

Szegedi Tudományegyetem
Informatikai Intézet

Diplomamunka

Kun Réka Bianka

2025

**Szegedi Tudományegyetem
Informatikai Intézet**

**Mesterséges Intelligencia alapú azonosítás és
ajánlórendszerek alacsony erőforrású
környezetekben**

Diplomamunka

Készítette:

Kun Réka Bianka
programtervező
informatikus MSc szakos
hallgató

Témavezető:

Kiss-Vetráb Mercedes
PhD hallgató

**Szeged
2025**

Feladatkiírás

A Különböző mesterséges intelligencia algoritmusok segítségével megvalósított ajánlórendszerek, valamint autentikáció egyre elterjedtebbek, azonban nagy számítási kapacitást igényelnek. Emiatt alacsony erőforrású környezetekben (pl: mobiltelefonok, laptopok, korlátozott memóriájú eszközök) gyakran csak kiegészítő mérnöki megoldásokkal együtt alkalmazhatóak. A hallgató feladata, hogy olyan MI-modelleket tudjon futtatni a saját maga által fejlesztett flutter mobilalkalmazásban, melyek hatékonyan működnek korlátozott erőforrások mellett is, miközben a lehető legjobban megőrzik hatékonyságukat. Ehhez elégséges mélységben körbe kell járnia és meg kell ismernia az aktuálisan legfejlettebb releváns programozói módszereket (pl: TinyML, TFLite, quantisation, stb.). Emellett elengedhetetlen, hogy a hallgató megfelelő módon előkészítse a mobil alkalmazás mögött futó adatbázist, hogy az kiszolgálhassa a modelleket. A téma alkalmazható egészségügyi diagnosztikától kezdve oktatási ajánlórendszerekig, így a hallgató dönthet a konkrét alkalmazási terület kiválasztásáról.

Tartalmi összefoglaló

- **A téma megnevezése:**

Mesterséges Intelligencia alapú azonosítás és ajánlórendszerek alacsony erőforrású környezetekben.

- **A megadott feladat megfogalmazása:**

Egy olyan ajánlórendszer megvalósítása, amely a felhasználók érdeklődési köreit, preferenciáit és korábbi aktivitásait figyelembe véve személyre szabott eseményajánlásokat nyújt, elősegítve így a számukra legmegfelelőbb rendezvények gyors és egyszerű megtalálását. Ezáltal nemcsak az események keresése válik egyszerűbbé, hanem a felhasználói élmény is fokozódik, mivel a rendszer lehetővé teszi számukra, hogy gyorsabban rátaláljanak azokra a rendezvényekre, amelyek valóban felkeltik az érdeklődésüket. Továbbá, az arcfelismerési funkció integrálása lehetőséget biztosít arra, hogy a felhasználók gyorsan és egyszerűen lépjenek be az alkalmazásba, elfelejtve a hagyományos bejelentkezési folyamatokat, mint a jelszavak vagy bonyolult hitelesítési lépések. Ezáltal nemcsak kényelmesebbé válik az alkalmazás használata, hanem egy biztonságos és zökkenőmentes élményt is biztosít a felhasználók számára.

- **A megoldási mód:**

Az alkalmazás fejlesztése során fontos volt számomra, hogy a felhasználók személyre szabott, gyors és releváns eseményajánlásokat kapjanak, anélkül hogy az ajánlórendszer működése kizárólag a szerveroldalon történjen. Ennek megvalósításához egy gépi tanulási modellt integráltam közvetlenül az alkalmazásba, amely képes az eszközön futva, valós időben feldolgozni az adatokat és eseményeket kategorizálni. Ez a megközelítés csökkenti a szerverre nehezedő terhelést, gyorsabb válaszidőt biztosít, és fokozza a felhasználói élményt.

- **Alkalmazott eszközök, módszerek:**

Az alkalmazás elkészítéséhez a Flutter mobilalkalmazás-fejlesztési keretrendszert használtam, amely a Dart programozási nyelven alapul. Az adatok kezelésére a Google Firebase adatbázis-szolgáltatását alkalmaztam a Firestore-t. A projekt kódjának fejlesztésére a Google Colabot is igénybe vettem, amely egy könnyen használható, felhő alapú fejlesztői környezetet biztosít, így biztosítva a zökkenőmentes és hatékony munkát a kód írása és tesztelése során.

- **Elért eredmények:**

Egy olyan ajánlórendszer és arcazonosításon alapuló hitelesítés, amely megbízható és felhasználóbarát. Könnyen integrálódik az eszközökkel.

- **Kulcsszavak:**

Deep Learning, modell, arcfelismerés, felügyelt tanulás, ajánlórendszer, embedding.

Tartalomjegyzék

Feladatkiírás	3
Tartalmi összefoglaló	4
Tartalomjegyzék.....	6
Motiváció.....	8
1. Bevezetés	9
1.1 A téma jelentősége és aktualitása.....	9
1.2 A gépi tanulási ajánlórendszerek szerepe a modern technológiákban	9
1.3 Az arcfelismerés szerepe a modern technológiákban.....	9
2. A gépi tanulás	10
2.1 A gépi tanulás alapjai és alapfogalmai	10
2.2 Alapvető algoritmusok és technikák.....	10
2.3 Felhasználási területek.....	13
3. Ajánlórendszerek típusai és alapvető megközelítései	14
3.1 Kollaboratív szűrés	15
3.2 Tartalom alapú ajánlás.....	16
3.3 Kollaboratív – Tartalom alapú hibrid rendszerek.....	17
3.4 Flutter alkalmazásban használt módszerek.....	19
3.4.1 Cosinus hasonlóság.....	19
4. Gépi tanulási modellek használata alacsony erőforrású eszközökön	23
4.1 Mélytanulási megközelítések (Deep Learning).....	23
4.2 Optimalizált modellformátumok mobil és edge eszközökhöz	24
4.3 Használt módszerek	26

4.3.1 MobileNetV2 modell alkalmazása mobil eszközön.....	26
4.3.2 Generatív nyelvi modell alkalmazása	28
5. Arcfelismerés.....	31
5.1 Arcfelismerés történeti áttekintése.....	31
5.2 Arcfelismerési rendszerek típusai	31
5.2.1 Arcdetektálás.....	34
5.2.2 Arcazonosítás	36
5.2.3 Arcverifikáció.....	37
5.2.4 Használt módszerek.....	38
Irodalomjegyzék.....	42
Nyilatkozat.....	48
Köszönetnyilvánítás	49

Motiváció

A technológiai rohamos fejlődése új lehetőségeket teremt az eseményszervezés területén, így a felhasználói élmény még inkább személyre szabottá válhat.

A felhasználók gyakran elvesznek a rengeteg elérhető esemény között, és nem mindig találják meg azokat, amelyek igazán érdeklik őket. Emellett a bejelentkezési folyamat során is felmerülhetnek nehézségeik, mint a jelszóemlékezés vagy az adatok újra beírása, ami zavaró és időigényes lehet. Ezáltal a felhasználói élmény csökkenthet, és a látogatók könnyen elfordulhatnak az alkalmazástól.

A célom egy olyan ajánlórendszer és arcazonosító rendszer megvalósítása volt, amelyek hatékonyan kezelik ezeket a problémákat a felhasználói élmény növelése érdekében.

1. Bevezetés

1.1 A téma jelentősége és aktualitása

1.2 A gépi tanulós ajánlórendszerek szerepe a modern technológiákban

A gépi tanulós ajánlórendszerek jelentős szerepet játszanak a modern technológiákban, mivel személyre szabott élményeket biztosítanak a felhasználók számára. Ezek az algoritmusok a folyamatosan növekvő adatmennyiséget és a feldolgozási kapacitást kihasználva képesek a felhasználói szokások, preferenciák és viselkedési minták alapján releváns tartalmat, termékeket vagy szolgáltatásokat javasolni. [1]

1.3 Az arcfelismerés szerepe a modern technológiákban

Az arcfelismerés a modern technológiák egyik kiemelkedő és sokoldalúan alkalmazható területe, amely jelentős hatással van számos iparágra és mindennapi életünkre. Az alábbiakban egyik fő szerepét emelem ki:

- **Biztonság és azonosítás**

Az arcfelismerés megbízható és érintésmentes megoldást kínál a személyazonosításra.

- **Alkalmazási példák:**

- Beléptetőrendszerek, például repülőtereken és vállalati környezetben.

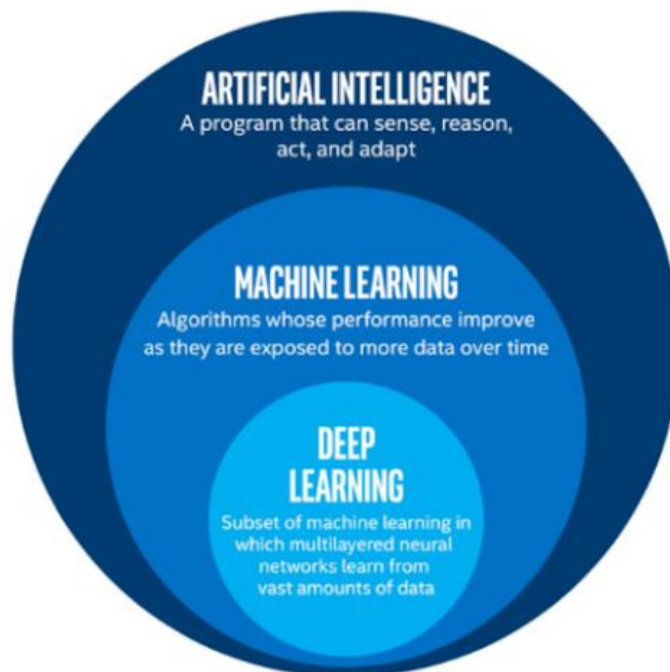
- Okoseszközök feloldása (pl. Face Id).

Ezek a rendszerek nemcsak kényelmesebbé teszik a mindennapi tevékenységeket, hanem növelik a biztonságot is. [2]

2. A gépi tanulás

2.1 A gépi tanulás alapjai és alapfogalmai

A gépi tanulás a mesterséges intelligencia (MI) algoritmusok egy részhalmaza, (2.1 ábra) amely az adathalmazokon betanított algoritmusokat foglalja magába. Ezek a modellek képesek előrejelzéseket készíteni, információkat osztályozni vagy egyéb feladatokat is ellátni, emberi beavatkozás nélkül.



2.1 ábra - A Mesterséges Intelligencia és a gépi tanulás kapcsolata

A gépi tanulás segítségével előállított modelleket napjainkban számos kereskedelmi célra alkalmazzák, például termékek ajánlására a vásárlók korábbi vásárlásai alapján, tőzsdei ingadozások előrejelzésére, vagy szövegek fordítására egyik nyelvről a másikra. [3, 4]

2.2 Alapvető algoritmusok és technikák

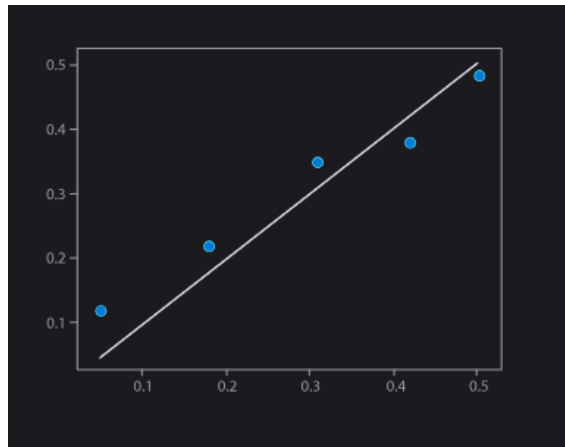
A gépi tanulás alapvető algoritmusai és technikái azok az eszközök, amelyek lehetővé teszik különböző problémák megoldását, mint például osztályozás, előrejelzés, mintázatfelismerés vagy adatbányászat.

Ezek az algoritmusok az adatok feldolgozására, elemzésére és értelmezésére szolgálnak, és az alkalmazott technikák alapvetően meghatározzák, hogy a gépi tanulási modellek milyen gyorsan és hatékonyan képesek a kívánt eredményeket elérni.

Ilyen algoritmusok a következők:

- **Lineáris Regresszió (Linear Regression)**

Működés: A lineáris regresszió egy gépi tanulási algoritmus (2.2 ábra), amelyet elsősorban előrejelzések készítésére alkalmaznak. Célja, hogy egy lineáris kapcsolatot találjon a bemeneti változók és a kimeneti változó között. Az algoritmus egy egyenest illeszt az adatokhoz, amely a legjobban képes előre jelezni a kívánt kimenetet.

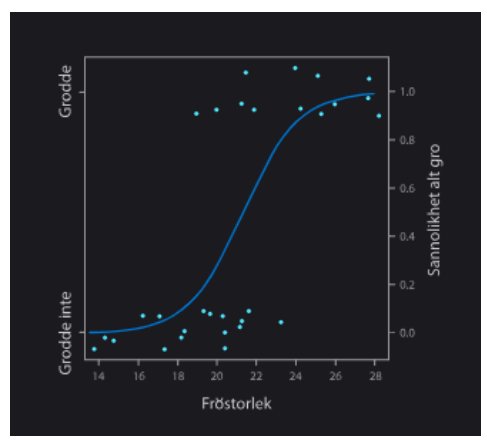


2.2 ábra - Lineáris regresszió algoritmus ábrázolása

Alkalmazás: Használható például ingatlanárak, részvények árai vagy bármely olyan probléma esetén, ahol a kimeneti érték folyamatos és lineárisan összefügg a bemeneti változókkal.

- **Logisztikus Regresszió (Logistic Regression)**

Működés: A logisztikus regresszió egy osztályozási algoritmus (2.3 ábra), amely a kimeneti változót valószínűségi értékként kezeli. Ez a módszer különösen hasznos olyan esetekben, amikor a kimenet két kategóriába sorolható, mint például „igen” vagy „nem”.

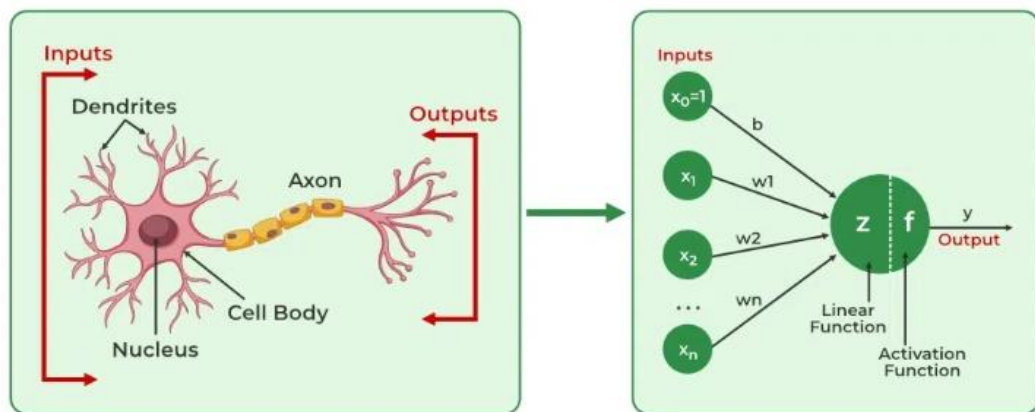


2.3 ábra - Logisztikus regresszió algoritmus ábrázolása

Alkalmazás: Alkalmas osztályozási problémák megoldására, például spam-e-mailek felismerésére vagy betegségek előrejelzésére.

- **Neurális Hálózatok (Neural Networks)**

Működés: A neurális hálózatok a biológiai idegrendszer működését modellezzik, és több rétegből állnak, mint például bemeneti, rejtett és kimeneti rétegek (2.4 ábra). Az adatok áthaladnak ezeken a rétegeken, ahol minden réteg különböző mintázatokat ismer fel, lehetővé téve ezzel a komplex problémák megoldását.

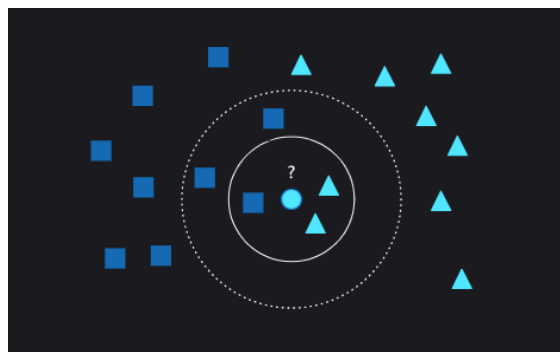


2.4 ábra - Neurális hálózat szemléltetése

Alkalmazás: Képfeldolgozás, hangfelismerés, természetes nyelv feldolgozás, valamint számos más bonyolult problémák megoldása során alkalmazzák.

- **K Legközelebbi Szomszédok (K Nearest Neighbors, KNN)**

Működés: A KNN egy egyszerű, de hatékony algoritmus, amely a legközelebbi szomszédok elvén alapul. Az új adatpontokat a legközelebbi K szomszédjuk alapján osztályozza a már meglévő adatok figyelembevételével. Az osztályozás a szomszédok többségi osztálya szerint történik (2.5 ábra).



2.5 ábra - KNN algoritmus ábrázolása

Alkalmazás: Klasszifikációs és regressziós problémák megoldására, például ajánlórendszerekben vagy ügyfélszegmentálásban. [5, 6, 7]

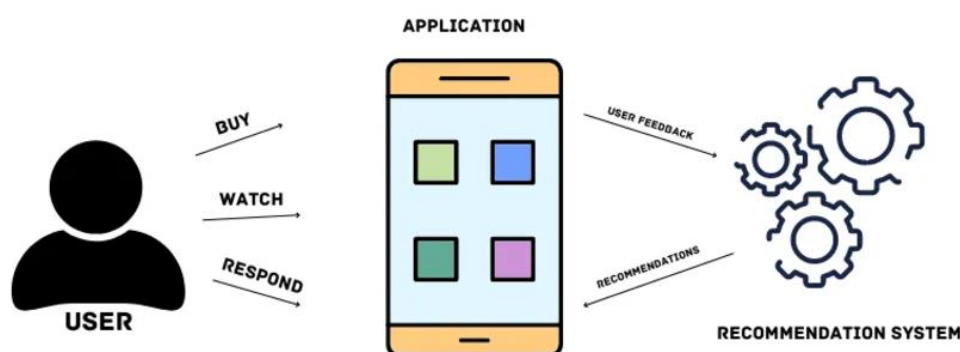
2.3 Felhasználási területek

A gépi tanulás napjainkban a mesterséges intelligencia szélesebb körben használt technológiája. A mindennapi életben számos területen találkozhatunk vele, például:

- Beszédfelismerés: A neurális hálózatok különösen a rekurzív vagy transzformer alapú modellek képesek a hanghullámokat elemezni és azokat pontos szöveggé alakítani, így segítve például diktálást vagy hangalapú keresést.
- Csalásmegelőzés a banki szektorban: Itt gyakran alkalmaznak logisztikus regressziót, döntési fákat vagy mély neurális hálózatokat arra, hogy a rendszer észrevegye azokat a tranzakciós mintákat, amelyek eltérnek a megszokottól például szokatlan időpontban vagy helyszínen végzett vásárlásokat.
- Önvezető autók és vezetést segítő rendszerek: A konvolúciós neurális hálózatok (CNN) segítségével az autók képesek érzékelni és azonosítani a gyalogosokat, az útburkolati jeleket és a közlekedési táblákat.
- Ajánlórendszerek: A gépi tanulás egyik legismertebb alkalmazása a személyre szabott termék-, zene- vagy tartalomajánlás, amelyet olyan platformokon találunk, mint a Netflix, a Spotify vagy az Amazon. A KNN algoritmus segítségével a rendszer a hasonló érdeklődési körű felhasználók viselkedését figyelembe véve ajánl új tartalmakat. A logisztikus regresszió vagy a neurális hálózatok képesek előre jelezni, hogy egy felhasználó valószínűleg kedvelni fogja-e a tartalmat, figyelembe véve a korábbi interakciókat. [8, 9, 10]

3. Ajánlórendszerek típusai és alapvető megközelítései

Az ajánlórendszerek olyan algoritmusokon és technológiákon alapulnak, amelyek célja, hogy a felhasználóknak személyre szabott javaslatokat kínáljanak különféle termékek, szolgáltatások vagy tartalmak tekintetében. Ezek a rendszerek kiemelten fontosak az e-kereskedelemben, a szórakoztatóiparban, valamint számos egyéb területen, ahol a releváns ajánlások javíthatják a felhasználói élményt és növelhetik az üzleti sikereket. Az ajánlórendszerek működésének általános folyamata a 3.1. ábrán kerül bemutatásra.



3.1 ábra - Általános ajánlási rendszer folyamata

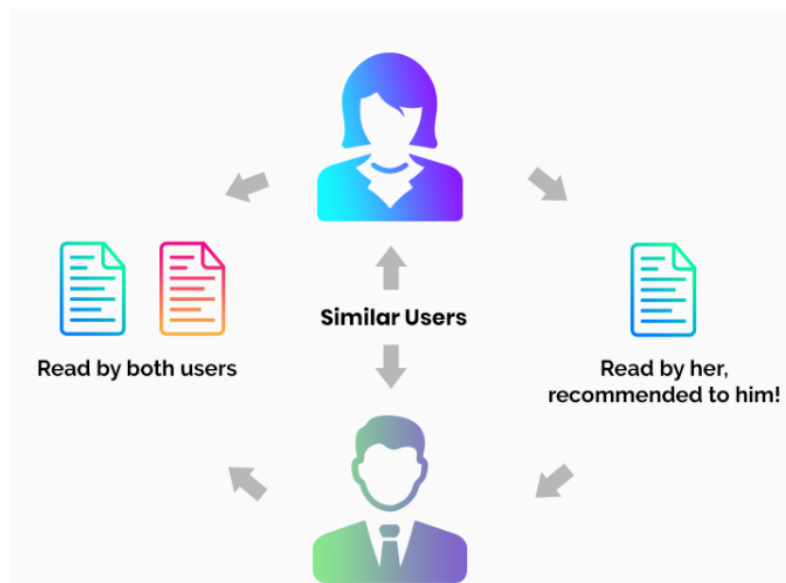
Az ajánlórendszerek több különböző megközelítés alapján működnek:

- Kollaboratív szűrés: Azoknak az elemeknek az ajánlása, amelyek a hasonló érdeklődésű felhasználók preferenciái alapján kerülnek kiválasztásra.
- Tartalom-alapú szűrés: Olyan elemek ajánlása, amelyek összhangban vannak a felhasználó profiljával vagy korábbi interakcióival.
- Kontekstuális ajánlások: Ajánlások, amelyek figyelembe veszik az időpontot, évszakot vagy az aktuális trendeket, hogy releváns tartalmakat ajánljanak.
- Szekvenciális ajánlások: A felhasználó aktuális tevékenységeinek elemzése alapján történő előrejelzés, hogy mi lehet a következő érdeklődési pont.
- Szociális ajánlások: A szociális kapcsolatok és hatások felhasználása az ajánlott elemek meghatározásában.

Mindegyik megközelítés egyaránt arra törekszik, hogy személyre szabott ajánlásokat nyújtson a felhasználóknak, azonban eltérő módszereket alkalmaznak. [11, 12, 13]

3.1 Kollaboratív szűrés

A kollaboratív szűrés azon a feltételezésen alapul, hogy a hasonló érdeklődési körű felhasználók hajlamosak hasonló döntéseket hozni. (3.2 ábra) Az algoritmus a rendelkezésre álló viselkedési adatokat például vásárlásokat, értékeléseket, kattintásokat vagy tartalomfogyasztási szokásokat használja, hogy összefüggéseket találjon a felhasználók és a hozzájuk kapcsolódó elemek között. Ezen adatok alapján az algoritmus megjósolja, milyen termékeket vagy tartalmakat találhat vonzónak egy adott felhasználó. [14]

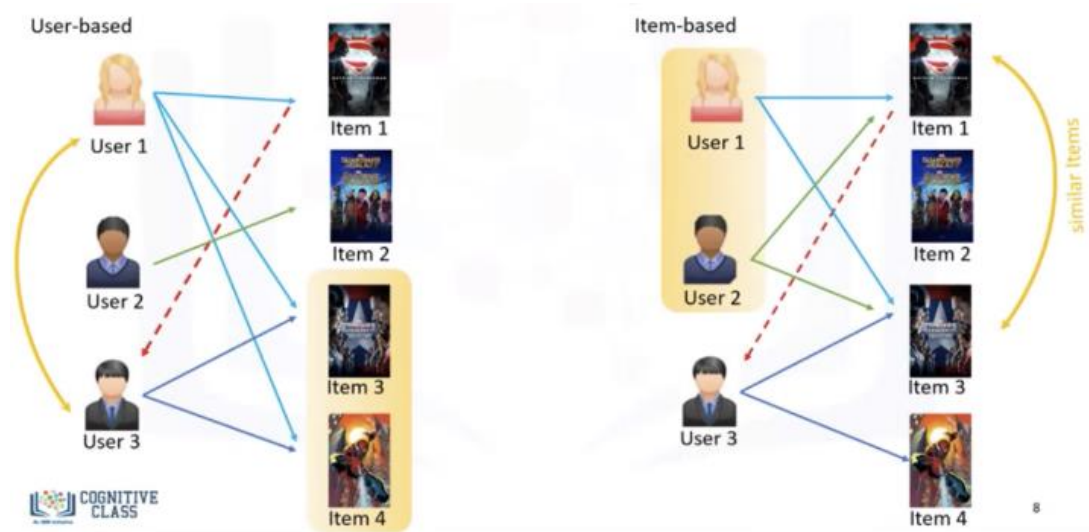


3.2 ábra - Kollaboratív szűrés

A kollaboratív szűrés két fő módszert használ: (3.3 ábra)

- Felhasználó-alapú kollaboratív szűrés (User-Based Collaborative Filtering): Az algoritmus azokat a felhasználókat elemzi, akik hasonló érdeklődési körrel vagy preferenciákkal rendelkeznek, és az ő döntéseik alapján ajánl új elemeket. Például, ha egy felhasználó egy adott filmet kedvelt, akkor olyan filmek kerülnek számára ajánlásra, amelyeket más, hasonló ízlésű felhasználók szintén kedveltek.
- Elem-alapú kollaboratív szűrés (Item-Based Collaborative Filtering) Ebben a módszerben az algoritmus az elemek közötti hasonlóságokat vizsgálja. Az algoritmus nem a felhasználók preferenciáit veszi figyelembe, hanem a termékek közötti kapcsolatokat és interakciókat elemzi. Például, ha egy felhasználó vásárolt egy bizonyos típusú laptopot, a rendszer olyan további

termékeket javasol, amelyek hasonló vásárlási minták alapján népszerűek más felhasználók körében. [15, 16, 17]



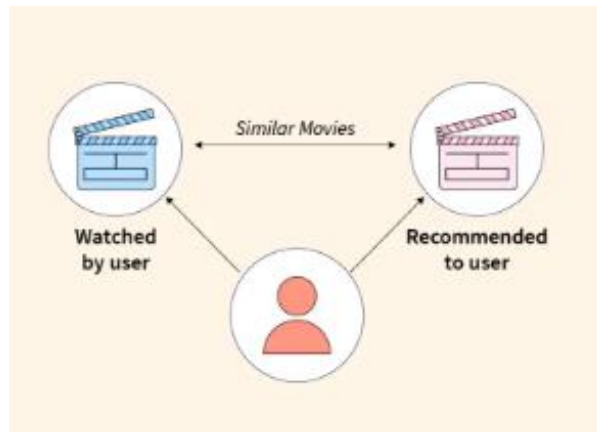
3.3 ábra - Felhasználó és elem alapú megközelítés

A kollaboratív szűrés alapvető eleme a modern ajánlórendszereknek, mivel egyszerű és hatékony megoldást nyújt a személyre szabott ajánlások készítésére anélkül, hogy az ajánlott tartalmakról részletes információval kellene rendelkezni.

3.2 Tartalom alapú ajánlás

A tartalomalapú ajánlás azon alapul, hogy az algoritmus a termékek vagy tartalmak tulajdonságait, mint például a cím, szerző, műfaj, kulcsszavak, vagy bármilyen egyéb jellemző alapján végez elemzést.

Az algoritmus a felhasználó korábbi interakcióit, például kedvenc termékeit, filmjeit vagy cikkeket, figyelembe véve ajánl új tartalmakat, amelyek hasonlóak azokhoz, amiket már kedvelt vagy megtekintett (3.4 ábra). A cél az, hogy a felhasználó érdeklődési köréhez passzoló, releváns ajánlásokat nyújtson, figyelmen kívül hagyva más felhasználók preferenciáit. [18, 19]



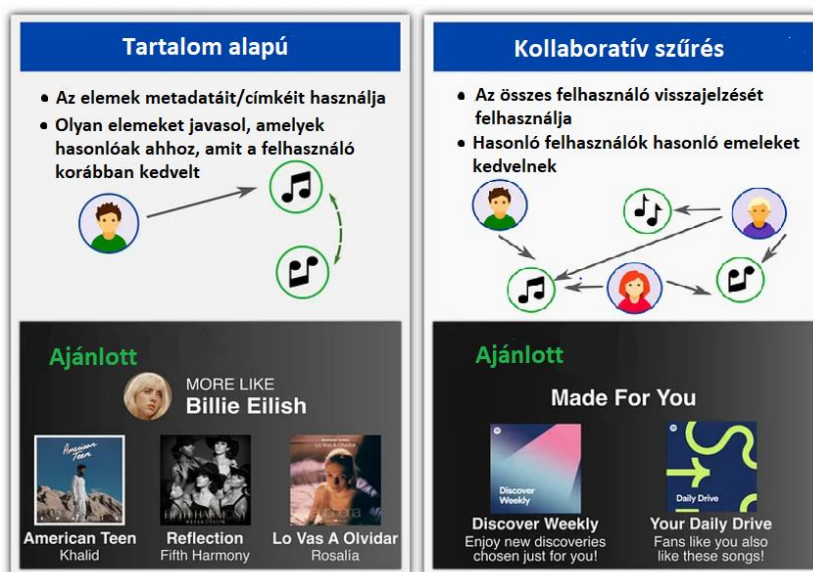
3.4 ábra - Tartalom alapú ajánlás

A tartalomalapú ajánlási rendszerek széles körben elterjedtek különböző online platformokon, például e-könyv ajánló rendszerekben, mint a Goodreads, ahol az olvasók érdeklődési körének megfelelő könyveket ajánlanak. Ugyanezt a módszert alkalmazzák a film- és sorozatajánló platformok, például a Netflix, a Hulu vagy az Amazon Prime Video is, amelyek a felhasználók korábbi megtekintései és preferenciái alapján ajánlanak tartalmakat. Ezen kívül hírszolgáltatók, mint a Google News vagy a Flipboard is tartalomalapú ajánlásokat használnak, hogy személyre szabott híreket és cikkeket kínáljanak a felhasználóknak.

A zenei platformok, mint a Spotify és az Apple Music, szintén alkalmazzák ezt a megközelítést, hogy a felhasználók korábbi zenei preferenciái alapján új zeneszámokat vagy albumokat ajánljanak. [20]

3.3 Kollaboratív – Tartalom alapú hibrid rendszerek

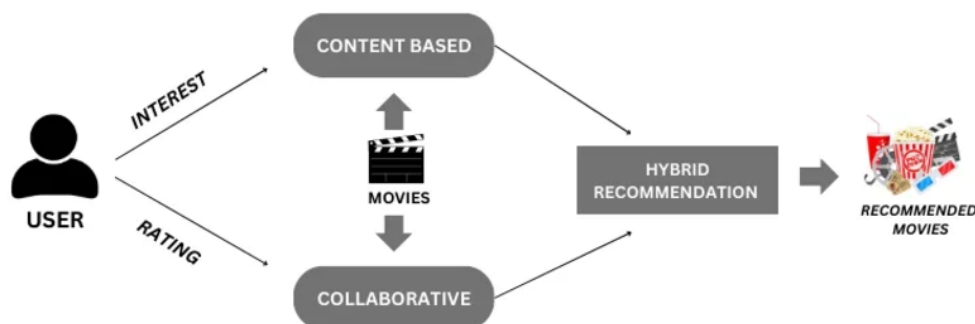
A kollaboratív szűrés más, hasonló érdeklődésű felhasználók választásaira épít, míg a tartalomalapú szűrés az események jellemzőit elemzi, és a felhasználó által korábban kedvelt eseményekhez hasonlókat ajánl. A kollaboratív szűrés előnye, hogy változatosabb és újabb eseményeket is képes ajánlani, még akkor is, ha a felhasználó korábban nem találkozott azokkal, míg a tartalomalapú szűrés hajlamos inkább a már ismert érdeklődési körhöz hasonló eseményeket ajánlani. Ugyanakkor a tartalomalapú módszer független a többi felhasználótól, így kevesebb felhasználó mellett is jól működik, míg a kollaboratív szűrés hatékonysága a rendelkezésre álló felhasználói adatok mennyiségétől függ. A 3.5. ábrán a tartalomalapú és a kollaboratív szűrés működésének összehasonlítása látható.



3.5 ábra - Tartalom alapú és Kollaboratív szűrés

A legjobb eredményt gyakran a két módszer kombinálásával, egy hibrid ajánlórendszer kialakításával lehet elérni, amely ötvözi a tartalomalapú szűrés precizitását és a kollaboratív szűrés változatosságát. Így a felhasználók nemcsak az érdeklődési körükhöz leginkább illeszkedő eseményeket találhatják meg könnyebben, hanem új, releváns ajánlásokat is kaphatnak más, hasonló preferenciájú emberek választásai alapján.

A hibrid rendszerek a kollaboratív szűrés és a tartalomalapú ajánlás előnyeit ötvözik (3.6 ábra), hogy javítsák az ajánlások pontosságát és kiküszöböljék az egyes módszerek hátrányait. Ezek a rendszerek képesek dinamikusan alkalmazkodni a változó felhasználói preferenciákhoz, és figyelembe venni a felhasználói viselkedést, valamint a termékek vagy tartalmak specifikus jellemzőit.



3.6 ábra - Hibrid ajánlórendszer

Az ilyen rendszerek különösen hasznosak lehetnek azokban az esetekben, amikor egyik megközelítés önállóan nem elegendő. Például amikor egy új felhasználó

csatlakozik a rendszerhez. Mivel a kollaboratív szűrés a korábbi felhasználói interakciókat veszi figyelembe, az új felhasználók esetében, akik még nem végeztek elegendő tevékenységet az algoritmus nem rendelkezik elegendő információval ahhoz, hogy személyre szabott ajánlásokat tegyen. A hibrid megközelítés ebben az esetben hatékony megoldást nyújt, mivel tartalomalapú elemzés segítségével képes pótolni a hiányzó adatokat. A felhasználó által megadott érdeklődési körök, preferenciák vagy egyéb profiljellemzők alapján már a kezdetektől hasznos ajánlásokat tud generálni. [21, 22, 23]

3.4 Flutter alkalmazásban használt módszerek

A rendszer megvalósítása során fontos szerepet játszott a megfelelő módszer kiválasztása annak érdekében, hogy a rendszer minél jobban a felhasználók igényeire összpontosítson. A fejlesztett rendszerben a tartalomalapú ajánlás módszere került alkalmazásra, amely lehetővé teszi, hogy a felhasználók saját preferenciáik és érdeklődési köreik alapján kapjanak személyre szabott eseményajánlásokat. Ennek nemcsak az az előnye, hogy az ajánlások pontosabban illeszkednek az adott felhasználó ízléséhez, hanem az is, hogy képes új eseményeket is ajánlani, amelyeket más, hasonló érdeklődésű felhasználók talán figyelmen kívül hagytak. Ezáltal a felhasználók nemcsak a megszokott események közül válogathatnak, hanem olyan programokat is felfedezhetnek, amelyek új élményekkel gazdagítják őket. A rendszer így nemcsak személyre szabott, hanem figyelemfelkeltő ajánlásokat is ad, szélesítve a felhasználók látókörét.

Az ajánlórendszer fejlesztése során két eltérő megközelítést alkalmaztam annak érdekében, hogy a felhasználók számára minél pontosabb és relevánsabb ajánlásokat biztosítsak. Az egyik módszer a Cosinus hasonlóságon alapul, amely a felhasználók korábbi interakcióit és az események jellemzőit összevetve határozza meg a hasonlóság mértékét. A másik megközelítés a MobileNetV2 modell alkalmazása, amely lehetővé teszi a vizuális tartalmak, mint az eseményképek hatékony feldolgozását még alacsony erőforrású eszközökön is.

3.4.1 Cosinus hasonlóság

A fejlesztés során először a Cosinus hasonlóság módszerét használtam mivel ez egy hatékony és széles körben alkalmazott eljárás a tartalomalapú ajánlórendszerekben.

A módszer lényege, hogy az események jellemzőit vektoros reprezentációk (embeddingek) segítségével ábrázolja. Ezeket a vektorokat összehasonlítva számítja ki a hasonlósági értéket, amely megmutatja, mennyire illeszkedik egy adott esemény a felhasználó érdeklődési köréhez. [24, 25]

A Cosinus hasonlóság számításához alkalmazott embeddingek generálása (3.7 ábra) egy Google Colab környezetben lett megvalósítva, ahol a Firestore adatbázisból lekérdezett események leírásait a TF-IDF (Term Frequency-Inverse Document Frequency) vektorizáló segítségével numerikus vektorokká alakítom. A TF-IDF a szavak fontosságát méri egy dokumentumban, figyelembe véve azok gyakoriságát és ritkaságát az egész dokumentumkészletben. A TF-IDF segítségével az események leírása alapján meghatározhatjuk, mely kulcsszavak és kifejezések a legrelevánsabbak az adott esemény szempontjából. Ennek köszönhetően az esemény típusát is meghatározhatjuk, például hogy sport, művészet vagy családbarát program, attól függően, mely szavak jelennek meg gyakrabban a leírásban. A TF-IDF vektorok segítségével a rendszer képes felismerni az események témáját és típusát, ezáltal relevánsabb ajánlásokat tud kínálni a felhasználóknak. A leírások alapján generált embeddingek ezután visszakerülnek az adatbázisba, így minden esemény rendelkezik egy egyedi, vektoros reprezentációval. [26, 27]

A folyamat automatizálása érdekében a GitHub Actions-t alkalmaztam, amely óránként végrehajtja a számításokat, biztosítva ezzel az adatok folyamatos és naprakész frissítését. A GitHub Actions egy fejlett CI/CD eszközkészlet, amelyet a GitHub biztosít a kódok kezelésére és automatizálására. Az Actions segítségével különböző munkafolyamatokat (workflow-kat) lehet automatizálni közvetlenül a GitHub repository-ban, külső eszközök használata nélkül. [28, 29]

Ha egy újonnan létrehozott eseményhez még nincs embedding, a rendszer azt a következő frissítés alkalmával hozzáadja, biztosítva ezzel, hogy minden esemény a szükséges adatokkal legyen ellátva.

```

collection = firestore_client.collection("events")
docs = collection.stream()

data = []
doc_ids = []

for doc in docs:
    doc_data = doc.to_dict()
    data.append({
        "id": doc.id,
        "description": doc_data.get("description"),
    })
    doc_ids.append(doc.id)

df = pd.DataFrame(data)

tfidf_vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf_vectorizer.fit_transform(df['description'])

embeddings = tfidf_matrix.toarray()

collection = firestore_client.collection("events")

for i in range(len(doc_ids)):
    doc_ref = collection.document(doc_ids[i])

    doc = doc_ref.get()

    if doc.exists and 'embedding_field' not in doc.to_dict():
        doc_ref.update({
            "embedding_field": embeddings[i].tolist()
        })

```

3.7 ábra – Az alkalmazáshoz használt embedding számítás

A Cosinus hasonlóság számítása (3.8 ábra) közvetlenül az alkalmazásban történik, ahol a rendszer a felhasználó által korábban részt vett események leírásaiból létrehozott embeddingeket hasonlítja össze a többi esemény leírását reprezentáló embeddingekkel (5.10 ábra).

```

class CosineSimilarity {

    double cosineSimilarity(List<double> vec1, List<double> vec2) {
        double dotProduct = 0.0;
        double magnitude1 = 0.0;
        double magnitude2 = 0.0;
        for (int i = 0; i < vec1.length; i++) {
            dotProduct += vec1[i] * vec2[i];
            magnitude1 += vec1[i] * vec1[i];
            magnitude2 += vec2[i] * vec2[i];
        }
        if (magnitude1 == 0 || magnitude2 == 0) return 0.0;
        return dotProduct / (sqrt(magnitude1) * sqrt(magnitude2));
    }

}

```

3.8 ábra – Az alkalmazásban használt Cosinus hasonlóság számítás

A kiszámított hasonlósági értékek alapján az alkalmazás a főképernyőn, a CarouselSlider widget segítségével, dinamikusan jeleníti meg a felhasználó számára ajánlott eseményeket. (3.9 ábra)



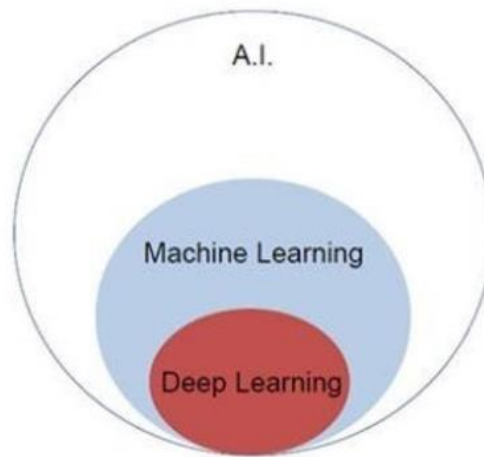
3.9 ábra – Az alkalmazásban megjelenített ajánlott események Cosinus hasonlósággal

A CarouselSlider lehetővé teszi, hogy a felhasználó könnyedén végig pörgessen az ajánlott eseményeken, amelyek frissítése automatikusan megtörténik, amint a felhasználó egy új eseményen vesz részt, így az ajánlások mindig az aktuális érdeklődéséhez igazodnak. Az egyes események képekkel és rövid leírásokkal jelennek meg, hogy még könnyebbé váljon a felhasználók számára az érdeklődésüknek leginkább megfelelő események gyors kiválasztása.

4. Gépi tanulási modellek használata alacsony erőforrású eszközökön

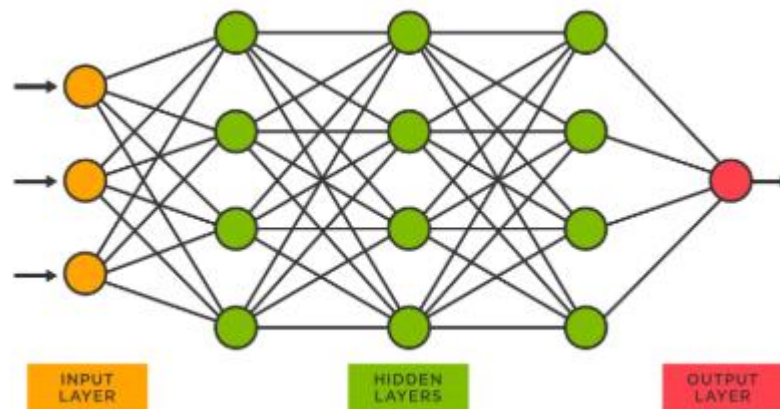
4.1 Mélytanulási megközelítések (Deep Learning)

A mélytanulás (Deep Learning) a gépi tanulás egy speciális ága, amely mesterséges neurális hálózatok segítségével old meg bonyolult problémákat.



4.1 ábra - Mélytanulás és a gépi tanulás

A mélytanulás alapját a több rétegből álló neurális hálózatok képezik, amelyek nagy mennyiségű adat feldolgozására és értelmezésére képesek. A neurális hálózatok legkisebb építőeleme a neuron, amely egy olyan számítási egység, amely bemeneti jeleket fogad, azokat súlyozza, majd egy aktivációs függvény segítségével továbbítja az eredményt a következő réteg felé. Ezeknek a modelleknek a célja, hogy önállóan tanuljanak, mintázatokat ismerjenek fel és döntéseket hozzanak, mindezt anélkül, hogy előre meghatározott szabályokat kellene alkalmazniuk. [30, 31, 32]



4.2 ábra - Mélytanulási rétegek

A mélytanulási hálózat rétegei:

- **Bemeneti réteg (Input Layer):** Ez a réteg önálló számításokat nem végez, csupán beolvassa és továbbítja az adatokat a következő rétegek felé. A bemeneti réteg minden egyes bemeneti jellemzőhöz, például egy kép pixelértékeihez vagy egy szenzor mérési adataihoz, egy külön neuron kapcsolódik. Az adatokat vektorként vagy mátrixként ábrázolják, majd továbbítják a következő rétegekhez, amelyek a rejtett rétegek. Az előfeldolgozási műveletek, mint például a normalizálás, általában a bemeneti réteg előtt zajlanak, ezzel biztosítva az adatok megfelelő formátumát és minőségét a hatékony tanulási folyamat érdekében.
- **Rejtett rétegek (Hidden Layers):** A rejtett rétegek felelősek az adatok feldolgozásáért és a kívánt kimenet előállításáért. Ezek a rétegek végzik el a legösszetettebb számításokat. Minden egyes réteg az előző rétegekből származó kimeneti adatokat használja fel a további feldolgozáshoz. A rejtett rétegek hierarchikus struktúrában működnek: az alsóbb rétegek az egyszerűbb adatjellemzőket, például vonalakat vagy éleket azonosítanak, míg a magasabb rétegek egyre bonyolultabb mintázatokat és összefüggéseket képesek felismerni, mint például tárgyakat, személyeket vagy egész jeleneteket.
- **Kimeneti réteg (Output Layer):** A kimeneti réteg a mélytanulási hálózat utolsó része, amely az előző rétegek által feldolgozott adatokat alapján arra törekszik, hogy visszaadja az elvárt kimenetet. Ez a réteg végzi el a korábbi rétegek által generált eredmények végső kiértékelését, és azokat a konkrét feladathoz igazítva alakítja át, például osztálycímkek (osztályozás esetén) vagy folytonos értékek (regresszió esetén) formájában. A kimeneti réteg biztosítja a háló választ, amelynek meg kell felelnie a várt célértékeknek. [33]

4.2 Optimalizált modellformátumok mobil és edge eszközökhöz

A gépi tanulási modellek mobil és edge eszközökön történő futtatása számos kihívást jelent a számítási kapacitás és memória szempontjából. Az ilyen eszközök gyakran korlátozott erőforrásokkal rendelkeznek, így a modellek futtatása optimalizált formátumokat igényelnek, amelyek biztosítják a gyors és hatékony működést. [34]

A TensorFlow Lite (rövidítve .tflite) formátumot a TensorFlow keretrendszerhez fejlesztették ki, hogy optimalizált módon támogassa a gépi tanulási modellek futtatását

mobil és edge eszközökön. A TensorFlow Lite kifejezetten olyan eszközökre lett tervezve, amelyek korlátozott számítási kapacitással és memóriával rendelkeznek, így a modellek kisebbek és gyorsabban futtathatók, mint a hagyományos .pb (TensorFlow Protocol Buffer) vagy .h5 (Keras) modellek.

A .tflite formátumban lévő modellek az eredeti TensorFlow modellek konvertálásával jönnek létre, amelyeket a TensorFlow Lite konverter optimalizál. A konvertálás során különböző technikákat alkalmaznak, például a kvantálást, amely lehetővé teszi a modellek számára, hogy kisebb memóriát és számítási kapacitást igényeljenek anélkül, hogy jelentősen csökkentenék a modell teljesítményét. A kvantálás a modellek paramétereinek átalakítását jelenti, amely során a modellek által használt nagy pontosságú, lebegőpontos számokat kisebb, alacsonyabb bitmélységű számokkal helyettesítik. [35]

Ezen kívül más gépi tanulási keretrendszerek is saját bináris modellformátumokat kínálnak, amelyek kifejezetten mobil és edge eszközökre lettek optimalizálva. Ilyen formátumok például az OpenVINO által használt .bin, a CoreML által támogatott .mlmodel, valamint az NCNN keretrendszer .param és .bin formátumai. Ezek a formátumok lehetővé teszik a modellek gyors és valós idejű futtatását közvetlenül az eszközön, miközben minimális számítási kapacitást és memóriahasználatot igényelnek. Ezáltal ideális megoldást jelentenek olyan környezetekben, ahol az erőforrások korlátozottak.

A modellek átalakítása során ezek a keretrendszerek az eredeti modelleket saját bináris formátumaikba konvertálják, hogy azok jobban kihasználják az adott hardver teljesítményét. Például egy TensorFlow vagy Keras modell átalakítható az OpenVINO .bin formátumára, vagy egy CoreML modell .mlmodel formátumra, amely az Apple eszközein futtatható. Az átalakítás során különböző optimalizálási technikákat alkalmaznak, mint például a kvantálás, műveletfúzió és pruning, hogy a modellek kisebbek legyenek, gyorsabban fussanak és kevesebb erőforrást igényeljenek. A műveletfúzió (operation fusion) egy olyan technika, amely lehetővé teszi, hogy több különálló számítási lépést egyetlen, összetett műveletté alakítsunk. Ezáltal csökkenthető a végrehajtási idő és a számítási költség, mivel a modellek kevesebb lépésben hajtják végre a feladatokat, ami gyorsabb futtatást eredményez. A pruning során a modell paramétereinek egy részét eltávolítják, amelyek nem járulnak hozzá jelentősen a kimenethez vagy a teljesítményhez. Ez csökkenti a modell komplexitását, mivel kevesebb súly és kapcsolódó paraméter marad a hálóban. A pruning hatására kisebb

modellek keletkeznek, amelyek gyorsabban futtathatók, kevesebb memóriát használnak, és könnyebben alkalmazhatók erőforrás-korlátozott környezetekben. [36, 37, 38, 39]

Ezek a formátumok lehetővé teszik, hogy a modellünk közvetlenül az eszközön fusson, anélkül, hogy külső szerverekre vagy internetkapcsolatra lenne szükség. Ez jelentősen csökkenti a késleltetést, mivel az adatokat nem kell továbbítani, és a feldolgozás az eszközön, valós időben történik. Emellett az eszköz erőforrásait jobban kihasználva, az optimalizált modellek gyorsabb válaszidőt és hatékonyabb működést biztosítanak, miközben minimális memória és energiahasználatot igényelnek.

4.3 Használt módszerek

4.3.1 MobileNetV2 modell alkalmazása mobil eszközön

Az alkalmazásban használt MobileNetV2 modell egy előre betanított konvolúciós neurális hálózat (CNN), amelyet a képek hatékony és gyors feldolgozására terveztek, különösen mobil és beágyazott eszközökön történő futtatásra. A modell a mélytanulás (Deep Learning) egyik felügyelt tanulási (Supervised Learning) módszerét alkalmazza, ahol a betanítás során minden képhez ismert címke tartozik. A tanulási folyamat során a modell megtanulja az egyes vizuális jellemzők és a hozzájuk tartozó címkék közötti kapcsolatokat. [40]

A MobileNetV2 modellt az ImageNet adathalmazon tanították, amely egy nagy méretű adatbázis, amely több mint 14 millió képet tartalmaz, 1001 különböző kategóriában. Ez az adathalmaz különböző kategóriákat tartalmaz, mint például állatok, járművek, épületek, növények, így lehetővé téve a modell számára a képek különböző vizuális jellemzőinek megtanulását és az ezekhez tartozó címkékhez való hozzárendelést. Az osztályozás felügyelt tanulással történik, ahol a modell a címkézett adatokat használja az egyes képekhez tartozó osztályok megtanulásához. A tanulás során a MobileNetV2 a softmax aktivációs függvényt alkalmazza, amely a valószínűségek alapján határozza meg, hogy az adott képet melyik osztályba sorolja, figyelembe véve az egyes kategóriák közötti valószínűségeket. Ezáltal a modell képes az objektumok felismerésére és gyors osztályozására, azaz a képek gyors és hatékony kategorizálására. [41, 42]

A MobileNetV2 modellt optimalizált formátumban .tflite kiterjesztéssel integráltam az alkalmazásba. Az alkalmazásban a modell akkor kerül alkalmazásra, amikor a felhasználó új eseményt hoz létre, és ehhez képet is csatol (4.3 ábra).

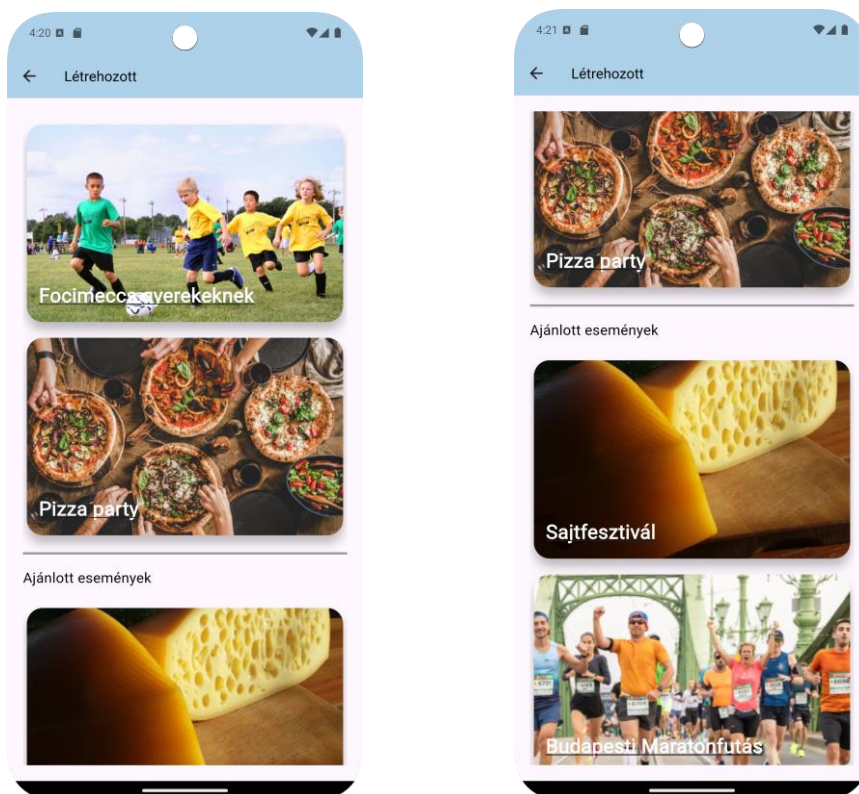
4.3 Az alkalmazás új esemény képernyője

A modell a feltöltött képekből releváns vizuális jellemzőket nyer ki, és ezek alapján az eseményeket automatikusan kategóriákba sorolja, például „Sport”, „Zene” vagy „Művészet”, a képen felismerhető tárgyak alapján. A kategorizálás egy előre meghatározott címke–kategória leképezés szerint történik. A leképezések egy részét a 4.4. ábra szemlélteti.

```
void initializeLabelMapping() {
    labelToEventCategory = {
        'baseball': 'Sport',
        'basketball': 'Sport',
        'tennis ball': 'Sport',
        'football helmet': 'Sport',
        'soccer ball': 'Sport',
        'rugby ball': 'Sport',
        'golf ball': 'Sport',
        'golf cart': 'Sport',
        'sports car': 'Sport',
        'snowmobile': 'Sport',
        'volleyball': 'Sport',
        'swimming trunks': 'Sport',
        'accordion': 'Music',
        'acoustic guitar': 'Music',
        'bassoon': 'Music',
        'cello': 'Music',
        'drum': 'Music',
        'drumstick': 'Music',
        'electric guitar': 'Music',
    }
}
```

4.4 ábra – Az alkalmazásban használt címke-kategória leképezés

A felhasználó által létrehozott események egy külön felületen jelennek meg az alkalmazásban, ahol az egyes események alatt megjelennek az automatikusan generált ajánlott események is (4.5 ábra).



4.5 ábra – Az alkalmazásban megjelenített ajánlott események MobileNetV2 modell használatával

Ezek az ajánlások dinamikusan frissülnek az események kategóriája alapján, amelyet a feltöltött kép tartalma határoz meg. Az alkalmazás olyan további eseményeket jelenít meg, amelyek azonos kategóriába tartoznak, így tematikailag illeszkednek az adott eseményhez. Ez lehetővé teszi, hogy a felhasználók könnyedén felfedezzenek más, hasonló érdeklődési körbe tartozó programokat anélkül, hogy manuálisan kellene keresniük. Ennek eredményeként az alkalmazás személyre szabottabb és felhasználóbarátabb élményt nyújt.

4.3.2 Generatív nyelvi modell alkalmazása

Az alkalmazás egyik különleges funkciója a generált szövegek automatikus előállítás, amelyhez a Puli GPT-2 nevű, magyar nyelvre finomhangolt generatív nyelvi modellt használtam. A modell a GPT-2 (Generative Pretrained Transformer 2) architektúrán

alapul, amelyet a Puli AI csapat adaptált és továbbfejlesztett magyar nyelvi környezetre, nagy mennyiségű magyar szövegen történő tanítással. [43, 44]

A fejlesztésem során a Google Colab felhőalapú környezetében promptok segítségével generáltam leírásokat előre megadott eseménynemekhez a modell felhasználásával (5.11 ábra). Az elkészült leírásokat egy JSON fájlba mentettem, ahol minden eseménynévhez hozzárendeltem a hozzá tartozó leírást egy kulcs-érték párokból álló struktúrában. Az alkalmazás ezeket az előre generált leírásokat használja fel azokban az esetekben, amikor a felhasználó új eseményt hoz létre, de nem ad meg saját leírást. A 4.6. ábrán látható, amikor a felhasználó nem ad meg leírást az új esemény létrehozásakor.

4.6 ábra - Az alkalmazás új esemény képernyője

Ilyenkor a rendszer fuzzy szövegillesztési módszerrel megkeresi az eseménynévhez leginkább hasonlító előre generált esemény nevet, és ha talál megfelelő egyezést, automatikusan beilleszti annak leírását a mezőbe. Ha nincs megfelelő találat, a rendszer hibaüzenetet jelenít meg, amely figyelmezteti a felhasználót, hogy a leírás nem maradhat üresen. A kódban a `StringSimilarity.findBestMatch()` függvény végzi az eseménynév és az elérhető eseménykulcsok közötti összehasonlítást. A `StringSimilarity.findBestMatch()` függvény a Dart `string_similarity` csomag része, amely a szövegek közötti hasonlóságok mérésére szolgál. A függvény a Jaro-Winkler

algoritmus alkalmazásával végzi el a szövegek közötti fuzzy szövegillesztést. Ez az algoritmus figyelembe veszi a karakterek közötti sorrendet, az egyező karaktereket és a közvetlen eltéréseket, miközben a karakterek közötti kisebb különbségeket, mint az elgépeléseket vagy elírásokat is felismeri. [45, 46, 47]

`StringSimilarity.findBestMatch()` függvény visszaadja a legjobb egyezést, amelyhez egy rating érték (hasonlósági pontszám) is tartozik. Amennyiben ez az érték 0.4-nél kisebb, a találat nem számít elég jónak, így nem rendel leírást az eseményhez (4.7 ábra).

```
String? getDescriptionForEvent(String eventName) {  
    if (eventDescriptions.isEmpty) {  
        return null;  
    }  
    var bestMatch = StringSimilarity.findBestMatch(eventName, eventDescriptions.keys.toList());  
    if (bestMatch.bestMatch.rating! < 0.4) {  
        return null;  
    }  
    final bestMatchName = bestMatch.bestMatch.target;  
    return eventDescriptions[bestMatchName];  
}
```

4.7 ábra - Az alkalmazásban használt fuzzy szövegillesztés

Ez a megközelítés rugalmasabbá teszi a rendszer használatát, hiszen nem szükséges az eseményneveket pontosan megadni elég, ha azok kellően hasonlítanak egy már ismert eseményre.

5. Arcfelismerés

5.1 Arcfelismerés történeti áttekintése

Az arcfelismerés kutatása a 20. század közepén indult, amikor megjelentek az első matematikai modellek és algoritmusok, amelyek célja az arcok azonosítása volt. Az 1960-as években kezdődtek az első kísérletek, de akkoriban az arcfelismerő rendszerek még meglehetősen korlátozottak voltak és nem rendelkeztek nagy pontossággal. A 1990-es évek közepére, a számítógépes látás fejlődése és a gépi tanulás elterjedése révén jelentős előrelépések történtek, és az olyan arcfelismerő algoritmusok, mint az eigenfaces, egyre megbízhatóbbá váltak. [48]

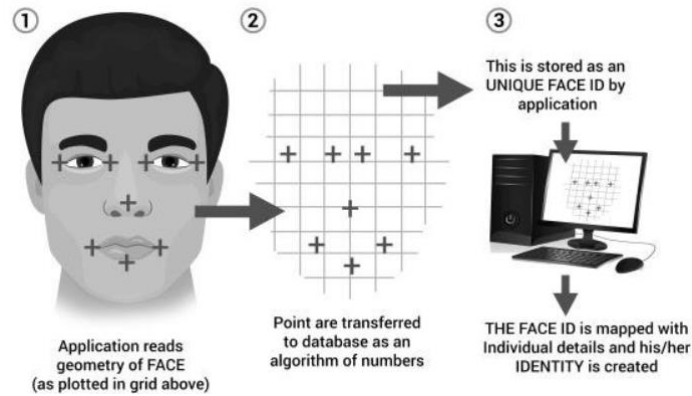
A 2000-es évek elejére az arcfelismerési technológia szélesebb körben elterjedt, és számos iparági alkalmazás vált elérhetővé, például biztonsági rendszerekben, biometrikus azonosításban és arcfelismerő szoftverekben, melyeket okostelefonokban és közlekedési rendszerekben is használtak. A legújabb áttörés a mélytanulási algoritmusok bevezetésével történt, amelyek jelentősen növelték az arcfelismerés pontosságát. A konvolúciós neurális hálózatok (CNN) alkalmazása lehetővé tette, hogy a rendszerek automatikusan tanuljanak az arcok jellemzőiről a képeken, így a felismerés még gyorsabbá és megbízhatóbbá vált. [49]

Manapság az arcfelismerés széleskörűen használt technológia, amely jelen van a mobiltelefonokban, közlekedési rendszerekben, biztonsági intézkedésekben és számos más iparági alkalmazásban.

5.2 Arcfelismerési rendszerek típusai

Az arcfelismerési rendszerek különböző technológiai megközelítéseken alapulnak, amelyek különböző módon végzik el az arcok azonosítását és elemzését. Ezek a rendszerek különböző módszereket alkalmaznak, amelyek az adatforrástól és az alkalmazási környezettől függően változnak. A legelterjedtebb típusok közé tartoznak a következők: [50]

- 1) Geometriai alapú rendszerek: Ezek a rendszerek az arc jellegzetes geometriai vonásait, mint például a szemek, orr és száj elhelyezkedését elemzik. Az arc felismerése a jellemzők, mint például a távolságok és arányok pontos mérésén alapul, és ezek az adatok alapjául szolgálnak az arcnak a tanult mintákkal való összehasonlítására. (5.1 ábra) [51]

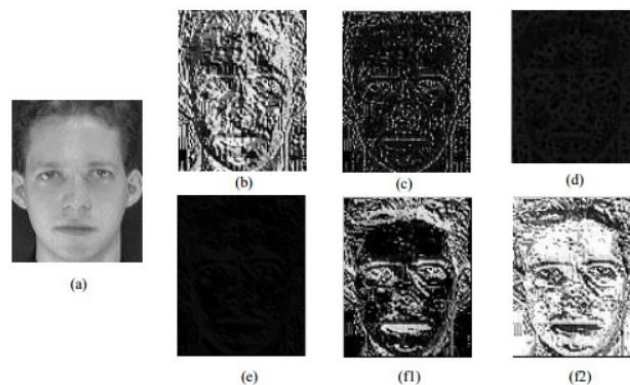


5.1 ábra - Geometriai alapú arcfelismerő rendszerek működése

2) Textúra-alapú rendszerek: Az arc bőrének textúráját és mintázatait elemzik, figyelembe véve a finom részleteket, mint például a pórusokat, ráncokat és egyéb bőrszerkezeti jellemzőket. Ezek a rendszerek gyakran használnak lokális bináris minták (LBP) vagy Gabor-szűrők alkalmazásával kinyert jellemzőket, amelyek az arc textúrájának egyedi mintázatát kódolják.

Az LBP egy egyszerű, mégis hatékony textúra-analízis módszer, amely minden egyes pixel környezetében bináris mintát generál a szomszédos pixelek intenzitásának összehasonlításával, így létrehozva egy jellemző vektort, amely jól reprezentálja a textúrabeli különbségeket. A Gabor-szűrők képesek érzékelni a képen található mintázatokat különféle irányokban és részletességben, így hatékonyan azonosítják az arc szerkezeti jellemzőit, például a kontúrokat. [52, 53, 54]

Az ilyen megközelítések különösen hasznosak lehetnek olyan helyzetekben, ahol az arc részleges elforgatása vagy elmozdulása történik, mivel a bőrminták stabilabbak lehetnek a geometriai jellemzőkhöz képest. (5.2 ábra)

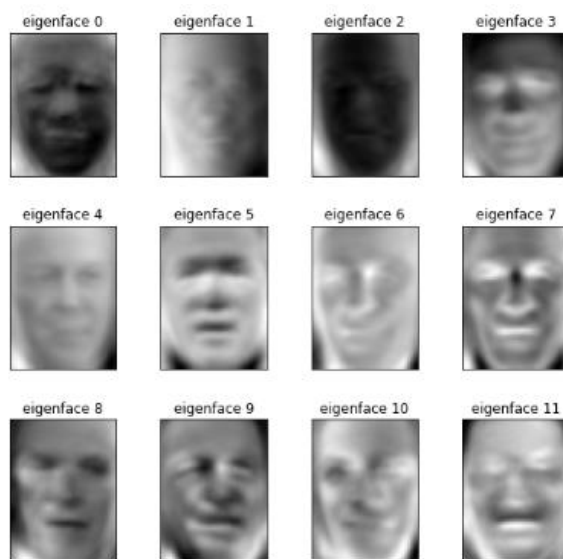


5.2 ábra - LBP alkalmazása arcképeken

- 3) Statisztikai alapú rendszerek: Ezek a rendszerek statisztikai módszereket alkalmaznak, mint például az eigenfaces-t, hogy matematikai modelleket alkossanak az arcok jellemzőiről. Az eigenfaces egy arcfelismerési technika, amely az arcképek matematikai reprezentációján alapul. A módszer az arcképek közös jellemzőit (azaz az "eigenfaces"-eket) használja az arcok azonosításához.

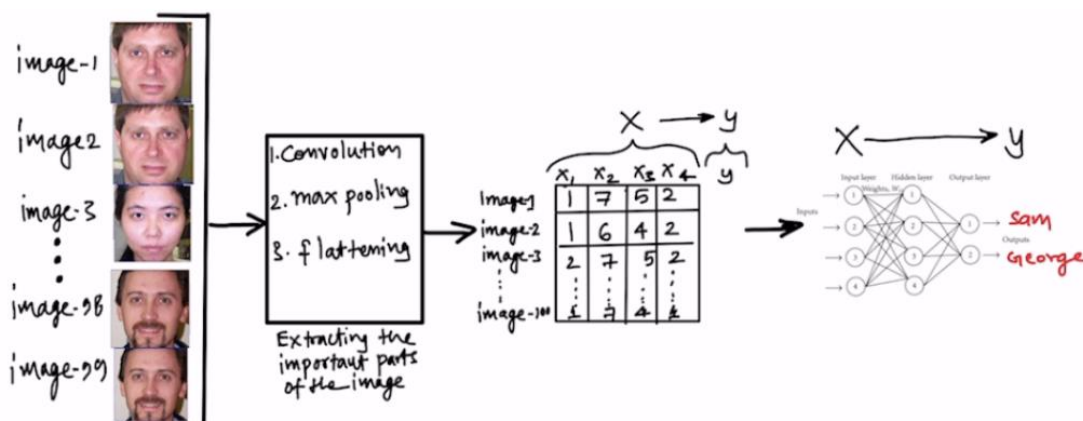
Az eigenfaces technika lényege, hogy a képeket főkomponens-analízissel (PCA) dekomponálja, így a magas dimenziójú képadatokat egy alacsonyabb dimenziós térben ábrázolja, amely az arcok megkülönböztetéséhez szükséges legfontosabb információkat tartalmazza (5.3 ábra).

Az arcokat jellemző adatokat, például a szemek közötti távolságot, statisztikai eloszlások segítségével elemzik. Ezáltal az arcfelismerés feladata egy vektorokkal végzett összehasonlításra egyszerűsödik. [55, 56]



5.3 ábra - Eigenfaces alkalmazása arcképeken

- 4) Mélytanulás alapú rendszerek: A mélytanuláson alapuló rendszerek, különösen a konvolúciós neurális hálózatok (CNN), közvetlenül a képadatokból tanulják meg az arcok felismerését, anélkül hogy előre meghatározott jellemzők azonosítására lenne szükség (5.4 ábra).

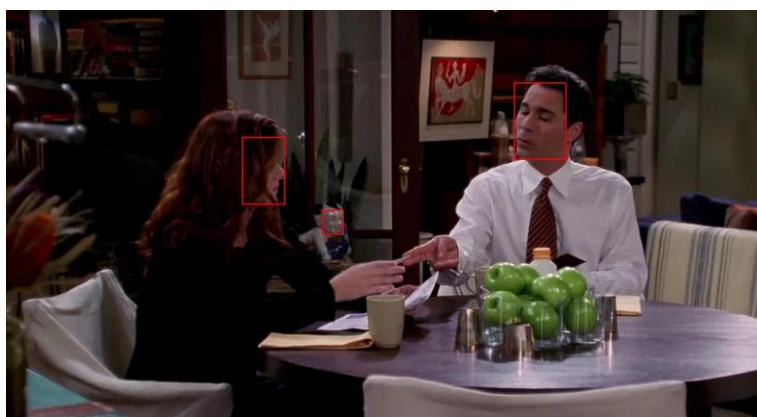


5.4 ábra - Mélytanulás alapú arcfelismerő rendszer működése

Ezek a rendszerek nagy mennyiségű adat feldolgozására képesek, és összetett mintázatok felismerésére is alkalmasak, például az arc különböző vonásainak, arckifejezéseinek és a megvilágítási különbségeknek a kezelésére. [57, 58]

5.2.1 Arcdetektálás

Az arcdetektálás egy olyan technológiai folyamat, amelynek célja az emberi arcok automatikus azonosítása képeken, videókon vagy élő kamerafelvételeken (5.5 ábra). Ez a folyamat nem tévesztendő össze az arcfelismeréssel, mivel itt a rendszer kizárólag az arcok jelenlétét és helyzetét határozza meg, anélkül hogy az egyes arcokat konkrét személyekhez kapcsolná. [59]



5.5 ábra - Arcdetektálás

Az arcdetektáló szoftverek mesterséges intelligenciát és gépi tanulási algoritmusokat, valamint statisztikai elemzést és képfeldolgozást használnak annak érdekében, hogy az emberi arcokat felismerjék nagyobb képeken, és elkülönítsék őket az arcokon kívüli objektumoktól, mint például tájak, épületek vagy más emberi testrészek. Az arcdetektálás megkezdése előtt az elemzett média előfeldolgozása

történhet a minőség javítása érdekében, valamint a felismerést zavaró objektumok eltávolítása.

Az arcdetektáló algoritmusok jellemzően az emberi szemek keresésével kezdődnek, mivel ezek az egyik legegyszerűbben felismerhető jellemzők. Ezt követően próbálják azonosítani az arc egyéb jellemzőit, mint például a szemöldököt, a száját, az orrot, és az orrlyukakat. Miután az algoritmus úgy ítéli meg, hogy arcot talált, további teszteket végez annak megerősítésére, hogy valóban arcot észlelt.

A nagy pontosság biztosítása érdekében az algoritmusokat nagy adatbázisokon tanítják, amelyek több százezer pozitív és negatív képet tartalmaznak. A tanulás javítja az algoritmusok képességét abban, hogy meghatározzák, van-e arc a képen, és pontosan hol helyezkednek el azok határai.

Az arcdetektálásra számos szoftver létezik. Ezek a szoftverek folyamatosan fejlődtek, miközben egyre fejlettebb technológiákat integráltak a pontosabb eredmények és jobb teljesítmény elérése érdekében.

Az egyik első sikeres módszer a Viola-Jones algoritmus volt, amely egy modellt tanít arra, hogy miként lehet megkülönböztetni az arcokat az egyéb objektumoktól. Bár ez a megközelítés még mindig széles körben használatos valós idejű arcfelismerési alkalmazásokban, problémákat okoz, ha az arcok takarva vannak, vagy ha nem megfelelő irányban vannak elhelyezve.

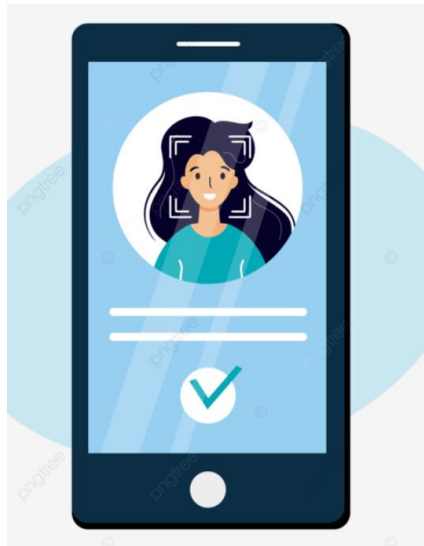
A mesterséges intelligencia (AI) technológiák fejlődésével az arcdetektálás képességei is jelentősen javultak. Manapság ezen rendszerek gépi tanulást és mélytanulást alkalmaznak az arcok felismerésére. Emellett konvolúciós neurális hálózatokat (CNN) is alkalmaznak, amelyek a képi adatokat elemzik, és lineáris algebrát használva azonosítják a képen lévő mintázatokat és jellemzőket.

Az egyik CNN-alapú arcdetektálás egyik megoldása a Regionális CNN (R-CNN), amely lokalizálja és osztályozza az objektumokat a képeken, majd javaslatokat generál a potenciális arcvonásokra. Ezek a javaslatok a kép olyan területeire összpontosítanak, amelyek hasonlítanak más területekhez, például a szemek pixelált területeihez. Ha a szemek régiója egyezik egy másik szem régióval, az algoritmus felismeri a találatot. A legújabb arcdetektáló megközelítések közé tartozik a Fast R-CNN és a Faster R-CNN is.

Az R-CNN és annak továbbfejlesztett változatai egyik legnagyobb problémája, hogy bonyolulttá válhatnak, és hajlamosak túltanulásra. A CNN-alapú megközelítések másik problémája, hogy gyakran szűk keresztmetszetekbe ütköznek, mivel a rendszer

kétszer fut végig a CNN-en: először a régiók javaslatainak generálására, majd az objektumok detektálására.

A Single Shot Detector (SSD) módszer segít orvosolni ezt a problémát, mivel csak egyszer fut át a hálózaton a képen lévő objektumok felismeréséhez. Ennek eredményeként az SSD gyorsabb, mint az R-CNN, de hátránya, hogy nehezebben képes kis arcokat vagy távolról lévő arcokat felismerni. [60]



5.6 ábra - Arcdetektálás okostelefonoknál

Az okostelefonok és táblagépek gyakran használják az arcdetektálást a kamerák autofókusz funkciójának aktiválására fényképezéskor és videózáskor. Emellett számos mobil eszköz helyettesíti a jelszavakat vagy PIN-kódokat arcdetektálással. Például az Apple iPhone képes arcdetektálást alkalmazni a telefon feloldásához, ha egy engedélyezett felhasználó próbál hozzáférni (5.6 ábra).

5.2.2 Arcazonosítás

Az arcazonosítás egy olyan biometrikus technológia, amely lehetővé teszi egy személy egyedi azonosítását az arcvonásai alapján. A folyamat célja, hogy meghatározza, ki van a képen vagy videón, azáltal, hogy az adott arcot összehasonlítja egy előre rögzített adatbázisban tárolt arcokkal. Ez különbözik az arcdetektálástól, amely csak észleli, hogy egy arc jelen van a képen, de nem feltétlenül próbálja megmondani, ki az adott személy.

Az arcazonosító rendszerek először detektálják az arcot egy képen vagy videóban, majd kivonják annak jellegzetes vonásait (például: szemek közötti távolság, orr formája, száj elhelyezkedése). Ezeket az arcvonásokat ezután numerikus jellemzőkké

(embeddingekké) alakítják, amelyeket össze lehet vetni a rendszerben tárolt ismert személyek jellemzőivel. [61]

5.2.3 Arcverifikáció

Az arcverifikáció egy biometrikus azonosítási folyamat, amelynek célja annak ellenőrzése, hogy egy adott személy valóban ő-e, az arca alapján. A folyamat során az arckép, amelyet egy személy benyújt, összehasonlításra kerül egy előzetesen rögzített referenciaképpel, például egy adatbázisban tárolt képpel. Az arcverifikáció tehát nem az egyes személyek azonosítására szolgál, hanem annak megerősítésére, hogy az arc valóban egy előzőleg regisztrált személyhez tartozik.

Ezt a technológiát számos területen alkalmazzák, például biztonsági rendszerekben, mobiltelefonoknál (5.7 ábra), banki alkalmazásoknál, illetve olyan platformokon, amelyek a személyes adatokat védik, és lehetővé teszik a felhasználói hozzáférés engedélyezését vagy elutasítását. Az arcverifikációs rendszerek gyakran gépi tanulást és mélytanulást használnak, hogy javítsák a felismerés pontosságát és sebességét. [62, 63]



5.7 ábra - Arcverifikáció mobiltelefonon

Az arcverifikációs rendszerek folyamatosan fejlődnek, hogy még pontosabbak és megbízhatóbbak legyenek. A modern technológiák, mint a mélytanulás és a konvolúciós neurális hálózatok (CNN), lehetővé teszik a rendszer számára, hogy az arcok jellemzőit még finomabb részletekben, például a szemek közötti távolság, az orr és a száj elhelyezkedése alapján elemezze. Ezenkívül az arcverifikáció képes alkalmazkodni az arc különböző elhelyezkedéséhez, szögeihez, fényviszonyaihoz és az esetleges arckifejezésekhez is. A megfelelő környezetben és megfelelő minőségű képeken az arcverifikáció egy rendkívül gyors és hatékony módszerré válik a

személyazonosításra, mivel nem szükséges fizikai eszközöket, például kártyákat vagy jelszavakat használni.

A technológia egyre inkább elterjedt, mivel az emberek egyre inkább keresnek kényelmesebb és biztonságosabb módokat a digitális eszközökhöz való hozzáféréshez. A bankok, pénzügyi szolgáltatók és mobiltelefon-gyártók is egyre inkább integrálják az arcverifikációs megoldásokat az ügyfelek védelme és a tranzakciók biztonságának növelése érdekében. Azonban, ahogy minden biometrikus azonosítási technológia, úgy az arcverifikáció is etikai és adatvédelmi kérdéseket vet fel. Az adatkezelés átláthatósága, a személyes adatok védelme és a felhasználói beleegyezés biztosítása kulcsfontosságú tényezők, amelyek meghatározzák a jövőbeni alkalmazások biztonságát és elfogadottságát. [64, 65]

5.2.4 Használt módszerek

Az arcfelismeréses hitelesítés megvalósításához az alkalmazásban a `local_auth` csomagot használtam, amely az eszköz operációs rendszerének beépített biometrikus hitelesítési megoldásait veszi igénybe, mint az iOS-en a Face ID-t, Androidon pedig az elérhető arcfelismerő rendszert. A csomag támogatja az eszközön történő helyi autentikációt, így lehetőséget biztosít ujjlenyomat, PIN-kód, jelszó, minta vagy arcfelismerés, használatára a felhasználó azonosításához (5.8 ábra).

```
Future<UserModel> authenticateUser(UserModel userModel) async {
  bool isAuthorized = false;
  try {
    errorMessages = [];
    isAuthorized = await localAuthentication.authenticate(
      localizedReason: faceIdAuthentication,
    );
  } catch (e) {
    if (e.toString().isNotEmpty) {
      errorMessages = [e.toString()];
    } else {
      errorMessages = [standardErrorMessage];
    }
  }
  if (isAuthorized) {
    isUserAuthorized = true;
    String? userId = userLoginSingleton.getRememberedUserId();
    String? userJson = userLoginSingleton.getRememberedUserData();
    String? token = userLoginSingleton.getRememberedUserToken();
    if (userJson != null && userId != null && token != null) {
      UserDTO userDTO = UserDTO.fromJson(jsonDecode(userJson), userId);
      userModel = UserModel.fromDTO(userDTO);
    }
  } else {
    isUserAuthorized = false;
  }
  return userModel;
}
```

5.8 ábra - Az alkalmazásban használt arcfelismerés autentikáció

A Face ID az Apple által fejlesztett korszerű arcfelismerő technológia, amelyet az iPhone és iPad Pro készülékeken alkalmaznak a biztonságos és felhasználóbarát azonosítás érdekében. A rendszer központi eleme a TrueDepth kamera, amely több összetevőből áll: egy infravörös pontprojektor több mint 30 000 láthatatlan pontot vetít az arcra, egy infravörös kamera rögzíti ezek elhelyezkedését, és egy infravörös megvilágító biztosítja a megfelelő fényviszonyokat. Ezek az adatok egy háromdimenziós térképet alkotnak az arcról, amelyet a készülék biztonságos alrendszere, a Secure Enclave dolgoz fel és hasonlít össze a korábban tárolt mintával. A Face ID működése során matematikai modelleket használ az arcfelismeréshez. Az infravörös képek és pontminták alapján egy matematikai modellt hoz létre az arcról, amelyet összevet a regisztrált adatokkal. Ez a megközelítés lehetővé teszi a rendszer számára, hogy alkalmazkodjon az arc megjelenésének változásaihoz. [66, 67, 68]

Az Androidos arcfelismeréses hitelesítés a BiometricPrompt API használatával történik, és a működése attól függ, hogy az adott eszköz milyen hardverrel és szoftverrel rendelkezik. Az arcfelismerést két fő kategóriába sorolhatjuk: 2D-alapú (kameraalapú) és 3D-alapú (mélységérzékelős) megoldásokra.

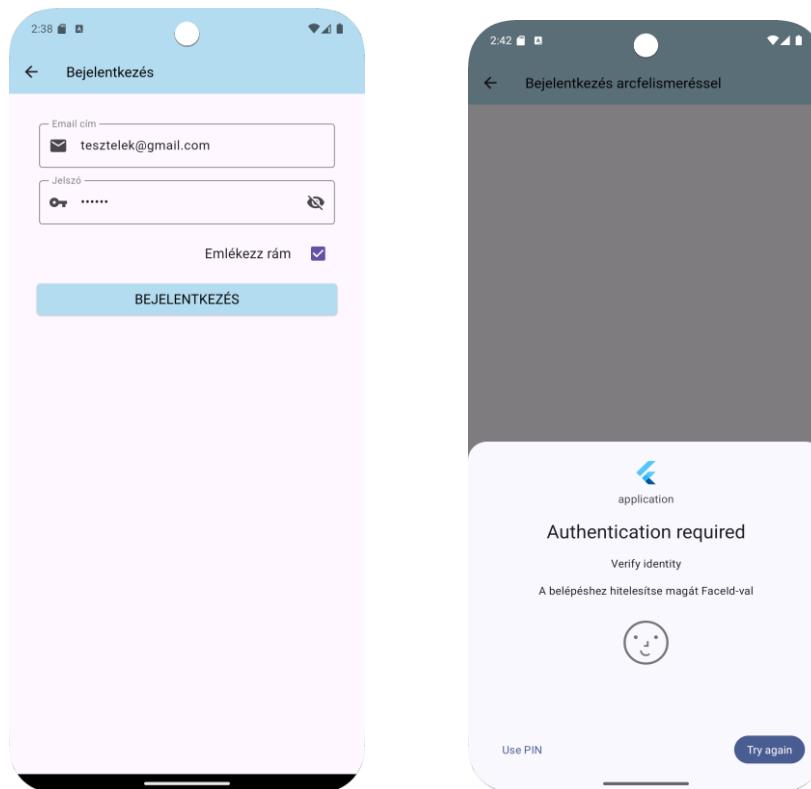
- 2D-alapú arcfelismerés hagyományos kamerát használ az arc kétdimenziós képének rögzítésére. Ez a módszer még nem használ mesterséges intelligenciát a felismerés során, helyette az algoritmusok az aktuális arckép és az adatbázisban tárolt képek pixelszintű mintázatait hasonlítják össze. Bár gyors, de kevésbé biztonságos, és könnyebben megtéveszthető például egy fényképpel. A 2D-s arcfelismeréses modellek gyakran egyszerűbb képfeldolgozási eljárásokat alkalmaznak, mint például a Haar Cascade Classifier vagy Eigenfaces.
- 3D-alapú arcfelismerés speciális szenzorokat, például infravörös kamerát és mélységérzékelőt alkalmaz, hogy az arc térbeli szerkezetét is figyelembe vegye, ezáltal nagyobb biztonságot nyújt. A 3D-s arcfelismerési megoldások általában mélytanulási modelleket, például konvolúciós neurális hálózatokat (CNN) használnak. Az ilyen típusú rendszerek például a Google Pixel 4-en alkalmazott 3D mélységérzékelést és infra kamerát, amelyek egy neurális hálózat segítségével elemzik az arc jellemzőit és azok térbeli elhelyezkedését. A neurális hálózat az arcról kinyert jellemzők alapján egy embeddinget állít, amely az adott arc legfontosabb és legjellemzőbb tulajdonságait foglalja össze egy többdimenziós térben. A telefon ezt a vektort összeveti a korábban elmentett,

hitelesített mintával. Ha a két vektor közötti különbség egy meghatározott küszöbértéken belül van, a rendszer az arcot egyezőnek ismeri fel, és engedélyezi a hozzáférést. [69, 70, 71]

Ezek a 3D-s megoldások fejlettebbek és kevésbé érzékenyek a manipulációkra, mint a 2D-s modellek, mivel az arc háromdimenziós struktúráját is figyelembe veszik. A mélytanulási modellek (például CNN) lehetővé teszik a pontosabb arcazonosítást a bonyolultabb, több jellemzőt figyelembe vevő algoritmusokkal. Ezáltal az Androidos arcfelismerés rendszerei a készülék típusától és az alkalmazott hardver-támogatástól függően különböző modelleket használhatnak, mindkét kategóriában alkalmazva a megfelelő biometrikus technológiai megoldásokat a biztonságos és gyors autentikáció érdekében. [72, 73]

A mobilalkalmazás multiplatform kialakításának köszönhetően mind iOS, mind Android rendszereken működik, és képes kihasználni a platformok beépített arcfelismerő (biometrikus) hitelesítési megoldásait. Ennek köszönhetően a hitelesítés gyors, kényelmes és biztonságos, így a felhasználók könnyedén kihasználhatják a natív biometrikus azonosítás előnyeit a bejelentkezéskor.

Az alkalmazásban az arcfelismeréses belépés úgy működik, hogy amennyiben a felhasználó a bejelentkezés során aktiválja az „Emlékezz rám” funkciót, az alkalmazás biztosítja, hogy legközelebb, amikor újra megnyitja azt, ne kelljen ismét bejelentkeznie (5.9 ábra). Ha az alkalmazás bezárása után nem jelentkezik ki, a következő alkalommal az arcfelismerés segítségével automatikusan hozzáférhet az alkalmazáshoz.



5.9 ábra - Az alkalmazás bejelentkezés arcfelismeréssel felülete

Ez a megoldás lehetővé teszi, hogy a felhasználók gyorsan és egyszerűen hozzáférjenek az alkalmazáshoz, miközben biztosítja a kényelmes és biztonságos autentikációt. A biometrikus azonosítás segítségével az alkalmazás automatikusan felismeri a felhasználót, így a belépés minden alkalommal zökkenőmentes és gyors. Ezáltal a felhasználóknak nem kell minden alkalommal újra megadniuk a jelszavukat, hiszen a Face ID vagy arcfelismerés gondoskodik a hitelesítésről.

Az elkészült dolgozat elérhető az alábbi GitHub oldalon:

<https://github.com/krekab16/Szakdolgozat-MSc>

Az dolgozathoz használt automatizáció:

<https://github.com/krekab16/CosineEmbeddingAutoRun>

A dolgozat colab notebookja:

https://colab.research.google.com/drive/1E6LTGMmS_wGOJ4IggeBOj9_AmJnGiBvT?usp=sharing

Irodalomjegyzék

- [1] Gépi tanulás a modern technológiában - <https://kangadesign.hu/gepi-tanulas.html>
- [2] Az arcfelismerés a modern mobiltechnológiában - <https://sciamus.hu/2020/01/14/az-arcfelismeres-a-modern-mobiltechnologia-legnagyobb-tevedese/>
- [3] Mi az a gépi tanulás? - <https://www.coursera.org/articles/what-is-machine-learning>
- [4] A gépi tanulás meghatározása- <https://itszotar.hu/2024/10/mi-a-gepi-tanulas/>
- [5] Machine Learning Algorithms - <https://www.geeksforgeeks.org/machine-learning-algorithms/>
- [6] Gépi tanulási algoritmusok - <https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-are-machine-learning-algorithms>
- [7] What is a machine learning algorithm? - <https://www.ibm.com/topics/machine-learning-algorithms>
- [8] AI and Self-Driving Cars - <https://www.quytech.com/blog/ai-in-self-driving-cars/>
- [9] How AI is Revolutionizing Self-Driving Cars - https://hackertraining.org/ai_research/ML_Fundamentals/ai_generated/data/use_cases/Self-Driving_Cars_usecase/
- [10] How Netflix and Amazon Use Deep Learning to Enhance User Experience - <https://medium.com/@zhonghong9998/personalized-recommendations-how-netflix-and-amazon-use-deep-learning-to-enhance-user-experience-e7bd6fcd18ff>
- [11] Recommendation Engines Using Different Technology - <https://www.argoid.ai/blog/under-the-hood-comparison-recommendation-engines-using-different-technology>
- [12] Recommendation Systems Explained: Understanding the Basic to Advance - <https://utsavdesai26.medium.com/recommendation-systems-explained-understanding-the-basic-to-advance-43a5fce77c47>
- [13] Ajánlórendszerek a gyakorlatban - <https://onespire.hu/ajanlorendszerek-a-gyakorlatban/>
- [14] Collaborative Filtering in Recommender System - <https://medium.com/@evelyn.eve.9512/collaborative-filtering-in-recommender-system-an-overview-38dfa8462b61>
- [15] Recommender Systems: User-Based and Item-Based Collaborative Filtering - <https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f>
- [16] User-Based Collaborative Filtering - <https://www.geeksforgeeks.org/user-based-collaborative-filtering/>
- [17] Item Based Collaborative Filtering - <https://www.geeksforgeeks.org/item-to-item-based-collaborative-filtering/>
- [18] What is content-based filtering? - <https://www.ibm.com/think/topics/content-based-filtering>
- [19] Recommendation Systems: Content-Based Filtering - <https://medium.com/@zbeyza/recommendation-systems-content-based-filtering-e19e3b0a309e>
- [20] Content Based Recommender System - <https://www.geeksforgeeks.org/ml-content-based-recommender-system/>

- [21] What is Hybrid Recommender Systems - <https://marketsy.ai/blog/hybrid-recommender-systems-beginners-guide>
- [22] Hybrid Recommendation Systems - <https://umair-iftikhar.medium.com/part-4-building-hybrid-recommendation-systems-bbe1c8b77552>
- [23] 7 Types of Hybrid Recommendation System - <https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8>
- [24] What is Cosine Similarity - <https://www.datastax.com/guides/what-is-cosine-similarity>
- [25] Cosine Similarity - <https://www.geeksforgeeks.org/cosine-similarity/>
- [26] Understanding TF-IDF - <https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/>
- [27] Understanding TF-IDF for Machine Learning - <https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/>
- [28] Understanding GitHub Actions - <https://docs.github.com/en/actions/about-github-actions/understanding-github-actions>
- [29] GitHub Actions - <https://medium.com/@dmosyan/understanding-the-basics-of-github-actions-7787993d300c>
- [30] Mély gépi tanulás (deep learning) - https://www.inf.u-szeged.hu/~rfarkas/ML20/deep_learning.html
- [31] What is deep learning? - <https://www.ibm.com/think/topics/deep-learning>
- [32] Introduction to Deep Learning - <https://www.geeksforgeeks.org/introduction-deep-learning/>
- [33] Layers in Artificial Neural Networks - <https://www.geeksforgeeks.org/layers-in-artificial-neural-networks-ann/>
- [34] Introduction to deep learning on mobile - <https://medium.com/nanogiants/introduction-to-deep-learning-on-mobile-a176c6a8d356>
- [35] LiteRT overview - <https://ai.google.dev/edge/litert>
- [36] Model Optimization Techniques (Pruning, Quantization, Knowledge Distillation, Sparsity, OpenVino Toolkit) - https://medium.com/@VK_Venkatkumar/model-optimization-techniques-pruning-quantization-knowledge-distillation-sparsity-2d95aa34ea05
- [37] What is model optimization? - <https://www.ultralytics.com/blog/what-is-model-optimization-a-quick-guide>
- [38] OpenVino - <https://docs.openvino.ai/2025/index.html>
- [39] Core ML - <https://developer.apple.com/documentation/coreml/>
- [40] What Is Mobilenet V2? - <https://www.geeksforgeeks.org/what-is-mobilenet-v2/>
- [41] MobileNetV2 - <https://arxiv.org/abs/1801.04381>
- [42] Image Classification Model MobileNet V2 - <https://medium.com/@nutanbhogendrasharma/image-classification-model-mobilenet-v2-from-tensorflow-hub-8191b28a202a>
- [43] What is a Generative Model? - <https://developers.google.com/machine-learning/gan/generative>
- [44] PULI GPT2 - chrome-extension://efaidnbmnnnibpcapglefindmkaj/https://acta.bibl.u-szeged.hu/78417/1/msznykonf_019_247-262..pdf
- [45] String-similarity - https://pub.dev/documentation/string_similarity/latest/index.html

- [46] FindBestMatch static method - https://pub.dev/documentation/string_similarity/latest/string_similarity/StringSimilarity/findBestMatch.html
- [47] Jaro-Winkler similarity - <https://www.geeksforgeeks.org/jaro-and-jaro-winkler-similarity/>
- [48] Arcfelismerés története - <https://hu.myzr-tech.com/news/face-recognition-history-27743231.html>
- [49] History of Facial Recognition - <https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facial-recognition/>
- [50] Everything You Need to Know About Face Recognition Algorithm - <https://hyperverge.co/blog/face-recognition-algorithm/>
- [51] Face Recognition using Geometric Measurements - <https://www.sciencedirect.com/science/article/pii/S2212017312006597>
- [52] Gabor Filter - https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97
- [53] Face recognition using Gabor Filters - <https://ieeexplore.ieee.org/document/4803030/metrics#metrics>
- [54] LBP & Gabor Filter - <https://medium.com/@hemapaun105/feature-extraction-techniques-lbp-gabor-filter-58e86a2df445>
- [55] Face Recognition Using Eigenfaces - <https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/>
- [56] Face Recognition using eigenfaces technique - <https://medium.com/@devalshah1619/face-recognition-using-eigenfaces-technique-f221d505d4f7>
- [57] Face Recognition Using CNN - <https://medium.com/@raguwing/face-recognition-using-cnn-architecture-in-python-f3c302c2164f>
- [58] Facial recognition - <https://datascientest.com/en/facial-recognition-how-it-works-2>
- [59] Face Detection - <https://www.innovatrics.com/glossary/face-detection/>
- [60] What is face detection and how does it work? - <https://www.techtarget.com/searchenterpriseai/definition/face-detection>
- [61] Face Identification - <https://www.innovatrics.com/glossary/face-identification/>
- [62] Face Verification - <https://alicebiometrics.com/en/face-verification-vs-face-recognition/>
- [63] What is Biometric Verification? - <https://hyperverge.co/blog/biometrics-identity-verification-system/>
- [64] Face verification using convolutional neural networks - https://www.researchgate.net/publication/321407418_Face_verification_using_convolutional_neural_networks_with_Siamese_architecture
- [65] Face verification - <chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://archive.legmt.gov/content/Committees/Interim/2021-2022/Economic%20Affairs/Meetings/November%202021/Facial-verification-vs-facial-identification.pdf>
- [66] Face ID & Privacy - <https://www.apple.com/legal/privacy/data/en/face-id/>
- [67] Tudnivalók a Face ID fejlett technológiájáról - <https://support.apple.com/hu-hu/102381>
- [68] Apple Face ID: what is it and how does it work? - <https://www.stuff.tv/features/apple-face-id-explained/>
- [69] Face authentication - <https://source.android.com/docs/security/features/biometric/face-authentication>

- [70] Biometric Authentication in Mobile Apps - <https://www.dogtownmedia.com/biometric-authentication-in-mobile-apps-fingerprints-face-id-and-beyond/>
- [71] About face authentication - <https://developer.signicat.com/docs/mobile-identity/mobileid/features/face-authentication/>
- [72] 2D and 3D Models of Face Biometrics - <https://www.paymentsjournal.com/2d-and-3d-models-of-face-biometrics/>
- [73] 3D and 2D face recognition - <https://mxface.ai/blog/3D-face-recognition-VS-2D-face-recognition>

```

Future<List<EventDTO>> getRecommendedEventsForUser(String userId) async {
    try {
        final userEventsSnapshot = await _firestore
            .collection('events')
            .where('participants', arrayContains: userId)
            .get();
        final List<EventDTO> userEvents = [];
        for (var doc in userEventsSnapshot.docs) {
            final data = doc.data();
            final event = EventDTO.fromJson(data, doc.id);
            userEvents.add(event);
        }
        final filteredUserEvents = userEvents
            .where((e) => e.embeddingVector != null && e.embeddingVector!.isNotEmpty)
            .toList();
        if (filteredUserEvents.isEmpty) return [];
        List<double> averageVector = List.filled(filteredUserEvents[0].embeddingVector!.length, 0.0);
        for (var event in filteredUserEvents) {
            for (int i = 0; i < event.embeddingVector!.length; i++) {
                averageVector[i] += event.embeddingVector![i];
            }
        }
        averageVector = averageVector.map((val) => val / filteredUserEvents.length).toList();
        final allEventsSnapshot = await _firestore.collection('events').get();
        final List<EventDTO> allEvents = [];
        for (var doc in allEventsSnapshot.docs) {
            final data = doc.data();
            final event = EventDTO.fromJson(data, doc.id);
            allEvents.add(event);
        }

        final List<Map<String, dynamic>> scoredEvents = [];
        for (var event in allEvents) {
            final bool isAlreadyAttended = userEvents.any((e) => e.id == event.id);
            final bool hasValidEmbedding = event.embeddingVector != null &&
                event.embeddingVector!.isNotEmpty &&
                event.embeddingVector!.length == averageVector.length;
            if (!isAlreadyAttended && hasValidEmbedding) {
                final ratingsSnapshot = await _firestore
                    .collection('ratings')
                    .where('eventId', isEqualTo: event.id)
                    .get();
                final ratings = ratingsSnapshot.docs
                    .map((doc) => (doc.data()['rating'] as num?)?.toDouble())
                    .whereType<double>()
                    .toList();
                if (ratings.isNotEmpty) {
                    final avgRating = ratings.reduce((a, b) => a + b) / ratings.length;

                    if (avgRating > 3.0) {
                        final similarity = cosineSimilarity.cosineSimilarity(
                            averageVector,
                            event.embeddingVector!,
                        );
                        scoredEvents.add({
                            'event': event,
                            'similarity': similarity,
                        });
                    }
                }
            }
        }
    }
}

```

5.10 ábra - Az alkalmazásban használt Cosinus hasonlóságot használó ajánlás megvalósítása

```

model_name = "NYTK/PULI-GPT-2"
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)

base_events = [
    "Jazz Fesztivál", "Kolbász Fesztivál", "Virágkarnevál", "Színházi előadás", "Bábszínház",
    "Sportverseny", "Kézműves workshop", "Karácsonyi vásár", "Bor fesztivál", "Múzeumi Tárlatvezetés",
    "Operaelőadás", "Stand-up comedy show", "Gyereknapi", "Hagyományőrző fesztivál", "Gasztronómiai fesztivál",
    "Bor kóstoló", "Rock koncert", "Filmfesztivál", "Könyvvásár", "Táncház", "Szilveszteri buli",
    "Képzőművészeti kiállítás", "Virtuális valóság élmény", "Történelmi emlékhely látogatás",
    "Tudományos előadás", "Karácsonyi koncert", "Tavaszi piac", "Kézműves vásár", "Egyetemi nyílt nap",
    "Zenei workshop", "Képzőművészeti workshop", "Légibemutató", "Vízilabda mérkőzés", "Futóverseny",
    "Biciklis túra", "Fesztivál utca", "Fiatal tehetségek koncertje", "Bográcsozás", "Borkóstolás",
    "Gasztronómiai túra", "Hegyi túra", "Légivásár", "Bábszínházi előadás", "Főzőverseny",
    "Közösségi táncst", "Divatbemutató", "Borkóstoló rendezvény", "Reggeli jóga", "Falkaszínház",
    "Zenei fesztivál", "Bajnokság döntő", "Felhőtúra", "Népzenei koncert", "Kertészeti kiállítás",
    "Kertészeti workshop", "Állatvédelmi rendezvény", "Kézműves termék bemutató", "Sportnap",
    "Bálnales", "Indián napok", "Őszköszöntő buli", "Hegyi futás", "Kerékpáros verseny",
    "Úszóverseny", "Képzőművészeti alkotótábor", "Filmklub", "Vadászati kiállítás", "Régészeti túra",
    "Tudományos fesztivál", "Társadalmi kezdeményezés bemutató", "Szent Iván-éji buli",
    "Hegyi túrázó verseny", "Befőzés workshop", "Fotókiállítás", "Fejlesztő workshop",
    "Interaktív színházi előadás", "Autókiállítás", "Képzőművészeti rendezvény",
    "Mesefilm maraton", "Alkotói közösség est", "Fenntartható divat nap", "Tavaszcímzés kézműves foglalkozás"
]

prompt_templates = [
    "Írj egy izgalmas leírást erről az eseményről: {}.",
    "Mutasd be részletesen, mi várható a rendezvényen: {}.",
    "Adj vonzó ismertetőt a következő eseményhez: {}.",
    "Meséld el élményszerűen, milyen a(z) {} esemény!"
]

generated_texts = []

for index, event in enumerate(base_events, start=1):
    prompt = random.choice(prompt_templates).format(event)

    inputs = tokenizer(prompt, return_tensors="pt")

    output = model.generate(
        **inputs,
        max_length=250,
        num_return_sequences=1,
        temperature=1.2,
        top_k=50,
        top_p=0.9,
        no_repeat_ngram_size=2
    )

    generated_text = tokenizer.decode(output[0], skip_special_tokens=True)

    cleaned_text = generated_text.replace(prompt, "").strip()

    generated_text = "\n".join([line.strip() for line in cleaned_text.strip().split("\n")])

    generated_texts.append(generated_text)

df = pd.DataFrame({
    "event_name": base_events,
    "description": generated_texts
})

```

5.11 ábra - Az alkalmazásban használt eseményleírás generálás megvalósítása


Nyilatkozat

Alulírott, Kun Réka Bianka programtervező informatikus MSc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Számítógépes Algoritmusok és Mesterséges Intelligencia Tanszékén készítettem, programtervező informatikus MSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Diplomamunka Repozitóriumban tárolja.

Dátum 2025. május 15.

A handwritten signature in blue ink, reading "Kun Réka Bianka".

Aláírás

Köszönetnyilvánítás

Szeretnék ezúton köszönetet nyilvánítani mindazoknak, akik támogattak abban, hogy ez a szakdolgozat elkészülhessen. Köszönettel tartozom a barátaimnak és családomnak, akik állandóan bíztattak és támogattak, amikor nehéz pillanatokat éltem át a szakdolgozat írása során.

Külön szeretnék köszönetet mondani, Kiss-Vetráb Mercedesnek, aki időt és energiát szánt rám, valamint hasznos tanácsokkal és javaslatokkal látott el. A szakdolgozat készítésének folyamata alatt sokat tanulhattam tőle és nagyra értékelem a szaktudásáért, és tapasztalatáért.

Végezetül, köszönöm mindenkinek, aki hozzájárult ahhoz, hogy ezt a szakdolgozatot elkészíthettem.