a különböző osztályozók futásidejének vizsgálatára. A mért eredményeket (a következő oldalon) az 5.11. ábra tartalmazza.

Megfigyelhető, hogy a legtöbb esetben a páronkénti megközelítés a lassabb – a már említett okból: több modellt kell építeni. Ez alól látványos kivétel az AdaBoost algoritmus, ami annak belső működésével magyarázható: ez az együttes osztályozó parametrizálható számú (az aktuális beállításokkal 100 db) „gyenge osztályozóval” modellez, és úgy tűnik, hogy ezek futásideje erősen függ a tanítóhalmaz nagyságától. Páronkénti tanítás esetén ez a halmaz lényegesen kisebb (egy-egy pár esetén), míg a teljes halmaz felhasználása esetén nagyobb, és az ebből fakadó különbség szignifikánsabb, mint a páronkénti tanítás külső iterációja (amely jó közelítéssel egy szorzótényezőt jelent).



5.11. ábra Osztályozók futásidejének összehasonlítása

A legrövidebb futásidővel mindkét esetben a logisztikus regresszió, az SVM és a kNN hármas rendelkezik. Ezek közül az előző alfejezetben láttuk, hogy az SVM bizonyult a leghatékonyabbnak, tehát ha a feladat időszenzitív lenne, érdemes lenne ezt az osztályozót választani.

Az öt egyszerű osztályozó közül a páronkénti esetben a Random Forest, a teljes esetben az AdaBoost volt kiemelkedően a leglassabb, ráadásul ilyen kombinációban nem is értek el jó eredményt, tehát ilyen tanító halmaz képző módszerekkel semmiképp nem tekinthetők kedvező választásnak. Fordítva viszont ezek voltak a leghatékonyabb egyszerű osztályozók, tehát az, hogy a futásidejük nagyobb a másik háromnál megfelelő trade-off lehet.

Ami az együttes osztályozót illeti, ez a mérés is azt mutatja, hogy érdemes csak a három legjobb egyszerű osztályozót felhasználni a Bagging-hez.

5.3 Összefoglalás

Összefoglalva a két alfejezetet a következő megállapításokra jutottam:

- Az érintés aggregátumok, valamint az egyéb játékmenetek alapján történő osztályozás mind hatékonyság, mind teljesítmény tekintetében túlszárnyalja az érintéssorozatok összehasonlításán alapuló

algoritmusokat (még egyszer hangsúlyozom, hogy ezen a mintán), tehát a következő kutatásokban érdemes ezekre fokuszálni.

- A Business Understanding során kitűzött adatelemzési célt minden implementált algoritmus teljesíti.
- A mérések alapján az idő komponens priorizálásától függően a következő algoritmusokat tartjuk legjobbnak:
    - minimális idő prioritás esetén: AdaBoost vagy együttes osztályozó páronkénti tanító halmaz képzéssel
    - közepes idő prioritás esetén: Random Forest vagy metatanuló a teljes tanító halmazon, vagy SVM páronkénti tanító halmaz képzéssel
    - maximális idő prioritás esetén: SVM a teljes tanító halmazon

5.4 Evaluation

A CRISP-DM következő fázisa az adatelemző modellek és az adatelemzési cél teljesítésének vizsgálata utána az üzleti cél (Business Success Criteria) teljesítésének kiértékelése.

Mivel itt nem üzleti jellegű feladatról van szó, hanem a projekt elsősorban kutatás jellegű, az üzleti és adatelemzési célok jelen esetben hasonlónak tekinthetők, ezért ettől a kiértékeléstől itt eltekintünk, vagy más megközelítéssel az előző alfejezet az Evaluation fázis részévé tehető.

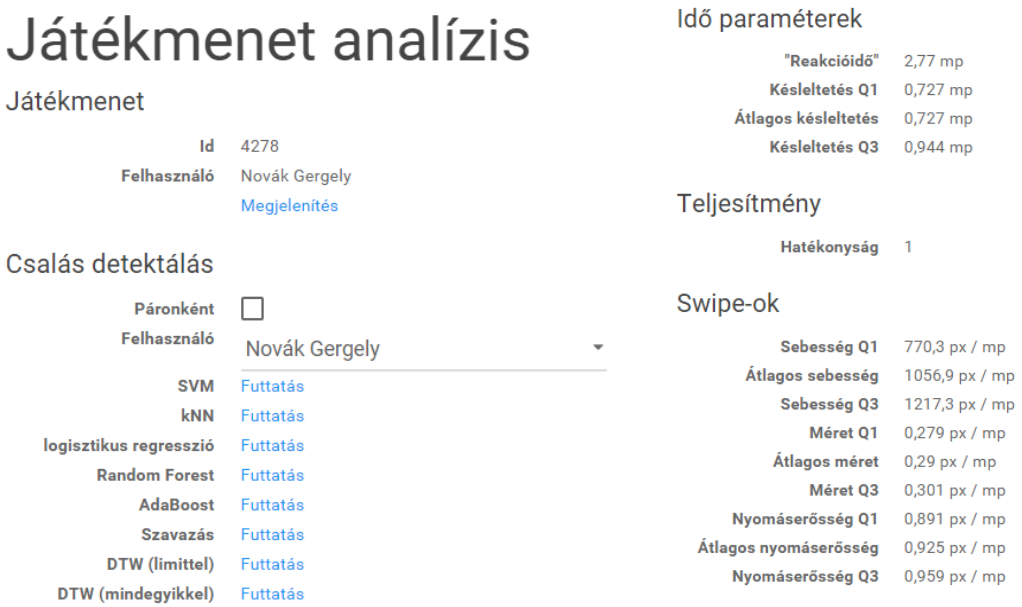
5.5 Deployment

A hivatalos folyamatmodell utolsó fázisa az eredmények beépítése az üzleti folyamatba. Ez a motiváció során megfogalmazott két igény kielégítésével történhet: egyrészt a pedagógusok tájékoztatásával, másrészt a későbbi adatelemzési feladatok során a Data Preparation fázisban levő adattisztítás megvalósításával.

A tanárok tájékoztatása érdekében a webalkalmazást kiegészítettem egy „Csalásdetektálás” oldallal, amely az 5.12. ábra képernyőképein tekinthető meg.

Jobb oldalt az adatelőkészítés utáni aggregátumok vannak felsorolva, bal oldalt pedig lefuttathatók a már bemutatott algoritmusok. A legördülő listából kiválasztható

egy tanuló, és az osztályozók azt becsülik meg, hogy ez a tanuló hajtotta-e végre a játékmenetet. Alapértelmezésben a játékmenethez bejelentkezett felhasználó van kiválasztva: egy ilyen ellenőrzés felel meg az eredetileg kiírt feladatnak. Egy másik tanulót kijelölve egyrészt meggyőződhetünk az algoritmusok helyességéről (vagyis hogy ez esetben csalásnak minősítik a játékmenetet), másrészt csalás esetén megfordíthatjuk a kérdést: ha az osztályozók nem fogadták el a bejelentkezett felhasználót, megpróbálhatjuk megkeresni, hogy a többiek közül kié lehet a játékmenet (természetesen előfordulhat, hogy nem találunk párosítást, mert például egy tanuló szülei játszottak az alkalmazással).

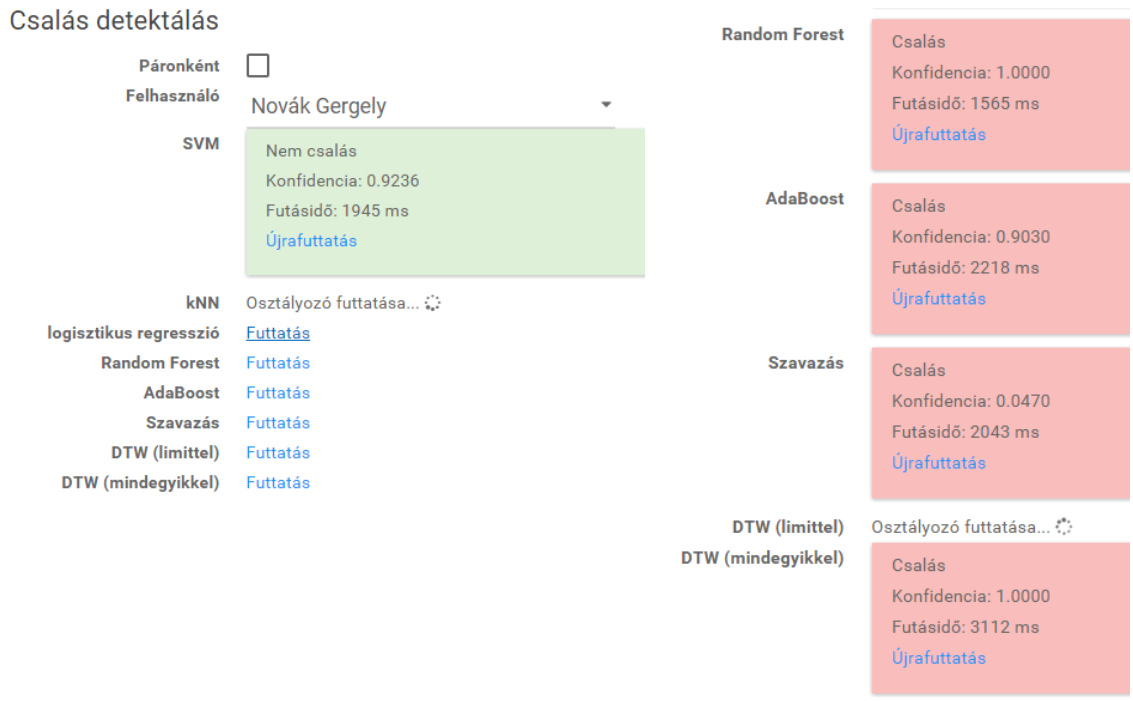


5.12. ábra Részletek a "Játékmenet analízis" oldalról

A „Futtatás” gombok megnyomásakor a rendszer végrehajtja az osztályozást. Mivel ez időigényesebb folyamat is lehet (például a DTW-alapú algoritmusoknál) ez a funkció egy AJAX hívással történik: a futást a felületen egy animált töltőikon mutatja, majd a hívás visszatérésekor megjelenik az eredmény.

Az AJAX hívást kiszolgáló Controller továbbhív az `AdaptEd.FraudDetection` könyvtárban található `GamePlayAnalyzer` osztályba, ami a tényleges osztályozást végzi. A tisztán C# nyelvű modulok esetén (DTW, Accord.NET) ez egy egyszerű függvényhívást jelent, míg a python nyelvű moduloknál egy folyamat futtatását. A függvényparaméterek (játékmenet, osztályozó típusa, vizsgált felhasználó, stb.) átadása ez esetben parancssori argumentumokkal történik. Ezt követően a script csatlakozik az MS SQL adatbázishoz (a pypyodbc [42] könyvtár segítségével), lekérdezi a tanító

halmazt, osztályozza a kapott játékmenetet és az eredményt a standard output átirányításával küldi vissza a `GamePlayAnalyzer`nek.



5.13. ábra "Játékmenet analízis" oldal: AJAX hívás, eredmények megjelenítése

Az eredmény a felületen szövegesen és színkóddal is megjelenik (lásd 5.13. ábra): piros jelöli a csalást, emellett a konfidencia értékek, valamint a futásidők is láthatóak. A jobb oldali képen például az összes osztályozó (helyesen) csalásnak minősíti játékmenetet a kiválasztott felhasználóval.

5.5.1 További tervek

A további tervek közé tartozik a csalásdetektálás fordított irányú megközelítésének megvalósítása: azaz, hogy az alkalmazás ne csak a pedagógus által kiválasztott játékmeneteket osztályozza, hanem minden (éles) játékmenetet, majd a „leggyanúsabbakról” (amelyeket a kiválasztott osztályozók a legnagyobb konfidenciával minősítenek csalásnak) küldjön értesítést.

Ami a Deployment fázis másik lehetséges metódusát illeti, a dolgozatomban bemutatott csalásdetektáló megoldás az AdaptEd projekt távolabbról vizsgált kutatási kontextusában jelentősen hozzájárulhat ahhoz, hogy egy nagy, sokszínű, validált és a legjobb tudásunk szerint egyedülálló adathalmazt hozzunk létre, amelynek vizsgálatával közelebb juthatunk tanulással kapcsolatos kérdések megválaszolásához.

Irodalomjegyzék

[1] Brett Molina and Marco della Cava: "Apple beats Samsung in Q4 smartphone sales", USA TODAY (2015 március 3.)

[2] Marc Prensky, On the Horizon (NCB University Press, Vol. 9 No. 5: "Digital natives, digital immigrants part 1.", On the Horizon, NCB University Press, Vol. 9 No. 5 (2001 október)

[3] Kelemen Rita: "Az interaktív tábla néhány módszertani lehetősége a közoktatásban és a tanárképzésben", Iskolakultúra Online (2008).

[4] Jennifer A. Mautone, George J. DuPaul, Asha K. Jitendra: „The Effects of Computer-Assisted Instruction on the Mathematics Performance and Classroom Behavior of Children With ADHD", Journal of Attention Disorders 9.1: 301-312 (2005)

[5] Julie Clarfield, Gary Stoner: „The Effects of Computerized Reading Instruction on the Academic Performance of Students Identified with ADHD", School Psychology Review 34.2: 246 (2005)

[6] Miia Ronimus, Janne Kujalab, Asko Tolvanenc, Heikki Lyytinenc: "Children's engagement during digital game-based learning of reading: The effects of time, rewards, and challenge." Computers & Education 71: 237-246 (2014)

[7] Dusza Andrea: „Biofeedback alapú mobil fejlesztőjáték diszkalkuliás gyermekeknek" (2014)

[8] Meixner Alapítvány, http://meixner.hu (2015)

[9] Microsoft: IIS, https://www.iis.net/ (2015)

[10] Microsoft: SQL Server, https://www.microsoft.com/en-us/server-cloud/products/sql-server/ (2015)

[11] MSDN: Entity Framework, https://msdn.microsoft.com/en-us/data/ef.aspx (2015)

[12] MSDN: What Is Windows Communication Foundation, https://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx (2015)

[13] Microsoft: ASP.NET, http://www.asp.net/mvc (2015)

[14] Razor, https://github.com/aspnet/Razor (2015)

[15] Zhang Shichao, Chengqi Zhang, Qiang Yang: "Data preparation for data mining", Applied Artificial Intelligence (2003)

[16] Karin Elke Krüll: „A diszkalkuliás (számolásgyenge) gyerekek", Akkord Kiadó (1996)

[17] John McCann, Matt Hanson: „Android Marshmallow release date, news and features”, http://www.techradar.com/news/software/operating-systems/android-m-10-things-we-d-like-to-see-1269443 (2015. október 12.)

[18] Kerry Flynn: „Your iPhone Can Be Hacked With A Photo Of Your Thumb”, http://www.huffingtonpost.com/2014/12/30/hack-phone-fingerprint-photographs_n_6395730.html (2014. december 31.)

[19] Rüdiger Wirth, Jochen Hipp: "CRISP-DM: Towards a standard process model for data mining." Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining (2000)

[20] Thomas G. Tape: „The Area Under an ROC Curve”, Interpreting Diagnostic Tests, http://gim.unmc.edu/dxtests/roc3.htm (2015)

[21] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, Dawn Song: “ Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication” Information Forensics and Security, IEEE Transactions on (2013)

[22] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, Heinrich Hussmann: „Touch me once and I know it’s you! Implicit Authentication based on Touch Screen Patterns”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2012)

[23] Cheng Bo, Lan Zhang, Xiang-Yang Li: "Silentsense: silent user identification via touch and movement behavioral biometrics." Proceedings of the 19th annual international conference on Mobile computing & networking (2013)

[24] Android Developer Reference: MotionEvent, http://developer.android.com/reference/android/view/MotionEvent.html (2015)

[25] Tao Feng, Jun Yang, Zhixian Yan, Emmanuel Munguia Tapia, Weidong Shi: „TIPS: Context-Aware Implicit User Identification using Touch Screen in Uncontrolled Environments”, Proceedings of the 15th Workshop on Mobile Computing Systems and Applications (2014)

[26] Meinard Müller: "Dynamic time warping", Information retrieval for music and motion (2007)

[27] Elena Tsiporkova: "Dynamic Time Warping Algorithm for Gene Expression Time Series" (2010)

[28] G. A. ten Holt, M. J. T. Reinders, E. A. Hendriks: "Multi-dimensional dynamic time warping for gesture recognition" (2007)

[29] scikit-learn, http://scikit-learn.org/stable/ (2015)

[30] Accord.NET Framework, http://accord-framework.net (2015)

[31] scikit-learn: „Generalized Linear Models”, http://scikit-learn.org/stable/modules/linear_model.html (2015)

[32] ID3 Demonstration, http://dtrees.cs.teiath.gr:10000/ (2015)

[33] J. R. Quinlan: „Induction of Decision Trees. Mach. Learn.” (1986), 81-106

[34] J. R. Quinlan: „C4.5: Programs for Machine Learning”, Morgan Kaufmann Publishers (1993)

[35] Y. Freund, R. Schapire: “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting” (1997)

[36] scikit-learn: „Support Vector Machines”, http://scikit-learn.org/stable/modules/svm.html (2015)

[37] pandas: „Python Data Analysis Library”, http://pandas.pydata.org/ (2015)

[38] scikit-learn: „Receiver Operating Characteristic (ROC)”, http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html (2015)

[39] James A. Hanley, Barbara J McNeil: "A method of comparing the areas under receiver operating characteristic curves derived from the same cases", Radiology (1983)

[40] Jingu Heo, M. Savvides: „Gender and Ethnicity Specific Generic Elastic Models from a Single 2D Image for Novel 2D Pose Face Synthesis and Recognition”, IEEE Transactions on Pattern Analysis & Machine Intelligence Issue No. 12. (2012)

[41] Google Play: AdaptiveR, http://tinyurl.com/AdaptiverTdk (2015)

[42] pypyodbc, https://code.google.com/p/pypyodbc/ (2015)