

DUNAÚJVÁROSI EGYETEM

TUDOMÁNYOS DIÁKKÖRI KONFERENCIA

**Agilis szoftverfejlesztés támogatása
mesterséges intelligencia és eye-tracking
technológiák felhasználásával**

Készítette: Vámos Balázs
negyedéves mérnökinformatikus hallgató

Témavezetők: Dr. habil Katona József, PhD
egyetemi docens, tanszékvezető



Dunaújváros, 2025

Tartalomjegyzék

1.	Bevezetés.....	1
1.1.	Kutatás feltételei.....	2
1.2.	Kutatási célkitűzései.....	2
1.3.	Kutatás módszertana	3
2.	Szemmozgáskövetést megvalósító technikák alkalmazása.....	5
2.1.	Pupilla központú szaruhártya tükröződés (PCCR) elve	5
2.2.	Pupilla megvilágítási technikák részletezése	7
2.2.1.	Világospupilla módszer.....	7
2.2.2.	Sötétpupilla-módszer.....	7
2.3.	PCCR elvet alkalmazó szemmozgáskövető eszközök csoportosítása.....	7
2.3.1.	Képernyő alapú szemmozgáskövetők áttekintése	8
3.	A mesterséges intelligencia alapjai	12
3.1.	A mesterséges intelligencia alkalmazási területei	12
4.	Mesterséges intelligencia és szemmozgás-követés integrációja	14
4.1.	A modell keretének kialakítása és célja	15
4.2.	A modell keretének komponensei	16
4.2.1.	Adatgyűjtési réteg	18
4.2.2.	Mesterséges intelligencia réteg	26
4.2.3.	Visszacsatolási réteg	30
5.	Agilis fejlesztés támogatása meglévő univerzális modellel.....	35
5.1.	Az integrálási vizsgálat lefolyásának körülményei.....	35
5.1.1.	Vizsgálati személyek.....	36
5.1.2.	Vizsgálati feladat.....	36
5.2.	Likert-skálán adott válaszok értékelése és elemzése.....	38
5.3.	Hőtérkép alapú értékelés szemmozgás paraméterek alapján	40
5.4.	Kódrefaktorálás szemmozgás paraméterek és az MI modell integrálása alapján	42

5.5.	Az MI-Modell visszajelzés hatékonyságának elemzése	42
5.5.1.	Az MI visszajelzés hatékonyságának elemzése szemmozgás paraméterek kiértékelésével.....	44
5.6.	Következtetések	46
6.	Kutatás korlátai	48
7.	Összefoglalás.....	49
8.	Irodalomjegyzék.....	51
9.	Mellékletek.....	57

Ábrajegyzék

1. ábra: A PPCR módszer működési elve	6
2. ábra: A csillogás-pupilla vektor meghatározása	6
3. ábra: Mesterséges intelligencia és az eye-tracking integrációjának keretrendszere	17
4. ábra: A kód kezdetleges változata, amit elemeztek a diákok.....	37
5. ábra: Tanuló 1. szemmozgás paraméterek alapján elkészített hő térkép	41

Táblázatjegyzék

1. táblázat: Likert-skálán adott válaszok részletezése	39
2. táblázat: Új kód értékelése Likert skálán	43
3. táblázat: Rendelkezésre álló fixációs szám adatok:	45
4. táblázat: Rendelkezésre álló átlagos fixációs időtartamra vonatkozó elérhető adatok	46

1. Bevezetés

Az agilis szoftverfejlesztési módszertanok az elmúlt évtizedekben meghatározó szereplőkké váltak a szoftverfejlesztési iparágban, mivel rugalmasan képesek kezelni a gyorsan változó piaci igényeket és ügyfélelvárásokat [1]. Az agilis módszertan legfőbb célja, hogy folyamatos visszacsatolásra és iteratív fejlesztési ciklusokra építsen, lehetővé téve a projektcsapatok számára, hogy gyorsan reagáljanak a változásokra, és hatékonyan igazodjanak a felmerülő új követelményekhez [2] [3]. Az adaptív tervezés, az inkrementális fejlesztés, valamint a folyamatos visszajelzések révén az agilis módszerek hatékonyabban képesek kezelni a komplex projektek és rendszerek fejlesztésével járó kihívásokat. Azonban az agilis módszertanok alkalmazása során továbbra is számos akadályba ütközhetünk, mint például a csapatok közötti kommunikáció és a felhasználói élmény optimalizálása, amelyek különösen nagy kihívást jelentenek az agilis csapatok számára [4] [5] [6] [7].

A mesterséges intelligencia (MI) és a szemmozgás-követése (eye-tracking) technológia alkalmazása jelentős lehetőséget kínálhat arra, hogy áthidalják ezeket a kihívásokat. Az MI, amely képes automatizálni és támogatni a döntéshozatali folyamatokat, egyre nagyobb szerepet kap a modern szoftverfejlesztési gyakorlatokban [8] [9] [10]. Az MI alkalmazásával előrejelzéseket készíthetünk, optimalizálhatjuk a fejlesztési ciklusokat, és automatizálhatjuk a tesztelést, amely jelentősen növelheti az agilis fejlesztést. Az MI által vezérelt rendszerek képesek azonosítani a gyakori hibákat a kódban, és javaslatokat tenni a kód javítására, így gyorsítva a fejlesztési folyamatokat, miközben biztosítják a magasabb kód minőséget.

Az eye-tracking technológia más perspektívából támogathatja az agilis fejlesztési folyamatokat. Ez a technológia a felhasználói szemmozgásokat rögzíti és elemzi, ami lehetővé teszi, hogy a fejlesztők mélyebb betekintést nyerjenek a felhasználói interakciókba és viselkedési mintákba. Az eye-tracking alkalmazása révén pontosan követhető, hogy a felhasználók milyen módon lépnek kapcsolatba a szoftverekkel, hol akadnak el a használat során, vagy éppen mely felületek váltják ki a legnagyobb figyelmet [11] [12]. Ezek az adatok felhasználhatók a felhasználói felületek tervezésének optimalizálására, így a szoftverek intuitívabbá és könnyebben használhatóvá válhatnak. Emellett az eye-tracking a fejlesztők szemmozgását is követheti, hogy megértsék, hogyan reagálnak a kód különböző részeire, illetve milyen kognitív terhelés alatt állnak a kódolási folyamat során. Ez különösen hasznos lehet a fejlesztési hatékonyság növelésében, hiszen segít azonosítani azokat a területeket, ahol a fejlesztők több időt és energiát fordítanak a problémák megoldására.

A dolgozat célja egy olyan integrált keret létrehozása, amely a mesterséges intelligenciát és az eye-tracking technológiát ötvözi annak érdekében, hogy támogassa az agilis szoftverfejlesztési folyamatokat. Ez a keretrendszer lehetőséget biztosít arra, hogy a fejlesztők valós idejű visszajelzéseket kapjanak arról, hogyan teljesítik feladataikat, és hogyan lehetne optimalizálni munkafolyamataikat. A mesterséges intelligencia segítségével automatizált döntéstámogató rendszerek jöhetnek létre, amelyek képesek előre jelezni a projekt kimeneteleit, értékelni a projekt különböző aspektusait, és javaslatokat tenni a fejlesztési folyamatok finomhangolására. Az eye-tracking adatok alapján a modell részletes elemzést nyújt arról, hogy a felhasználók és a fejlesztők milyen módon lépnek kapcsolatba a szoftverekkel, és hogyan lehetne optimalizálni a felhasználói élményt.

1.1. Kutatás feltételei

A kutatás technikai követelménye a megfelelő hardveres és szoftveres infrastruktúra biztosítása. A mesterséges intelligencia és az eye-tracking technológiák integrációja speciális eszközöket és szoftvereket igényel:

- a) **Szemmozgáskövető eszköz és elemző szoftver integrációja:** A szemmozgások pontos rögzítése és elemzése elengedhetetlen a kutatáshoz. Az eye-tracking eszközöknek alkalmasnak kell lenniük a felhasználói interakciók és a fejlesztői viselkedés rögzítésére, a szemmozgás mintázatok (fixáció, szakkádok) alapján. Emellett szükséges egy megfelelő szoftveres környezet, amely az adatokat valós időben képes elemezni és összekapcsolni a fejlesztési folyamattal.
- b) **Mesterséges intelligencia alapú rendszerek:** Az MI algoritmusok és a gépi tanulási technikák elengedhetetlenek az összegyűjtött adatok feldolgozásához. Olyan algoritmusokra van szükség, amelyek képesek elemezni a szemmozgás-követés által gyűjtött adatokat, és támogatják a fejlesztői döntéshozatalt, például a hibakeresést, kódoptimalizálást és a felhasználói élmény javítását.

1.2. Kutatási célkitűzései

A kutatás elsődleges célja egy specifikus keretrendszer kifejlesztése, amely az agilis szoftverfejlesztési folyamatokat támogatja az MI és eye-tracking technológiák integrálásával. Ez a keretrendszer egy olyan komplex rendszer létrehozását célozza meg, amely képes elemzéseket végezni és azonosítani a fejlesztési folyamatokban rejlő lehetőségeket és kihívásokat, hozzájárulva ezzel a projektmenedzsment hatékonyságának növeléséhez és kódminőség javításához.

A konkrét célok a következők:

- A keret a szemmozgáskövető adatokat használja fel a fejlesztői viselkedés elemzésére, beleértve a kódolási időtöltést és a figyelem fókuszpontjait. Ezen információk alapján a modell javaslatokat tud tenni a kódolási felület és a felhasználói interfész optimalizálására.
- A mesterséges intelligencia alapú keret elkészítése, amely képesek felismerni a gyakori programozási hibákat és javaslatokat tenni a kód javítására. Ez nemcsak a fejlesztői időt csökkenti, hanem növeli a kódbázis stabilitását, átláthatóságát és karbantarthatóságát.
- A modell integrálja az agilis projektmenedzsment adatokat, mint például a sprint tervek és felhasználói visszajelzések, az MI segítségével elemzi ezeket. Az elemzés eredményeképpen a modell képes javaslatokat tenni az agilis folyamatok finomhangolására, javítva a csapat hatékonyságát és a projekt teljesítési sebességét.

1.3. Kutatás módszertana

A fejlesztés és kutatás során a témával kapcsolatos nemzetközi és hazai szakirodalom került áttanulmányozására, annak érdekében, hogy kitűzött célokat, továbbá megvalósítás során alkalmazható technikákat minél szélesebb ismeret alapján tudjam teljesíteni. A kutatás során különböző adatgyűjtési és elemzési technikákat alkalmaztunk, hogy átfogó képet kapjunk a fejlesztési folyamatok optimalizálásáról és a felhasználói viselkedésről. Az adatgyűjtési folyamat középpontjában a szemmozgás-követési technológia állt, amely lehetővé tette a fejlesztők és felhasználók vizuális interakcióinak rögzítését és elemzését. A kutatás során speciális eszközök biztosították a szemmozgások pontos nyomon követését, különös tekintettel a fixációkra és szakkádokra. Az így gyűjtött adatok részletes elemzésével betekintést nyertem a kódolási folyamatok és a felhasználói interakciók hatékonyságába.

A kutatás során alkalmazott módszerek a következő lépésekből álltak:

- **Adatgyűjtés:** A szemmozgás-követési eszközök segítségével valós idejű adatokat rögzítettem a fejlesztők és felhasználók viselkedéséről.
- **Adatfeldolgozás:** A nyers adatok mesterséges intelligencia alapú rendszerekkel történik meg az elemzés, amelyek képesek azonosítani a hibákat, valamint javaslatokat tenni a fejlesztési folyamatok javítására.

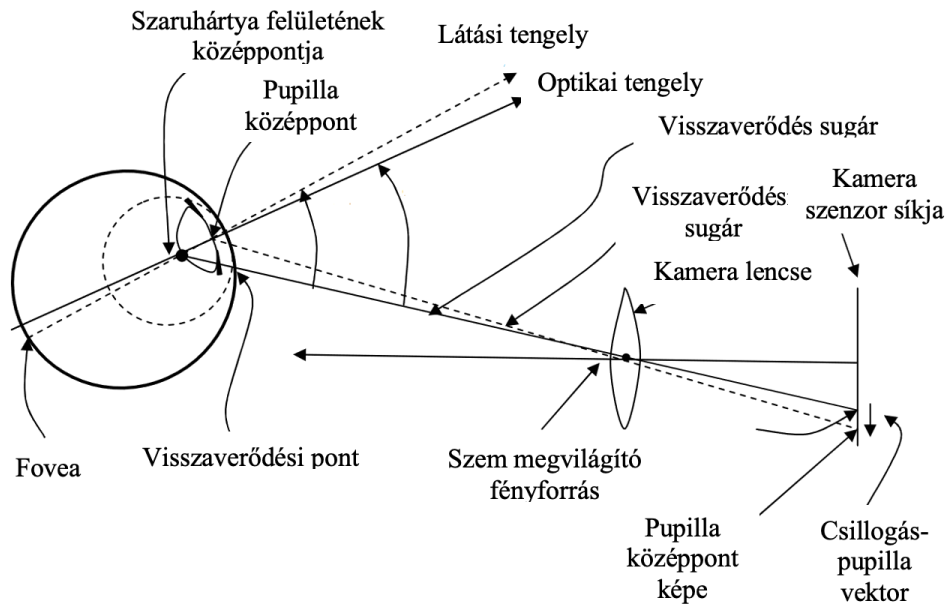
- **Visszacsatolás:** A fejlesztők valós idejű visszajelzéseket kaptak az MI és eye-tracking rendszerek elemzése alapján, amely segített az optimalizációs lehetőségek azonosításában.

2. Szemmozgáskövetést megvalósító technikák alkalmazása

A szemmozgáskövetés technikái az emberi szem által végzett mozgások rögzítésére és elemzésére szolgálnak, amelyek a felhasználó kognitív és vizuális viselkedéséről nyújtanak fontos információkat [13]. Az utóbbi évek technológiai fejlődésének köszönhetően egyre inkább elérhetővé váltak olyan eszközök, amelyek lehetővé teszik a szemmozgások pontos követését különböző környezetekben, például szoftverfejlesztési folyamatokban is [14] [15]. Ezen eszközök és technológiák integrálása a fejlesztési munkafolyamatokba hozzájárulhat a kódolás és a felhasználói élmény mélyreható elemzéséhez, valamint optimalizálhatja a munkafolyamatok hatékonyságát. A kutatás során alkalmazott szemmozgáskövetési technikák célja, hogy részletesen feltárják a fejlesztők kognitív terhelését és figyelmi mintáit a szoftverfejlesztés során. A rögzített adatok alapján a kód felhasználói felületének optimalizálására vonatkozó javaslatok tehetők, valamint lehetőség nyílik a kódminőség javítására és a hibák azonosítására is. A szemmozgási adatok gyűjtésére különböző hardvereszközök és szoftveres megoldások állnak rendelkezésre, amelyek biztosítják az adatok pontos rögzítését és elemzését.

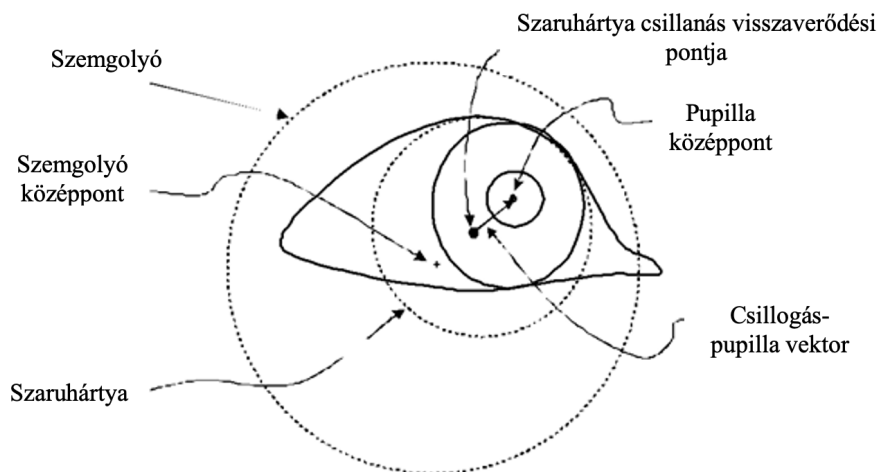
2.1. Pupilla központú szaruhártya tükröződés (PCCR) elve

A PCCR módszer alapjául szolgáló elmélet azt fejezi ki, hogy egy fényforrás segítségével a szemről erősen visszaverődő, továbbá a szaruhártyán és a pupillán létrehozott reflexió alkalmazása rögzíthető egy infravörös kamera segítségével. Ezáltal a kamera és a reflexió között egy olyan vektor jön létre, ami alapján kiszámítható a tekintet jelenlegi iránya [16] [17].



1. ábra: A PPCR módszer működési elve [17]

Amikor egy személy közvetlenül a kamerába néz, a szaruhártya csillanás a pupilla középpontja kerül visszaverődésre. Ahogyan mozog a pupilla, úgy a kamera is más szögben látja a szemet. Amennyiben valaki felfelé forgatja szemeit a kamerához képest, akkor a pupilla közepe egyaránt felfele mozdul a szaruhártya csillanás közepétől, így a vektor is magasabbra mutat. Az eset ugyanúgy játszódik le bármilyen irányban, például, ha balra néz (kamera szögében jobbra), a vektor is egyaránt a baloldali irányba mozdul, a pupilla középpontjához. A vektort gyakran nevezik csillogás-pupilla vektornak is, amelyet a 3. ábra ábrázol.



2. ábra: A csillogás-pupilla vektor meghatározása

Forrás:

https://www.researchgate.net/publication/32889349_Eye_Tracking_A_Perceptual_Interface_for_Content_Based_Image_Retrieval/figures

2.2. Pupilla megvilágítási technikák részletezése

Az infravörös alapú szemmozgáskövető kamerák a pupilla és a pillantás meghatározásához valamilyen megvilágítási módszert vesznek igénybe. Ezt kétféle úton megtehető, amelyek a következők:

2.2.1. Világospupilla módszer

Ebben az esetben az infravörös fény megvilágítja a szemet, ami a retináról visszaverődik egyenesen a kamerába. Ennek következtében a pupilla fényesen jelenik meg az adatfeldolgozó képen. Ugyanaz a jelenség játszódik le a „vörösszem effektusnál” is, amikor valakit lefényképezünk vaku használatával. A szemmozgás követő algoritmusok úgy azonosítják a pupillát, hogy fényes, ellipszis alakú formát keresnek a képen. Amennyiben a fényforrás kamera optikai tengelyével egybeeső, akkor a világospupilla-módszer alkalmazása javasolt pupilla megvilágítása céljából, továbbá a környezeti feltételeknek egyaránt meg kell felelni. Mivel az algoritmus fényesebbnek érzékeli a pupillát, mint a szivárványhártyát, így nagyon fontos, hogy a vizsgálat sötét térben történjen meg, illetve világos szemű (például kék) résztvevő legyen [17] [18] [19], [20].

2.2.2. Sötétpupilla-módszer

A sötétpupilla-módszerhez hasonlóan kamera szintén megvilágítja pupillát infravörös fénnel, azonban, ha a személy szeme és kamera egybeesési szöge eltérő, így a pupilla sötétebbnek tűnik, mint a képen látható szem többi része. Ebben az esetben a szemmozgás követő algoritmusok sötét elliptikus formát keresnek a szemképen. Ezt a módszert normál megvilágítású helyszíneken történő vizsgálatoknál célszerű alkalmazni, továbbá a fent említett problémát jobban tolerálja. Ha az alany barna szemű, esetleg sötét színű szempillafestéket használ, akkor ezek a hatások kevésbé befolyásolják a mérést. A szemmozgás követő termékekben általában mindkét módszer elérhető, legtöbbjükben beépített szoftver algoritmus határozza meg adott környezetben melyik működik precízebben, a nagyon speciális kamerákban pedig felhasználó ki tudja választani az adott helyszínen. A kutatásban alkalmazott eszköz is mindkét módszert képes használni [17] [18] [19], [20] .

2.3. PCCR elvet alkalmazó szemmozgáskövető eszközök csoportosítása

A videóalapú szemmozgáskövető eszközök két nagyobb csoportba oszthatók: képernyő alapú (statikus) és fejhez csatoltak (dinamikus). A képernyő alapú kamerákról akkor beszélünk, amikor a tekintetkövető egy megjelenítő eszköz (például számítógépes monitor, televízió), alatt helyezkedik el, így a kamera nagy távban nem mozdítható. Ekkor a szemmozgást vizsgáló

alkalmazás végrehajtja a feladatokat, és a kijelzőn lévő alkalmazások megjelenítését, miközben a személy a megjelenítőn látottakat vizsgálja. A dinamikus, a statikussal szemben pedig, a kamera nem fix helyzetben, hanem szemüveggént üzemel, és ezt viselnie kell a tesztalanynak a vizsgálat során. Ennek következtében, a résztvevő nincs közvetlen kapcsolatban a vizsgálandó termékkel, ezáltal a vizsgálati terület nem korlátozott, amely garantálja a szabadon mozgást. Mivel szoftverfejlesztés egy számítógépes kijelzőn történik, így kutatásomhoz statikus jellegű berendezést vettem igénybe.

2.3.1. Képernyő alapú szemmozgáskövetők áttekintése

Mivel kutatásom számítógép képernyőjéhez kötött, ezért statikus szemmozgáskövető eszközök listáját tártam fel. Számos eszköz létezik a piacon különböző árkategóriában, ebből emeltem ki néhányat. Az eszközök technikai információi a hivatalos weboldalukról gyűjtöttem ki.

- 1 **Gazepoint 3:** A GP3 (2. sz. melléklet) szemmozgáskövető kamera magas teljesítménnyel rendelkezik, amely Gazepoint Control és Analysis szoftvercsomaggal érkezik. A két szoftver lehetőséget biztosít, hogy szoftveren belül Gazepoint kamerát kalibráljon, illetve adatgyűjtést, elemzést tesz lehetővé. A kamera 11 x 25 cm nagyságú, legfeljebb 24” monitorig képes figyelni a tekintetet.
- 2 **Smart Eye Aurora:** Az Aurora (3. sz. melléklet) egy kis méretű, hordozható eszköz, amely lehetőséget ad a különböző méretű kijelzők egyszerű váltására, bármiféle teljesítmény- vagy pontosságromlás előfordulás nélkül.
- 3 **Tobii Eye X:** A Tobii (Tobii Technology AB) egy svéd csúcstechnológiai vállalat, amely főként szemmozgás követés paraméterek gyűjtésére, elemzésére alkalmas termékeket fejleszt és értékesít. A Tobii termékeit széleskörben használják különböző területen, mint tudományos kutatásban, marketingben, manapság játékokban is. Az eszköz nem igényel folyamatos újra kalibrálást, továbbá mérsékelt fejmozgásokat tesz lehetővé. A szemmozgás paraméterek számításához szükséges képfeldolgozást a Tobii Eye X Engine végzi, amely a USB 3.0 porton keresztül van csatlakoztatva a számítógéphez. Egyszerre több alkalmazás is letudja kérdezni a kamera adatait, akár az Eye X fejlesztői felületen, vagy LAN kapcsolaton keresztül. Az Eye X-szel együtt kiadott Tobii SDK fejlesztői könyvtárakat biztosít C, C++, .NET C#, illetve Unity 3D programozási nyelveken. A Tobii Eye X (4. sz. melléklet) vezérlő ugyan egy alacsony költségű, binokuláris szemkövetésre, amelyet játék- és fogyasztói alkalmazásokba való integrációját teszi lehetővé.

- 4 **EyeLink 1000 Plus:** Az EyeLink 1000 Plus (5. sz. melléklet) egy nagy teljesítményű mérőeszköz, amely lehetővé teszi széleskörű szemmozgás követő feladatok elemzését. Alacsony térzajjal, és magas mintavételi frekvenciával képes adatot rögzíteni, továbbá az elsők között szerepel, amely szoftveresen és műszakilag egyaránt lehetővé teszi működését MRI, MEG, EEG-n át viselkedési laboratóriumig. Kamera tartalmaz szemkamera mellett kartartót, fejtartót, továbbá asztali rögzítésre is képes, amely elősegíti a szemkövetési feladatok elsajátításának optimalizálását.

2.4. **The Eye Tribe:** Az Eye Tribe egy dán, 2011 októberében induló vállalat volt, amely hírnevét az alacsony költségű szemkövető eszköz gyártásáról nyerhette el. Az gyártó világrekordot döntött ezzel, mint legkisebb, egyben legolcsóbb szemmozgás követő eszköz. Képes szemmozgás adatokat rögzíteni egy 20 x 1.9 x 1.9 cm méretű eszközzel (6. sz. melléklet). Az eszköz követi a felhasználó pupillájának legapróbb mozgását is, illetve annak méretét. A kamera pontossága, körülbelül 0.5° és 60 adatot biztosít másodpercenként. Az alkalmazás fejlesztői könyvtára C++, C# és Java programozási nyelveken érhető el Windows vagy Mac környezet alatt. Szemmozgáskövetés paraméterek rögzítésére szolgáló szoftverek. Mivel az eszköz korábbi kutatásomban rendelkezésre állt, így az Eye Tribeot fogom használni a jelenlegi kutatásra.

A szemmozgáskövetési technológiák egyik legfontosabb része a szoftveres háttér, amely lehetővé teszi a szemmozgások rögzítését, feldolgozását és értelmezését. Ezek a szoftverek nagy szerepet játszanak abban, hogy a felhasználói interakciók során gyűjtött nyers adatokat hasznos információvá alakítsák, amelyeket a kutatók vagy fejlesztők felhasználhatnak a szoftverek és felhasználói felületek optimalizálására. Az alábbiakban részletesen bemutatom a szemmozgáskövetési adatok rögzítésére szolgáló szoftverek működését és felépítését. A szemmozgáskövetési szoftverek célja, hogy a felhasználók szemmozgásait valós időben rögzítsék, majd az adatokat a felhasználás céljának megfelelően elemezzék. Ezek a szoftverek többnyire fejlett adatfeldolgozó algoritmusokat és vizualizációs eszközöket kínálnak, amelyek segítségével az adatok könnyen értelmezhetők és használhatók.

1. **Adatfeldolgozó modul:** Az adatfeldolgozó modul a szemmozgáskövetési rendszer egyik legkritikusabb eleme, amely a nyers adatok tisztítását, feldolgozását és elemzését végzi. A nyers adatok gyakran zajosak és tartalmazhatnak hibákat, ezért az adatfeldolgozó algoritmusok első feladata az adatok tisztítása, azaz a hibás, hiányos vagy anomáliákat mutató adatok kiszűrése. Ezek az anomáliák általában hibás szemmozgás-követési mérésből vagy környezeti tényezők változásaiból adódhatnak. A

tisztítási folyamat után az adatfeldolgozó algoritmusok különféle statisztikai mutatókat generálnak, amelyek segítségével a felhasználói szemmozgásokat mélyebb szinten elemezhetjük. A leggyakrabban használt statisztikai mutatók közé tartoznak a következők:

- a. **Fixáció időtartama:** Mennyi ideig koncentrált a felhasználó egy adott pontra.
- b. **Szakkád:** A két fixációs pont közötti távolság, azaz a szem mozgásának útvonala egyik pontból a másikba.
- c. **Fókuszpont:** Adott pillanatban hova nézett a felhasználó.

Ezek az elemzések lehetővé teszik, hogy a kutatók megértsék, hogyan viselkednek a felhasználók egy adott felületen, és hol tapasztalhatnak nehézségeket vagy elakadást. A feldolgozás során az adatok normalizálása is megtörténik, hogy a különböző felhasználók, környezetek és kísérleti körülmények között gyűjtött adatok egységes formában kerüljenek tárolásra és elemzésre. Ez biztosítja, hogy az adatok összehasonlíthatók legyenek, és az MI alapú elemzésekhez is megfelelő formátumban álljanak rendelkezésre.

- 2. **Adattárolási modul:** A szemmozgáskövetési rendszerek jelentős mennyiségű adatot generálnak, különösen, ha hosszabb időn keresztül és több felhasználó bevonásával történik az adatgyűjtés. Az adattárolási modul célja, hogy ezek az adatok strukturált formában, adatbázisokban kerüljenek tárolásra, ami lehet helyi szervereken vagy távoli, felhőalapú rendszerekben. Az adatbázisok használata kulcsfontosságú, mivel ezek biztosítják a hatékony tárolást, gyors hozzáférést és a nagyméretű adathalmazok kezelését. Az adattárolási modul biztosítja, hogy az adatok időbélyegekkel és felhasználói azonosítókkal együtt, strukturált módon kerüljenek az adatbázisba, lehetővé téve a későbbi szűrést, keresést és elemzést. Ez különösen fontos, mivel a szemmozgáskövetési adatok nagy volumenűek, és a pontos elemzéshez szükség van az adatok megfelelő rendszerezésére és könnyű visszakereshetőségére. A mesterséges intelligencia algoritmusok nagymennyiségű adatot igényelnek ahhoz, hogy hatékonyan működjenek, tanuljanak a korábbi interakciókból, és pontos előrejelzéseket tudjanak generálni a jövőbeli felhasználói viselkedésről. Minél több adat áll rendelkezésre, annál megbízhatóbbak és pontosabbak lehetnek a gépi tanulási modellek. Az adatbázisok biztosítják, hogy a gépi tanulás során használt adatok rendszerezve, könnyen elérhetően és folyamatosan frissítve legyenek, lehetővé téve a modellek folyamatos fejlesztését és optimalizálását. Ez különösen fontos a dinamikusan változó környezetekben, ahol az

MI algoritmusok folyamatosan alkalmazkodnak a felhasználói viselkedéshez és újabb adatokhoz.

3. **Vizualizációs modul:** A vizualizációs modul rögzített adatokat olyan formában jeleníti meg, amely könnyen értelmezhető a fejlesztők és a kutatók számára. Ez a modul különféle grafikai megjelenítéseket kínál, amelyek vizuálisan ábrázolják a felhasználók szemmozgásait és figyelmi mintázatait. Az MI-alapú elemzésekben a vizualizációs modul különösen fontos, mivel ezek a vizuális reprezentációk segítik a modellek tanítását. A hő térképek és más vizualizációk alapján az MI algoritmusok könnyebben azonosítják a releváns mintákat, amelyek a felhasználói viselkedést leginkább befolyásolják.

3. A mesterséges intelligencia alapjai

A mesterséges intelligencia (MI) egyre nagyobb szerepet játszik a modern technológiai fejlesztésekben, legyen szó akár automatizált rendszerek működtetéséről vagy komplex adatfeldolgozási feladatok megoldásáról. Az MI alapjai olyan rendszerek fejlesztésére összpontosítanak, amelyek képesek emberi intelligenciát utánozni különböző feladatokban, beleértve a tanulást, a minták felismerését, döntéshozatalt, valamint önállóan történő problémamegoldást. A mesterséges intelligencia mögötti filozófiai és technikai megközelítések több évtizedes fejlődés eredményeképpen mára képesek átalakítani az ipar számos területét, beleértve az egészségügyet, a pénzügyeket, a gyártást, és még sok más területet. Ebben a fejezetben részletesen bemutatom a mesterséges intelligencia fogalmát, történetét, fő területeit, alapvető algoritmusait, valamint alkalmazási területeit. [21] [22]

Az első mesterséges intelligencia programokat az 1950-es és 1960-as években fejlesztették ki, amelyek főként szabályalapú rendszereken alapultak. Ezek a korai MI rendszerek merevek voltak, mivel előre programozott szabályokat követtek, és nem voltak képesek adaptálódni új helyzetekhez. Az 1980-as években megjelentek az első gépi tanulási algoritmusok, amelyek már képesek voltak adatokból tanulni, és azokat alkalmazni a döntéshozatal során. Ez a fejlődés jelentős előrelépést jelentett, mivel ezek az algoritmusok már nemcsak statikus szabályokat követtek, hanem dinamikusan tudtak alkalmazkodni a változó körülményekhez. Az MI fejlesztése új lendületet kapott a 21. század elején, amikor a számítási kapacitások növekedésével lehetővé vált a nagyméretű adathalmazok elemzése. Ez különösen a mélytanulási technikák kifejlődését segítette elő, amelyek több rétegű neurális hálózatokat alkalmaznak a komplex mintafelismerési feladatok elvégzésére. Az MI ma már a mindennapok részévé vált, legyen szó akár személyi asszisztensekről, okos eszközökről, önvezető autókról vagy prediktív elemzési rendszerekről. A technológia folyamatosan fejlődik, és egyre több területen nyer alkalmazást, mivel az MI képes olyan összetett feladatok megoldására, amelyek korábban kizárólag emberi intelligenciát igényeltek. [23] [24] [25]

3.1. A mesterséges intelligencia alkalmazási területei

A mesterséges intelligencia nem csupán egy technológiai eszköz, hanem egy olyan módszertan, amely átalakítja az iparágakat és a mindennapi életünket. A mesterséges intelligencia fejlődése lehetővé teszi, hogy a különböző iparágak előnyökhöz jussanak a gépi tanulás, a mélytanulás

és más MI-technológiák által nyújtott lehetőségek kihasználásával. Az alábbiakban bemutatásra kerül néhány alapvető alkalmazási területet:

1. **Egészségügy:** Az MI egyik legfontosabb és leggyorsabban fejlődő alkalmazási területe az egészségügy, ahol a mesterséges intelligencia a diagnosztikai folyamatok felgyorsításában, a betegségek korai felismerésében és a kezelési tervek optimalizálásában játszik központi szerepet. Az MI-alapú rendszerek képesek orvosi képeket, például röntgenfelvételeket vagy MRI-képeket elemezni, és a képek alapján diagnózist készíteni, sok esetben gyorsabban és pontosabban, mint az emberi orvosok. Ezenkívül az MI segíthet az orvosi kutatásokban is, például az új gyógyszerek kifejlesztésében, ahol a gépi tanulás algoritmusok képesek gyorsabban felfedezni az ígéretes gyógyszerjelölteket a hatalmas adathalmazokból. [26] [27] [28]
2. **Pénzügyi szektor:** Az MI jelentős szerepet játszik a pénzügyi szolgáltatásokban is, ahol az adatvezérelt döntéshozatal és a kockázatkezelés kulcsfontosságú. A mesterséges intelligencia rendszerek képesek elemezni a piaci trendeket, előre jelezni az árfolyamok mozgását, és olyan döntéseket hozni, amelyek növelik a befektetések megtérülését. Az MI rendszerek segítségével a pénzügyi intézmények gyorsabban és pontosabban tudják értékelni a hitelkockázatokat, felismerni a csalásokat, valamint személyre szabott pénzügyi termékeket kínálni az ügyfeleknek. [29] [30]
3. **Önvezető járművek:** Az MI kulcsszerepet játszik az önvezető járművek fejlesztésében, ahol a rendszereknek valós időben kell elemezniük a környezetet, felismerniük a közlekedési jeleket, a gyalogosokat és a többi járművet. Az MI-algoritmusok folyamatosan tanulnak a környezetből, és képesek arra, hogy biztonságosan navigáljanak különböző útviszonyok között. Az önvezető járművek MI-je nemcsak a jármű vezetéséért felel, hanem az ütközések elkerüléséért és a jármű működési hatékonyságának optimalizálásáért is. [31] [32]
4. **Gyártás és logisztika:** Az MI-t széles körben használják a gyártási folyamatok automatizálására és optimalizálására. A gépek és robotok MI-algoritmusok segítségével képesek javítani a gyártási hatékonyságot, csökkenteni a hibákat, és biztosítani a gyártási folyamatok zökkenőmentes működését. A logisztikában az MI segít optimalizálni a szállítási útvonalakat, csökkenteni a szállítási költségeket, és növelni a szállítási pontosságot, ami jelentős megtakarításokat és javuló ügyfélszolgálatot eredményez. [33] [34]

4. Mesterséges intelligencia és szemmozgás-követés integrációja

A mesterséges intelligencia és a szemmozgás-követési (eye-tracking) technológiák integrációja egy olyan fejlett megközelítés, amely a szoftverfejlesztés hatékonyságának és a felhasználói élmény optimalizálásának új dimenzióit nyitja meg. A két technológia kombinációja lehetőséget ad arra, hogy valós idejű adatokat gyűjtsünk a fejlesztési folyamatokról és a felhasználói interakciókról, amelyek mélyebb elemzése révén jelentős fejlődést érhetünk el az agilis módszertanok hatékonyabb alkalmazásában.

Az MI alapú rendszerek nagy mennyiségű adatot képesek rövid idő alatt feldolgozni, így automatizálhatják a hibakeresést, és gyorsíthatják a döntéshozatali folyamatokat. Ezek az algoritmusok nemcsak a kódolás során, hanem a projektmenedzsment területén is hasznosak, hiszen előrejelzéseket készíthetnek a projekt várható időkereteiről, a fejlesztési ciklusok hatékonyságáról, valamint az erőforrások optimális felhasználásáról.

A szemmozgás-követési technológia más szempontból támogatja a fejlesztési folyamatokat. Az eye-tracking segítségével rögzíteni tudjuk, hogy a fejlesztők és a felhasználók milyen vizuális interakciókat végeznek a szoftverekkel, és milyen figyelmi mintákat követnek. Ez a technológia segít feltárni, hogy a felhasználók hol akadnak el a használat során, mely felületek váltják ki a legnagyobb figyelmet, és hol szükséges a felhasználói élmény javítása. A szemmozgáskövetés tehát egyfajta vizuális visszacsatolást biztosít, amely alapján a fejlesztők optimalizálhatják a felhasználói felületeket, így a szoftverek intuitívabbá és könnyebben használhatóvá válhatnak.

Az MI és az eye-tracking integrációjának legfontosabb előnyei a következők:

1. **A döntéshozatal automatizálása:** Az MI egyik legnagyobb előnye, hogy képes automatizálni a döntéshozatali folyamatokat. Az MI modellek az adatokat elemezve javaslatokat tesznek a legjobb döntések meghozatalára, például a kód optimalizálásával, a felhasználói felületek javításával vagy a fejlesztési erőforrások hatékony elosztásával kapcsolatban. Ez a fajta automatizáció jelentősen csökkenti a fejlesztői csapatok terhelését, hiszen a döntéseket az MI rendszerek gyorsabban és pontosabban tudják meghozni az összegyűjtött adatok alapján.
2. **Kódoptimalizálás és hibajavítás:** Az MI algoritmusok lehetőséget biztosítanak a kód automatikus elemzésére és javítására. A gépi tanulási modellek képesek azonosítani a gyakran előforduló hibákat a kódban, és javaslatokat tenni azok kijavítására [35] [36]. Ezen kívül a szemmozgás-követési technológia segítségével a fejlesztők pontosan

láthatják, mely kódrészek okoznak számukra nehézséget, illetve hol szükséges további finomítás. Ez a megközelítés nemcsak a fejlesztési időt csökkenti, hanem a kód minőségét is jelentősen növeli, hiszen a hibák gyors és hatékony kijavításával megelőzhetjük a további problémákat.

3. **Felhasználói élmény javítása:** Az eye-tracking technológia különösen hasznos a felhasználói élmény elemzésében. Azáltal, hogy követi a felhasználók szemmozgását, pontos képet ad arról, hogy a felhasználók hol töltenek sok időt, hol akadnak el, és mely funkciókat használják a leggyakrabban. Ezek az adatok felhasználhatók a felhasználói felületek átalakítására és optimalizálására, hogy azok jobban igazodjanak a felhasználók igényeihez és elvárásaihoz. Az MI algoritmusok ezeket az adatokat elemezve javaslatokat tehetnek a felületek javítására, például olyan funkciók előtérbe helyezésére, amelyeket a felhasználók gyakran használnak, vagy az elakadási pontok megszüntetésére.

4.1. A modell keretének kialakítása és célja

A modell kerete elsődleges feladata, hogy a fejlesztők számára automatikus döntéshozatali eszközt biztosítson, amely képes valós időben elemezni a fejlesztési folyamatok során felmerülő problémákat, optimalizációs lehetőségeket és felhasználói viselkedési mintákat. Az agilis szoftverfejlesztési módszertanok egyik legnagyobb kihívása a folyamatosan változó követelményekhez való alkalmazkodás és a gyors iterációk kezelése [37] [38]. A fejlesztőknek és a projektmenedzsereknek gyakran az a feladata, hogy rövid idő alatt döntsék el, mely funkciók javítására vagy fejlesztésére van szükség, miközben figyelembe kell venniük a felhasználói visszajelzéseket, az erőforrások korlátait és a projekt határidejét. A keret célja, hogy megkönnyítse ezt a folyamatot azáltal, hogy automatizálja az adatgyűjtést, a hibajavítást és a felhasználói élmény javítását célzó optimalizációs folyamatokat. Az MI és a szemmozgáskövetés integrációja révén a keret valós idejű adatokat biztosít a fejlesztők és a felhasználók viselkedéséről, így képes előre jelezni a potenciális hibákat és a fejlesztési folyamatok nehézségeit. A modell kerete segít azonosítani azokat a területeket, ahol a felhasználók elakadnak, vagy ahol a kód optimalizálásra szorul. Ezenkívül a mesterséges intelligencia algoritmusai elemzik a szemmozgás adatait, és javaslatokat tesznek a felhasználói felület átalakítására és fejlesztésére. Ezáltal a fejlesztők gyorsabban és hatékonyabban tudnak reagálni a felhasználói igényekre, és képesek finomhangolni a szoftvert anélkül, hogy időigényes manuális tesztelésre lenne szükség.

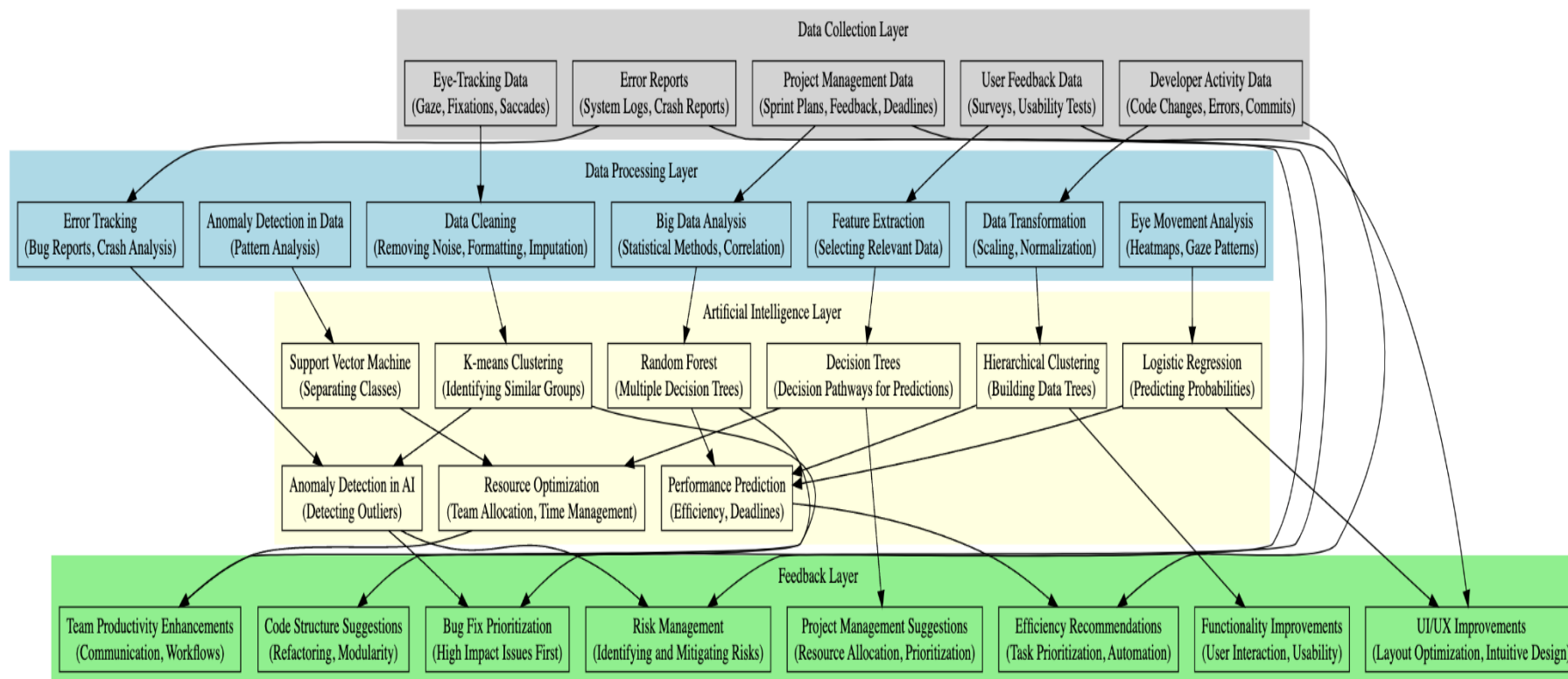
4.2. A modell keretének komponensei

A modell komponensei hierarchikus felépítésben működnek együtt, ahol az egyes rétegek különböző funkciókat látnak el: az adatok gyűjtése, azok előkészítése és feldolgozása, a mesterséges intelligencia alapú elemzések, valamint a visszajelzések biztosítása. Ez a felépítés lehetővé teszi, hogy a rendszer komplex adatfolyamokat kezeljen és ezek alapján olyan predikciókat és optimalizálási javaslatokat nyújtson, amelyek közvetlen hatással vannak a projekt sikerességére.

Az adatok forrásai sokrétűek, és magukban foglalják például a fejlesztői tevékenység nyomon követését, az eye-tracking technológia által rögzített felhasználói viselkedési adatokat, a projektmenedzsment adatait, valamint a felhasználói visszajelzéseket és hibajelentéseket. Ezek az adatok központi szerepet játszanak a modell működésében, mivel az elemzésre kerülő adatok minősége és relevanciája alapvetően befolyásolja a mesterséges intelligencia algoritmusok által nyújtott eredményeket.

A mesterséges modell több rétegből épül fel, amelyek mindegyike külön feladatokat lát el, és egymással folyamatos kommunikációban áll. Az első lépés az adatok gyűjtése, amelyet a feldolgozási réteg követ, ahol az adatok tisztításra és átalakításra kerülnek, hogy megfelelő formában legyenek az elemzéshez. Az elemzési réteg a mesterséges intelligencia különböző algoritmusait használja, amelyek predikciókat és klaszterezéseket végeznek, valamint azonosítják az adatban meglévő anomáliákat és mintázatokat. Végül, a visszajelzési réteg biztosítja, hogy a kapott eredmények konkrét javaslatokként hasznosíthatók legyenek, és közvetlen visszacsatolást nyújtanak a fejlesztőknek és projektmenedzsereknek.

Az 3. ábra bemutatja a mesterséges modell különböző komponenseit és azok egymáshoz való viszonyát. A diagram szemlélteti, hogyan kapcsolódnak az egyes rétegek egymáshoz, és miként zajlik az adatok folyamatos áramlása a rendszerben.



3. ábra: Mesterséges intelligencia és az eye-tracking integrációjának keretrendszere.
 Forrás: Saját készítésű ábra.

A komponensek közötti kapcsolatok pontosan tükrözik a rendszer működésének logikai sorrendjét, amely az adatok átalakulásán keresztül a döntési mechanizmusok felé halad. Az egyes komponensek szinergiája biztosítja a rendszer rugalmasságát és hatékonyságát. Az adatgyűjtés például közvetlenül kapcsolódik az adattisztítási és adattranszformációs folyamatokhoz, amelyek nélkül a mesterséges intelligencia algoritmusok nem tudnának pontos elemzéseket nyújtani. A visszajelzési mechanizmusok pedig lehetővé teszik, hogy a rendszer folyamatosan fejlődjön, az eredményekre építve javuljon a fejlesztési folyamat. A következő szakaszokban részletesen áttekintjük a mesterséges modell egyes komponenseit, azok funkcióit és a rendszeren belüli kapcsolataikat. Ezek a komponensek együtt biztosítják a keretrendszer átfogó és hatékony működését, hozzájárulva a döntéshozatali folyamatok javításához és a szoftverfejlesztési projektek sikeres kivitelezéséhez.

4.2.1. Adatgyűjtési réteg

Az adatgyűjtési réteg kiemelten fontos szerepet tölt be a mesterséges modell működésében, hiszen a rendszer összes további komponense ezen réteg által gyűjtött adatokra épül. A különböző forrásokból érkező adatok biztosítják, hogy a rendszer átfogó képet kapjon a felhasználói viselkedésről, a fejlesztők tevékenységeiről, a hibák kezeléséről, valamint a projektmenedzsment folyamatairól. Ez a réteg adja a keret gerincét, amely nélkül a feldolgozás, elemzés és optimalizálási mechanizmusok nem tudnának működni. Az adatok gyűjtése különböző típusú eszközök és módszerek segítségével történik, beleértve az eye-tracking technológiát, a fejlesztők tevékenységeinek nyomon követését, a hibajelentéseket, valamint a felhasználói visszajelzéseket. Az így megszerzett adatok feldolgozása és elemzése során a rendszer képes értékes következtetéseket levonni, amelyeket később a fejlesztési folyamatok optimalizálására használ. Az adatgyűjtési réteg jelentősége abban rejlik, hogy lehetővé teszi az átláthatóságot a fejlesztési ciklus minden szakaszában. A megfelelően begyűjtött és strukturált adatok biztosítják, hogy a rendszer megbízható és pontos predikciókat készíthessen. Az adatgyűjtés folyamata különböző forrásokból történik, amelyek mindegyike más-más aspektusát világítja meg a fejlesztésnek, ezáltal biztosítva a teljes képet. A következő szakaszokban részletesen bemutatom az adatgyűjtési réteg alkomponenseit, és megvizsgálom azok funkcióit és szerepét a rendszerben.

Eye-tracking adatok (Tekintet, fixációk, szakkádok)

Ez a technológia a felhasználók szemmozgásait követi nyomon, és adatokat gyűjt arról, hogy a felhasználók milyen elemekre fókuszálnak a felületen, milyen sorrendben járják be az oldalakat, és milyen interakciókat hajtanak végre. Az eye-tracking paraméterek három fő

kategóriába sorolhatók, melyek a tekintet, fixációk és szakkádok. Ezeket az adatokat az alábbi felsorolás részletezi:

- A tekintetadatok azt mutatják, hogy a felhasználó pontosan hová néz egy adott időpontban. Ezek az adatok segítségével meghatározhatjuk, hogy a felhasználó milyen információkat keres, és hol időzik hosszabb ideig a felületen.
- A fixációk azt mérik, hogy a felhasználó mennyi ideig koncentrál egy adott pontra. Ez az információ különösen hasznos a felhasználói felületek tervezése során, hiszen kiemeli azokat a részeket, ahol a felhasználó elidőzik, és ahol esetleg valamilyen probléma vagy nehezen érthető elem található. A fixációs adatok elemzése lehetőséget nyújt arra, hogy optimalizáljuk a felhasználói élményt, például egyes elemek átrendezésével vagy kiemelésével.
- A szakkádok a tekintet gyors mozgásait rögzítik egyik pontról a másikra. Ezek az adatok általában azt mutatják meg, hogy a felhasználó hol keres információt, és hogyan navigál a különböző elemek között. A szakkádok különösen hasznosak a navigációs problémák feltárásában, mivel segítenek megérteni, hogy a felhasználó milyen gyorsan találja meg a keresett információt, vagy esetleg eltéved-e a felületen.

Az eye-tracking adatok lehetővé teszik a hő térképek készítését is, amelyek vizualizálják a felhasználók figyelmének koncentrációját a felületen. Ezek a hő térképek pontos képet adnak arról, hogy mely elemek vonzzák a legnagyobb figyelmet, és melyek azok a területek, ahol a felhasználók kevésbé időznek. Az ilyen elemzések segítenek a felület optimalizálásában, és növelik a felhasználói elégedettséget, hiszen a felhasználók könnyebben és gyorsabban megtalálják a keresett információkat.

Fejlesztői tevékenység adatok

A hibajelentések kulcsfontosságú szerepet játszanak a rendszer megbízhatóságának és stabilitásának fenntartásában. Ezek az adatok biztosítják a fejlesztők számára a hibák azonosítását és gyors kijavítását, ezáltal növelve a rendszer általános teljesítményét és a felhasználói elégedettséget. A hibajelentések két fő típusba sorolhatók: rendszernaplók és összeomlási jelentések. A rendszernaplók a rendszer működése során keletkező eseményeket, műveleteket és folyamatokat rögzítik. Ezek az adatok lehetővé teszik a fejlesztők számára, hogy nyomon kövessék, mi történik a rendszerben egy adott időpontban, és azonosítsák azokat a műveleteket, amelyek hibát okozhatnak. A rendszernaplók elemzése során a fejlesztők felismerhetik azokat az ismétlődő problémákat, amelyek gyakran vezetnek hibákhoz, és ezek

alapján javaslatokat tehetnek a rendszer optimalizálására. Az összeomlási jelentések szintén fontos szerepet játszanak a rendszer stabilitásának fenntartásában. Ezek a jelentések részletes információkat tartalmaznak arról, hogy miért és hogyan történt egy rendszerösszeomlás. Az összeomlási jelentések elemzése során a fejlesztők azonosíthatják azokat a kódrészeket, amelyek hibát okoznak, és gyorsan kijavíthatják azokat. Az ilyen jelentések általában hibakódokat, naplófájlokat, valamint a rendszer futási környezetét is tartalmazzák, amely alapján a fejlesztők pontosan megérthetik, mi okozta a problémát.

A hibajelentések elemzése lehetőséget biztosít a rendszer folyamatos finomítására, hiszen a fejlesztők azonosíthatják azokat a területeket, amelyek hajlamosak a hibákra, és időben beavatkozhatnak. Ezáltal csökkenthető a felhasználói elégedetlenség, és növelhető a rendszer megbízhatósága.

Projektmenedzsment adatok (Sprint tervek, visszajelzések, határidők)

A projektmenedzsment adatok segítik a csapatokat abban, hogy hatékonyan nyomon kövessék a projekt előrehaladását, és biztosítsák a határidők betartását. Ezek az adatok magukban foglalják a sprint terveket, a csapat visszajelzéseit és a projekt határidejét, amelyek segítségével a fejlesztési folyamat folyamatosan optimalizálható [39]. A sprint tervek biztosítják a csapat számára a fejlesztési ciklus során elérendő célokat, és segítenek a feladatok rangsorolásában. A sprint során összegyűjtött adatok alapján a rendszer képes előre jelezni, hogy a csapat teljesíti-e az előírt feladatokat az adott időkereten belül. A visszajelzések kulcsfontosságúak a folyamat folyamatos fejlesztése érdekében, hiszen ezek az adatok lehetőséget biztosítanak a csapat számára, hogy azonosítsák azokat a területeket, ahol további fejlesztések szükségesek. A határidők nyomon követése biztosítja, hogy a projekt a megfelelő ütemben haladjon, és a csapat figyelemmel kísérje azokat a feladatokat, amelyek csúszásban vannak.

A projektmenedzsment adatok lehetővé teszik, hogy a fejlesztők és projektmenedzserek időben felismerjék a lehetséges akadályokat és késlekedéseket, valamint javaslatokat kapjanak a feladatok újraszervezésére, hogy a projekt a lehető legoptimálisabban haladjon.

Felhasználói visszajelzések (Kérdőívek, használhatósági tesztek)

A felhasználói visszajelzések adatgyűjtési rétegének fontos elemét képezik, mivel ezek az információk közvetlenül befolyásolják a rendszer fejlesztését és finomítását. A kérdőívek és használhatósági tesztek segítségével közvetlen kapcsolatba léphetünk a felhasználókkal, és megérthetjük, hogyan érzékelik a rendszert. A kérdőívek segítségével a felhasználók véleményét közvetlenül lehet gyűjteni a rendszer különböző funkcióiról, teljesítményéről és

felhasználhatóságáról. Ez különösen hasznos, amikor a rendszer további fejlesztéseihez és finomításához szükséges információkat szeretnénk gyűjteni. A használhatósági tesztek révén a fejlesztők közvetlenül figyelhetik meg a felhasználókat, miközben azok használják a rendszert. Ezek a tesztek lehetőséget nyújtanak arra, hogy azonosítsák azokat a területeket, ahol a felhasználók nehézségekbe ütköznek, és ennek megfelelően optimalizálják a felhasználói felületet. Ezek az adatok együttesen lehetővé teszik a fejlesztők számára, hogy gyorsan reagáljanak a felhasználói igényekre és problémákra, ami folyamatosan növeli a rendszer felhasználhatóságát és felhasználói elégedettségét.

Fejlesztői tevékenységi adatok (Kódváltoztatások, hibák, commitok)

A fejlesztői tevékenységi adatok központi szerepet játszanak abban, hogy a fejlesztési folyamat zökkenőmentesen és hatékonyan haladjon előre. Ezek az adatok nyomon követik a kódolási folyamat minden aspektusát, beleértve a kódváltoztatásokat, a commitokat és a hibák javítását [40] [41]. Ezek az információk segítik a fejlesztőket abban, hogy átlátható képet kapjanak a projekt haladásáról, és azonosítsák azokat a területeket, ahol további fejlesztések vagy hibajavítások szükségesek. A kódváltoztatások követése lehetővé teszi a fejlesztők számára, hogy pontosan lássák, milyen módosítások történtek a kódon, és mely részek igényelnek további figyelmet. A változtatások rögzítése segít visszatekinteni a korábbi verziókra, és azonosítani azokat a pontokat, ahol hibák vagy teljesítményproblémák merültek fel. Ez az átláthatóság biztosítja, hogy a fejlesztési folyamat során a csapat minden tagja tisztában legyen az aktuális állapottal, és gyorsan reagálhassanak a felmerülő problémákra. A commitok rögzítése szintén alapvető fontosságú a fejlesztési folyamat során. A commitok jelzik a kód állapotának változásait, és fontos mérföldköveket képviselnek a fejlesztési ciklusban. A commitok elemzése segít meghatározni, hogy a csapat milyen gyakran hajt végre módosításokat, és ezek a változások hogyan befolyásolják a projekt előrehaladását. A commitok alapján nyomon követhető a kód minősége és a fejlesztési ütem, amely lehetővé teszi a projektmenedzserek számára, hogy időben beavatkozzanak, ha a fejlesztési ütem lassul, vagy ha a hibák száma növekszik [42] [43] [44]. A hibák nyomon követése és javítása kritikus eleme a fejlesztői tevékenységi adatoknak. A hibajavítások rögzítése segít azonosítani azokat a kódrészeket, ahol gyakran merülnek fel problémák, és amelyek további optimalizálást igényelnek. A hibák azonosítása és javítása kulcsfontosságú a projekt sikeres végrehajtása érdekében, hiszen ezek a problémák gyakran akadályozhatják a fejlesztési folyamat előrehaladását. A hibajavítások gyors és hatékony kezelése lehetővé teszi, hogy a rendszer stabil maradjon, és a felhasználók elégedettsége ne csökkenjen. A fejlesztői tevékenységi

adatok összességében segítik a csapatot abban, hogy átláthatóan kövessék a projekt állapotát, és gyorsan reagáljanak a felmerülő problémákra. Ezek az adatok biztosítják, hogy a projekt időben haladjon előre, és hogy a rendszer fejlesztése során minden fontos mérföldkő rögzítésre kerüljön.

A hibák és commitok elemzése hozzájárul a projekt általános minőségének javításához, és segít minimalizálni a fejlesztési ciklus során felmerülő akadályokat.

Adatfeldolgozási réteg

Az adatfeldolgozási réteg kulcsfontosságú a mesterséges modell működésében, mivel itt történik meg a nyers adatok tisztítása, átalakítása, és az analitikai folyamatok előkészítése. Az összegyűjtött adatokat ezen a rétegen belül különböző lépések során dolgozzák fel, beleértve a hibák követését, az anomáliák észlelését, az adatok tisztítását, átalakítását, valamint a jellemzők kivonását. A megfelelően feldolgozott adatok biztosítják, hogy a mesterséges intelligencia modellek pontosan tudják előre jelezni a felhasználói viselkedést, optimalizálni a fejlesztési folyamatokat és biztosítani a projekt sikerét. A következő alfejezetek részletesen bemutatják az adatfeldolgozási réteg egyes komponenseit.

Hibakövetés (Hibajelentés, összeomlás elemzések)

A hibakövetés az adatfeldolgozási réteg egyik alapvető komponense, amely folyamatosan nyomon követi a rendszerben előforduló hibákat és összeomlásokat, valamint ezek részletes elemzését végzi. A hibakövetési rendszer célja, hogy a hibákat minél gyorsabban felismerje, és részletes jelentéseket készítsen a fejlesztők számára, hogy azok azonnal beavatkozhatnak. Az ilyen típusú adatok feldolgozása és elemzése lehetővé teszi a fejlesztési folyamat folyamatos optimalizálását, és biztosítja, hogy a rendszer megbízhatósága ne csökkenjen. A hibajelentések általában a felhasználók és a fejlesztők visszajelzésein alapulnak, amelyek rögzítik, hogy milyen problémák merültek fel a rendszer használata közben. Ezek a jelentések gyakran tartalmazzák a felhasználók által végrehajtott műveleteket, amelyek hibát okoztak, valamint a hibakódokat, amelyek segítenek a fejlesztőknek diagnosztizálni a problémát. A hibajelentések részletes elemzése feltárja a leggyakoribb hibákat, lehetővé téve a fejlesztők számára, hogy célzott javításokat végezzenek a rendszer stabilitásának és megbízhatóságának növelése érdekében. Az összeomlás elemzések szintén kritikus szerepet játszanak a rendszer hibáinak kezelésében. Az ilyen jelentések részletes információkat tartalmaznak arról, hogy miért és hogyan következett be a rendszer összeomlása, beleértve az érintett kódrészeket, a naplófájlokat és a rendszer futási környezetét. Az összeomlási jelentések lehetőséget biztosítanak arra, hogy

a fejlesztők azonosítsák azokat a kritikus pontokat, ahol a rendszer instabilitása jelentkezik, és gyorsan beavatkozhatnak a probléma megszüntetése érdekében. Az ilyen típusú elemzések segítségével a fejlesztők minimalizálhatják az összeomlások kockázatát, és biztosíthatják, hogy a rendszer megbízható maradjon, függetlenül attól, hogy mekkora terhelés alatt áll.

A hibakövetési rendszer automatizált elemzési eszközökkel dolgozik, amelyek lehetővé teszik a hibák gyors felismerését és kategorizálását. Ezáltal a fejlesztők pontosan tudják, mely hibák igényelnek azonnali figyelmet, és melyek azok, amelyek kisebb prioritást élveznek. Az ilyen rendszerek alkalmazása hozzájárul a hibajavítások hatékonyságának növeléséhez, mivel a hibák felderítése és elemzése sokkal gyorsabban és átfogóbban történik, mint manuális módszerekkel.

Adatok anomális észlelése (Mintázat elemzés)

Az anomáliák az adatokban olyan eltérések, amelyek nem illeszkednek a normál viselkedési mintákhoz, és sokszor olyan problémákra utalnak, amelyeket a hagyományos elemzési módszerek nem tudnak könnyen azonosítani.

Az anomáliák felismerése során az adatfeldolgozó rendszer mintázatelemzési technikákat alkalmaz, amelyek segítségével azonosítja azokat az adatpontokat, amelyek eltérnek a megszokott viselkedéstől. Az anomáliák sokféle formában jelentkezhetnek, beleértve a szélsőséges adatokat, a hibásan rögzített adatpontokat, valamint azokat az eseteket, amikor egy váratlan esemény befolyásolja az adatok minőségét. Ezek a szokatlan mintázatok komoly problémákat okozhatnak, ha nem észleljük őket időben, ezért az anomáliák felismerése létfontosságú a rendszer stabil működéséhez. Az anomáliák felismerése gyakran gépi tanulási algoritmusok segítségével történik, amelyek képesek megtanulni az adatbázis normál mintázatait, és azonnal felismerni, ha egy adatpont eltér ezektől a mintázatoktól. Az ilyen algoritmusok lehetővé teszik, hogy a rendszer automatikusan azonosítsa a problémákat, és figyelmeztetést küldjön a fejlesztőknek, mielőtt az anomáliák jelentős károkat okoznának. Az anomália észlelési rendszer kulcsszerepet játszik abban, hogy a rendszer képes legyen önállóan felismerni a hibákat, és proaktívan kezelni azokat, mielőtt komolyabb következményekkel járnának. Az adatokban előforduló anomáliák az esetek többségében olyan problémákat jeleznek, amelyek nagyobb vizsgálatot igényelnek. Ezek lehetnek technikai problémák, például szoftverhibák vagy hardverproblémák, de utalhatnak adatszivárgásra vagy biztonsági incidensekre is. Az anomália észlelési rendszerek alkalmazása lehetővé teszi, hogy a fejlesztők

pontosan lássák, hol szükséges beavatkozni, és milyen problémákat kell azonnal kezelni a rendszer stabilitásának megőrzése érdekében.

Adattisztítás (Zaj eltávolítása, formázás)

Az adattisztítás alapvető lépés az adatfeldolgozási folyamatban, mivel a nyers adatok gyakran tartalmaznak zajt, hibákat vagy hiányos adatokat, amelyek csökkenthetik a rendszer által végzett elemzések pontosságát. A tisztítási folyamat során a rendszer eltávolítja az irreleváns adatokat, kijavítja a hibákat és kitölti a hiányzó adatokat, hogy biztosítsa az adatok megbízhatóságát és felhasználhatóságát. Ezek az adatok általában véletlenszerű hibákból, érzékelési problémákból vagy emberi tévedésekből származnak, és az eltávolításuk elengedhetetlen a pontos eredmények elérése érdekében. A rendszer automatikusan felismeri a zajos adatokat, és eltávolítja őket az adatállományból. A formázás során az adatok egységes formátumba kerülnek, hogy azok kompatibilisek legyenek a későbbi elemzési folyamatokkal. Az adatforrásokból érkező adatok gyakran eltérő struktúrával rendelkeznek, és szükséges, hogy ezek az adatok egy szabványos formátumba kerüljenek, amelyet a rendszer könnyen kezelni tud. Az adatok egységesítése biztosítja, hogy az elemzések során minden adat megfelelően összehasonlítható legyen.

Nagy adathalmazok elemzése (Statisztikai módszerek, korreláció)

A rendszer különböző statisztikai módszereket és korrelációs elemzéseket alkalmaz annak érdekében, hogy a nagy adathalmazokban meglévő mintázatokat és összefüggéseket azonosítsa. A statisztikai módszerek lehetővé teszik az adatok kvantitatív elemzését, amely során a rendszer különböző mutatókat és értékeléseket készít az adatok alapján. Ezek a módszerek magukban foglalják az átlagok, szórások, mediánok és egyéb statisztikai értékek kiszámítását, amelyek segítenek az adatok jobb megértésében. A statisztikai elemzés során a rendszer képes azonosítani a rendellenes értékeket, anomáliákat és egyéb problémákat, amelyek befolyásolhatják az elemzési eredményeket. A korrelációs elemzés célja, hogy az adathalmazban meglévő összefüggéseket tárja fel. A korrelációs elemzés segítségével a rendszer azonosítani tudja azokat az adatpontokat, amelyek között statisztikai kapcsolat áll fenn, és így megértheti, hogy az egyes változók hogyan hatnak egymásra. Az ilyen típusú elemzés különösen fontos, amikor komplex rendszerekben dolgozunk, ahol számos különböző tényező együttesen befolyásolja az eredményeket.

Jellemzők kivonása (Releváns adatok kiválasztása)

A jellemzők kivonása kulcsfontosságú lépés az adatfeldolgozás során, amelynek célja a nagy mennyiségű adatból a leglényegesebb és leginkább releváns információk kiválasztása. Ez a folyamat alapvető annak érdekében, hogy az adatfeldolgozó rendszer hatékonyan tudjon működni, hiszen a nyers adatok között gyakran sok olyan elem található, amely nem járul hozzá közvetlenül a célzott elemzésekhez. A jellemzők kivonása segít az adatmennyiség optimalizálásában, csökkentve az elemzéshez szükséges számítási kapacitást, miközben biztosítja, hogy az adatok lényegi elemei ne vesszenek el.

Az eye-tracking adatok feldolgozásakor különösen fontos szerepet játszik a hőtésképek és egyéb vizuális eszközök használata a jellemzők kivonásához. Az eye-tracking technológiából származó adatok óriási mennyiségű információt tartalmaznak a felhasználók tekintetének mozgásáról, fixációiról és szakkádjaikról. A hőtésképek segítenek ezeket az adatokat vizuálisan megjeleníteni, kiemelve azokat a területeket, ahol a felhasználók tekintete leggyakrabban megpihen, és ahol a legnagyobb figyelmi koncentráció jelentkezik. Ezek a vizualizációk nemcsak a felhasználói viselkedést teszik könnyebben érthetővé, hanem lehetővé teszik, hogy a rendszer automatikusan azonosítsa a felhasználói élmény szempontjából legfontosabb elemeket, amelyeket érdemes további elemzésekre felhasználni.

A tekintetmintázat elemzés szintén fontos részét képezi a jellemzők kivonásának. A tekintetmintázatok segítségével a rendszer képes azonosítani, hogyan navigálnak a felhasználók a felületen, mely elemeket nézik meg először, és milyen sorrendben járják be a képernyőn látható információkat. Az ilyen mintázatok különösen hasznosak lehetnek a felhasználói felületek optimalizálásában, mivel azonosítani lehet azokat a területeket, ahol a felhasználók elakadnak vagy bizonytalanok. A jellemzők kivonása nemcsak az eye-tracking adatokra korlátozódik, hanem a fejlesztői tevékenységekből, hibajelentésekből és egyéb adatokból is kiemeli a releváns információkat. Például a hibajelentések esetében a rendszer automatikusan azonosítja azokat a kódrészeket, ahol a hibák a leggyakrabban jelentkeznek, vagy ahol az összeomlási jelentések szerint rendszeresen problémák lépnek fel. Ezek az információk lehetővé teszik a fejlesztők számára, hogy célzottan javítsák a rendszer azon elemeit, amelyek a legnagyobb hatással vannak a stabilitásra. A jellemzők kivonása során alkalmazott módszerek magukban foglalják a statisztikai elemzést, amely során az adatok különböző mutatói alapján kiválasztják a leginkább informatív változókat. A gépi tanulási algoritmusok szintén fontos szerepet játszanak a jellemzők kivonásában, mivel ezek a rendszerek képesek automatikusan megtanulni, mely adatpontok relevánsak a predikciókhoz vagy az osztályozási feladatokhoz. A

jellemzők kivonása biztosítja, hogy a rendszer ne dolgozzon felesleges adatokat, ezzel csökkentve a számítási terheket és növelve az elemzési folyamatok hatékonyságát. A releváns adatok kiválasztása optimalizálja a modell teljesítményét, és lehetővé teszi, hogy a rendszer gyorsan és pontosan reagáljon a változó körülményekre. Ezen folyamat segítségével az adatelemzők olyan információkhoz juthatnak, amelyek közvetlenül segítik a döntéshozatali folyamatokat, javítva a rendszer által nyújtott eredményeket. A jellemzők kivonása tehát nem csupán a redundáns és irreleváns adatok kiszűréséről szól, hanem a lényegi információk azonosításáról, amelyek lehetővé teszik, hogy a rendszer hatékonyabban és pontosabban működjön. Ez a folyamat kulcsszerepet játszik a rendszer teljesítményének növelésében, mivel biztosítja, hogy az adatok valódi értékkel bírjanak a predikciós és döntéshozatali folyamatok során.

4.2.2. Mesterséges intelligencia réteg

A Mesterséges Intelligencia réteg az adatfeldolgozási folyamat kulcskomponense, amely a nyers adatokat elemzi és használja fel predikciók, döntések és optimalizálási javaslatok előállítására. Ebben a rétegben különféle gépi tanulási algoritmusok dolgoznak együtt, hogy a rendszer számára minél több értékes információt nyerjenek ki az adatbázisból, és ezáltal hatékonyabbá tegyék a fejlesztési folyamatokat. Az MI réteg nemcsak az aktuális adatok alapján hoz döntéseket, hanem képes azonosítani a rejtett mintázatokat, előrejelzéseket készíteni a jövőbeli eseményekre, valamint figyelmeztetni az esetleges anomáliákra. Az alábbiakban bemutatom a mesterséges intelligencia réteg legfontosabb alkotóelemeit, amelyek szerepet játszanak a rendszerben.

Support Vector Machine osztályok szétválasztása

A Support Vector Machine (SVM) egy rendkívül erős gépi tanulási algoritmus, amely az osztályozási feladatok során különösen hatékynak bizonyul. Az SVM célja, hogy egyértelmű határvonalat húzzon az adatok különböző osztályai között, miközben maximális távolságot biztosít a különböző osztályok határán elhelyezkedő adatpontok között [45] [46]. Az SVM működése során egy hiper-síkot hoz létre, amely elválasztja az osztályokat úgy, hogy minimalizálja az osztályokon belüli hibákat, és a lehető legpontosabb predikciókat adja. Az SVM előnye különösen akkor mutatkozik meg, amikor az adatok komplexek és többdimenziósak, vagyis sok különböző tényező befolyásolja az osztályozási eredményt. A felhasználói viselkedési minták elemzésekor az SVM segít az adatok különböző osztályokba sorolásában, például felismeri a felhasználói csoportokat, akik eltérően reagálnak a felületi elemekre, vagy különböző módon használják a rendszert [47] [48]. Az algoritmus lehetővé

teszi, hogy a fejlesztők jobban megértsék a felhasználói viselkedést, és azonosítsák azokat a pontokat, ahol a felhasználói élmény javítására van szükség. Az SVM gyakran olyan problémák megoldására szolgál, ahol fontos, hogy az osztályok közötti különbségek világosan elkülöníthetők legyenek. Az algoritmus különösen hatékony abban, hogy az adatok nem lineáris módon osztályozhatók, vagyis amikor az osztályok közötti határok nem egyszerű geometriai alakzatok mentén húzhatók meg. Az SVM erőssége abban rejlik, hogy egy optimalizált hipersíkot keres, amely a lehető legjobban szétválasztja az osztályokat, miközben minimalizálja az esetleges hibákat. A mesterséges intelligencia rétegben az SVM használata kulcsfontosságú az osztályozási feladatok automatizálásában. Ez a gyors és pontos döntéshozatal elősegíti a rendszer hatékonyságát, mivel az adatok szétválasztása révén a felhasználói viselkedést, a hibák kategorizálását és egyéb fontos döntési mechanizmusokat is támogat.

K-Means klaszterezés (Hasonló csoportok azonosítása)

A K-Means klaszterezés egy rendkívül népszerű, nem felügyelt tanulási módszer, amely az adatok strukturálását szolgálja. Az algoritmus célja, hogy az adatok között rejlő hasonlóságokat felismerje, és azokat csoportokra (klaszterekre) bontsa. Az adatok középpontja körül elhelyezkedő klaszterekbe rendezve az algoritmus az egyes adatpontokat a legközelebbi középpontokhoz rendeli hozzá, így az adatok hasonló csoportokba kerülnek, ahol a klasztereken belüli pontok közelebb vannak egymáshoz, mint a más klaszterekhez tartozó pontokhoz. Az algoritmus egyik legnagyobb előnye, hogy képes automatikusan felfedezni az adatok belső szerkezetét, anélkül, hogy előzetesen felcímkézett adatokra lenne szükség. A K-Means algoritmus először véletlenszerűen választ középpontokat a csoportokhoz, majd az adatpontok hozzárendelésével és a középpontok folyamatos frissítésével addig finomítja a klasztereket, amíg azok pontosan tükrözik az adatok szerkezetét. [49] [50] [51]. A felhasználói adatok klaszterezése során a K-Means algoritmus segíthet azonosítani a felhasználói viselkedési mintázatokat, csoportosítani a hasonló felhasználókat, vagy akár feltárni a különböző típusú hibákat, amelyeket a felhasználók elkövetnek a rendszer használata közben. Az algoritmus segítségével a rendszer automatikusan felismerheti a leggyakoribb hibákat vagy azokat a fejlesztési területeket, ahol további optimalizálás szükséges. Az adatok ilyen típusú elemzése lehetővé teszi a rendszer számára, hogy célzott javaslatokat tegyen a fejlesztőknek a rendszer javítása érdekében. A K-Means algoritmus alkalmazása nemcsak a felhasználói adatok elemzésében segít, hanem a fejlesztői tevékenységek és hibák csoportosításában is. Például a rendszer felismerheti, hogy egy adott fejlesztői csoport rendszeresen hasonló típusú hibákat követ el, ami lehetővé teszi a problémák célzott kezelését. Az MI rétegben a K-Means

használata hatékony eszközt biztosít a fejlesztők számára a rendszeren belüli mintázatok gyors felismerésére, ami javítja a fejlesztési folyamatok hatékonyságát és a végfelhasználói élményt.

Random Forest (Többszörös döntési fák)

A Random Forest működési elve szerint több, egymástól független döntési fát hoz létre, amelyek külön-külön elemzik az adatokat, majd ezek eredményeit egyesítve hoznak meg egy végső döntést. Ez a módszer rendkívül hasznos olyan predikciós feladatok során, ahol több változó befolyásolja az eredményt, mivel a több fa együttes alkalmazása csökkenti a túlilleszkedés esélyét és növeli a modell pontosságát. A döntési fák olyan algoritmusok, amelyek az adatokat logikai útvonalak mentén osztályozzák, az egyes elágazási pontoknál döntéseket hozva az adatok változói alapján. A Random Forest algoritmus ezt a technikát továbbfejleszti azáltal, hogy a döntési fákat variálja, különböző adatmintákat használ, és az egyes fák eredményeit kombinálva pontosabb előrejelzéseket készít. Az algoritmus alkalmazása különösen akkor előnyös, ha sok változóval rendelkező adatbázisokkal dolgozunk, mivel képes kezelni a változók közötti bonyolult kapcsolatrendszereket, és kiszűrni azokat a változókat, amelyek nem relevánsak az adott predikciós feladat szempontjából. [52] [53] [54]

A Random Forest jelentős alkalmazása a fejlesztői tevékenységek monitorozása, ahol az algoritmus képes azonosítani azokat a változókat, amelyek befolyásolják a fejlesztési folyamat hatékonyságát. Például az algoritmus megfigyelheti, hogy melyik fejlesztői csoportok teljesítenek a legjobban, és milyen változók határozzák meg a csapat produktivitását. Ezen információk alapján a rendszer optimalizálási javaslatokat tehet a projektmenedzsment számára, hogy javítsák az erőforrások elosztását, és biztosítsák, hogy a projekt hatékonyan haladjon előre. Az algoritmus egyik legnagyobb előnye, hogy az eredményeket nemcsak predikciók formájában jeleníti meg, hanem a döntési fák révén világosan követhetővé teszi azokat a lépéseket, amelyek a predikcióhoz vezettek. Ez átláthatóságot és biztonságot nyújt a rendszer számára, mivel a projektmenedzserek pontosan láthatják, hogy milyen döntések születtek és miért.

Döntési fák (Predikciós utak)

A döntési fák egyszerűek és könnyen érthetők, ami különösen fontossá teszi őket a mesterséges intelligencia rendszerekben, ahol a döntési folyamatokat átláthatóvá kell tenni. Az algoritmus működése során egy fa szerkezetet hoz létre, amelyben minden csomópont egy döntést vagy kérdést képvisel, míg az egyes elágazások különböző kimenetekhez vezetnek. Az adatok különböző útvonalakon haladnak át, és az egyes csomópontoknál elvégzett döntések alapján osztályozódnak vagy predikciók születnek. A döntési fák egyik legnagyobb előnye, hogy a

folyamat minden lépése érthető és nyomon követhető, ami megkönnyíti a rendszer belső működésének megértését. Ez különösen hasznos a projektmenedzserek és fejlesztők számára, akiknek meg kell érteniük, hogy miért és hogyan hozott a rendszer egy adott döntést. [52] [53] [54] Az algoritmus működése során az adatokat olyan logikai struktúrába rendezi, amely alapján egyértelműen látható, hogy az egyes változók hogyan befolyásolják az eredményt. Mivel minden egyes csomópont egy döntést képvisel, az algoritmus lehetővé teszi, hogy az adatok különböző kategóriákba sorolódjanak, attól függően, hogy a döntési fák hogyan határozzák meg a legvalószínűbb kimenetelt. A predikciós feladatok során a döntési fák segíthetnek azonosítani a felhasználói viselkedést, előre jelezni a hibákat, vagy meghatározni, hogy melyik fejlesztési döntés hozza a legjobb eredményt. Például, ha a rendszer egy új funkció bevezetését tervezi, a döntési fák segítségével előre jelezhető, hogy a felhasználók hogyan reagálnak majd az új funkcióra, és mely tényezők befolyásolják a felhasználói elégedettséget. Az ilyen típusú előrejelzések segítenek a projektmenedzsereknek abban, hogy jobban felkészüljenek a felhasználói visszajelzésekre és minimalizálják a rendszer működésében fellépő hibákat.

Erőforrás-optimalizálás (csapat allokáció, időgazdálkodás)

Az erőforrás-optimalizálás az MI réteg egyik kulcsfontosságú funkciója, amely segít abban, hogy a projekt során a rendelkezésre álló erőforrásokat hatékonyan osszák el. A mesterséges intelligencia rendszerek képesek figyelemmel kísérni a fejlesztési folyamatokat, a csapat teljesítményét, valamint a projekt ütemtervét, és ezek alapján javaslatokat tenni az erőforrások legjobb kihasználására. Az optimalizálás nemcsak a csapatok erőforrásaira vonatkozik, hanem a fejlesztési időre és a feladatok priorizálására is. Az erőforrás-optimalizálás egyik legfontosabb része a csapat allokáció, amely során a rendszer figyelemmel kíséri a csapatok teljesítményét, és javaslatokat tesz arra, hogy mely feladatokat kell elvégezni az egyes csapatoknak. Ez különösen fontos a nagy projektek esetében, ahol több fejlesztői csoport dolgozik párhuzamosan különböző feladatokon. A rendszer automatikusan azonosíthatja azokat a csapatokat, amelyek nagyobb terhelés alatt állnak, és javaslatokat tehet a feladatok átcsoportosítására, hogy a projekt egyenletesen haladjon előre. Az időgazdálkodás szintén az erőforrás-optimalizálás része, ahol a rendszer segít a projekt ütemtervének pontos betartásában. Az MI rétegben az erőforrás-optimalizálás nagyban hozzájárul a rendszer hatékonyságának növeléséhez, mivel segít azonosítani azokat a területeket, ahol javításra van szükség, és javaslatokat tesz a fejlesztési folyamatok finomhangolására. A rendszer figyelemmel kíséri a projekt minden szakaszát, és folyamatosan értékeli a teljesítményt, hogy azonosítsa azokat a

tényezőket, amelyek lassíthatják a fejlesztési folyamatokat. Például, ha egy csapat túlterhelt, vagy egy feladat a tervezettnél hosszabb időt vesz igénybe, a rendszer előre figyelmeztethet a határidők csúszásának lehetőségére. Az ilyen előrejelzések lehetővé teszik a projektmenedzserek számára, hogy időben beavatkozzanak, és átcsoportosítsák az erőforrásokat, vagy új stratégiákat vezessenek be a határidők betartása érdekében.

4.2.3. Visszacsatolási réteg

A visszacsatolási réteg az adatfeldolgozási és mesterséges intelligencia réteg eredményeit használja fel a fejlesztési folyamat javítására, a csapatok hatékonyságának növelésére, a hibák prioritizálására és a projektmenedzsment döntéshozatalának optimalizálására. Ez a réteg fontos szerepet játszik abban, hogy a rendszer által elemzett adatokból hasznos információkat nyerjünk ki, amelyek segítenek a fejlesztési és menedzsment tevékenységek hatékonyabbá tételében. Az alábbiakban részletesen bemutatásra kerülnek a visszacsatolási réteg fő komponenseit, amelyek mind hozzájárulnak a projekt sikerességéhez.

Kódszerkezet javaslatok (Refaktorálás, modularitás)

A kódszerkezet javítása kiemelt fontosságú a fejlesztési folyamatok során, mivel a jól szervezett kód könnyebben karbantartható, gyorsabb fejlesztést és kevesebb hibát eredményez. Az MI rendszer figyelemmel kíséri a kód szerkezetét, és azonosítja azokat a területeket, ahol a kód refaktorálása vagy modularizálása szükséges lehet. Ezek a javaslatok elősegítik a fejlesztők munkáját, mivel lehetővé teszik, hogy a kód egyszerűbbé és hatékonyabbá váljon, miközben megőrzi annak funkcionalitását. A refaktorálás célja, hogy a meglévő kódot átstrukturáljuk anélkül, hogy annak viselkedését megváltoztatnánk. Ez magában foglalhatja az ismétlődő kódrészletek megszüntetését, a túl bonyolult kód egyszerűsítését, valamint a kód olvashatóságának javítását. Az MI rendszernek fel kell ismernie azokat a részeket, ahol a kód túlságosan összetett vagy nehezen követhető, és javaslatokat tehetnek arra, hogyan lehetne azt egyszerűbbé és áttekinthetőbbé tenni. A refaktorálás hozzájárul ahhoz, hogy a kód kevesebb hibát tartalmazzon, mivel a tisztább struktúra könnyebben érthetővé és tesztelhetővé teszi azt. A modularitás növelése szintén fontos szerepet játszik a kódszerkezet optimalizálásában. A modularizált kód kisebb, önálló egységekre bontja a fejlesztést, amelyek egymástól függetlenül működhetnek. Ez lehetővé teszi, hogy a fejlesztők egy adott kódrészen dolgozzanak anélkül, hogy az egész rendszer működését befolyásolnák.

Hibajavítási prioritások (Elsőbbséget élvező problémák)

A hibajavítások prioritásának meghatározása kulcsfontosságú része a fejlesztési folyamatnak, mivel a rendszer megbízhatósága és a felhasználói élmény fenntartása a kritikus hibák gyors javítását igényli. A visszacsatolási réteg ezen funkciója lehetővé teszi a fejlesztők számára, hogy a legfontosabb hibákat elsőként kezeljék, ezáltal növelve a rendszer megbízhatóságát és csökkentve a felhasználók számára okozott kellemetlenségeket.

Az elsőbbséget élvező hibák általában olyan problémák, amelyek jelentős mértékben befolyásolják a rendszer működését vagy a felhasználói élményt. Például, ha egy funkció teljesen leáll, vagy ha a rendszer összeomlik, ezek a hibák azonnali figyelmet igényelnek. A modul figyelemmel kíséri az ilyen eseményeket, és automatikusan magas prioritásúként jelöli meg azokat, hogy a fejlesztők azonnal elkezdhessék a hibák javítását. Az ilyen problémák gyors megoldása elengedhetetlen a felhasználói elégedettség fenntartásához, hiszen a kritikus hibák elhárítása biztosítja a szolgáltatás folyamatosságát. Az MI keret által biztosított hibajavítási prioritások lehetővé teszik, hogy a fejlesztők hatékonyabban kezeljék az erőforrásaikat. Az algoritmusok figyelemmel kísérik a felhasználói visszajelzések gyakoriságát és súlyosságát, valamint a hibák hatását a rendszerre, és ennek alapján rangsorolják a problémákat. Ez azt jelenti, hogy azok a hibák, amelyek nagy hatással vannak a felhasználókra vagy a rendszer teljesítményére, előre kerülnek a javítási listán, míg a kisebb jelentőségű problémák alacsonyabb prioritást kapnak. A hibajavítási prioritások meghatározása jelentős mértékben hozzájárul a rendszer stabilitásának fenntartásához, mivel a legkritikusabb problémák gyors megoldása minimalizálja a szolgáltatás kiesését és a felhasználók elégedetlenségét. Az MI segítségével a fejlesztők pontosan láthatják, mely hibák igényelnek azonnali beavatkozást, és hogyan oszthatják el a csapat erőforrásait a legnagyobb hatékonyság érdekében. Ez nemcsak a hibajavítási folyamatot teszi gyorsabbá, hanem segít abban is, hogy a fejlesztési ciklusok gördülékenyebbek legyenek, hiszen a kisebb problémákat is fokozatosan el lehet hárítani anélkül, hogy a projekt előrehaladását jelentős mértékben befolyásolnák.

Kockázatkezelés (Kockázatok azonosítása és mérséklése)

A kockázatkezelés az egyik legfontosabb tényező a projektmenedzsment és a fejlesztési folyamat során, mivel elősegíti a lehetséges problémák korai felismerését és azok mérséklését. Az MI rendszer rendkívül hasznos tud lenni a kockázatok felmérésében és kezelésében, mivel képesek azonosítani azokat a mintázatokat, amelyek alapján előre jelezhető, hogy egy adott projekt vagy rendszer milyen kihívásokkal nézhet szembe. A visszacsatolási rétegben a kockázatkezelés alapvető szerepet játszik, mivel az MI algoritmusok folyamatosan figyelik az

adatokat és előrejelzéseket készítenek arra vonatkozóan, hogy hol és milyen problémák merülhetnek fel. Az azonosított kockázatok lehetnek technikai, pénzügyi vagy erőforrás-alapú kihívások, amelyek befolyásolhatják a projekt sikerességét. Például, ha egy adott projekt szakaszban a fejlesztési idő jelentősen elmarad a tervezettől, az MI rendszerek figyelmeztetést küldenek, hogy a projektmenedzserek előre lássák a lehetséges késedelmeket. A kockázatsökkentés az MI által szolgáltatott adatok alapján történik. A kockázatsökkentési javaslatok segítenek a projektmenedzsereknek abban, hogy előre lássák a felmerülő problémákat, és még a problémák megjelenése előtt intézkedéseket hozzanak azok elkerülésére. Az MI által nyújtott kockázatkezelési megoldások lehetővé teszik a projektmenedzserek számára, hogy proaktív módon irányítsák a projektet, minimalizálják a lehetséges veszteségeket, és biztosítsák, hogy a projekt határidőn belül és a költségvetési korlátok között maradjon.

Projektmenedzsment javaslatok (Erőforrás-allokáció, priorizálás)

A projektmenedzsment javaslatok alapvető szerepet játszanak abban, hogy a fejlesztési folyamat során az erőforrásokat és feladatokat optimálisan osszák el. Az MI rendszerek figyelemmel kísérik a csapatok teljesítményét, a feladatok előrehaladását és a projekt általános állapotát, és ezek alapján javaslatokat tesznek arra vonatkozóan, hogyan lehetne az erőforrásokat hatékonyabban felhasználni. A visszacsatolási réteg ezen funkciója biztosítja, hogy a projektmenedzserek mindig a legfrissebb adatok alapján hozhassanak döntéseket, és minimalizálják a projekt során felmerülő problémákat. Az erőforrás-allokáció során az MI algoritmusok figyelemmel kísérik, hogy mely csapatok vagy projektszakaszok igényelnek több támogatást, és javaslatokat tesznek arra vonatkozóan, hogyan lehet az erőforrásokat egyenletesen elosztani. Például, ha egy csapat túlterhelt, az MI figyelmeztetést küldhet a projektmenedzsereknek, és javasolhatja, hogy további erőforrásokat csoportosítsanak át annak érdekében, hogy a feladatok időben befejeződjenek. Az erőforrások hatékony allokációja nemcsak a csapatok terhelését csökkenti, hanem a projekt általános sikerességét is növeli, mivel minden csapat pontosan annyi támogatást kap, amennyire szüksége van a feladatai elvégzéséhez. A feladatok priorizálása szintén központi szerepet játszik a projektmenedzsment javaslatokban.

Hatékonyagsnövelési ajánlások (Feladatpriorizálás, automatizálás)

A hatékonyságnövelési ajánlás segít a feladatok priorizálásában és automatizálásában, hogy a fejlesztési folyamat gyorsabb és hatékonyabb legyen. Az MI rendszer folyamatosan figyelik a fejlesztési ciklust, és azonosítják azokat a területeket, ahol a hatékonyság növelhető. Az ilyen

ajánlások lehetővé teszik a projektmenedzserek számára, hogy a legfontosabb feladatokra összpontosítsanak, miközben az automatizált folyamatok minimalizálják a manuális munka szükségességét. A feladatpriorizálás során az MI rendszerek figyelembe veszik a projekt sürgősségét és a feladatok fontosságát, és javaslatokat tesznek arra, hogy mely feladatokat kell előtérbe helyezni. A dinamikus feladatpriorizálás lehetővé teszi, hogy a fejlesztők és a projektmenedzserek rugalmasan reagáljanak a változó körülményekre, és biztosítsák, hogy a legfontosabb feladatok mindig időben elkészüljenek. Az automatizálás jelentős mértékben hozzájárul a fejlesztési folyamat hatékonyságának növeléséhez.

Funkcionalitás javítása (Felhasználói interakció, használhatóság)

A funkcionalitás javítása az MI keret egyik legfontosabb feladata a visszacsatolási rétegben, mivel ez biztosítja, hogy a rendszer folyamatosan fejlődjön és megfeleljen a felhasználói igényeknek. A funkciók fejlesztése és optimalizálása a felhasználói visszajelzések és az MI által gyűjtött adatok alapján történik. Képes lehet azonosítani azokat a területeket, ahol a felhasználói interakciók nem a kívánt módon működnek, és javaslatokat tesznek a funkcionalitás javítására. A felhasználói interakció elemzése során az MI figyelemmel kíséri, hogyan használják a felhasználók a rendszert, és mely pontokon fordulnak elő nehézségek. Ha egy adott funkciót nem használnak megfelelően, vagy ha a felhasználók gyakran hibáznak egy adott feladat végrehajtásakor javaslatokat tehet arra, hogyan lehetne ezt a funkciót javítani. Ez lehet új funkciók hozzáadása, a meglévő funkciók átdolgozása, vagy egyszerűsített felhasználói felületek bevezetése. A használhatóság javítása szintén központi szerepet játszik a funkcionalitás fejlesztésében. Az MI képes lehet továbbá azonosítani azokat a részeket, ahol a felhasználói élmény javításra szorul, például, ha egy adott feladat túl bonyolult vagy nehéz a felhasználók számára. A használhatósági javaslatok lehetővé teszik a fejlesztők számára, hogy a felhasználói élményt optimalizálják, és biztosítsák, hogy a rendszer intuitív és könnyen használható legyen. A funkcionalitás javításával az MI rendszerek hozzájárulnak a felhasználói elégedettség növeléséhez, mivel a rendszer folyamatosan fejlődik és alkalmazkodik a felhasználói igényekhez. A funkciók fejlesztése és a használhatóság javítása révén a rendszer nemcsak hatékonyabbá válik, hanem biztosítja, hogy a felhasználók gyorsan és egyszerűen végezhessék el a feladataikat, anélkül, hogy nehézségekbe ütköznének.

UI/UX fejlesztések (elrendezés optimalizálásra, intuitív tervezés)

A UI/UX fejlesztések alapvető szerepet játszanak a felhasználói élmény javításában, mivel ezek a fejlesztések biztosítják, hogy a rendszer könnyen használható, vizuálisan vonzó és funkcionálisan hatékony legyen. Az MI folyamatosan elemzi a felhasználói interakciókat, és

javaslatokat tesznek a felhasználói felületek és a felhasználói élmény javítására. Az MI által nyújtott javaslatok lehetővé teszik a fejlesztők számára, hogy az elrendezést és a dizájnt optimalizálják, és biztosítsák, hogy a rendszer intuitív és felhasználó barát legyen. Az elrendezés optimalizálása során az MI rendszer figyeli, hogyan navigálnak a felhasználók a rendszerben, és azonosítja azokat a területeket, ahol az elrendezés javításra szorul. Például, ha egyes funkciók nehezen elérhetők, vagy ha a felhasználók gyakran eltévednek a menükben, a modul javaslatokat tehet az elrendezés átalakítására. Az optimalizált elrendezés lehetővé teszi, hogy a felhasználók gyorsan és könnyedén megtalálják a szükséges funkciókat, ami növeli a rendszer hatékonyságát és használhatóságát. Az optimalizált felület nemcsak a rendszer esztétikai megjelenését javítja, hanem hozzájárul ahhoz is, hogy a felhasználók kevesebb hibát kövessenek el, és gyorsabban tudják elvégezni a feladataikat.

5. Agilis fejlesztés támogatása meglévő univerzális modellel

A meglévő univerzális mesterséges intelligenci modellek integrálása szemmozgás-követési technológiákkal ígéretes megoldás lehet e problémák kezelésére. Ez a fejezet bemutatja, hogyan illeszthető egy szemmozgás-követő eszköz egy univerzális MI modellhez az agilis fejlesztési folyamatok támogatására, kiemelve a kód hőtésképes elemzésének szerepét az MI elemzés előtt és után. Az MI és szemmozgás-követési adatok integrálása egy többkomponensű rendszeren keresztül valósul meg. Az eye-tracking eszközök rögzítik a fejlesztők és felhasználók figyelmi mintáit, beleértve a fixációkat és szakkádokat. Ezek az adatok az MI modell előfeldolgozási moduljába kerülnek. Az univerzális MI modell az előfeldolgozott adatokat elemzi, és mintázatokat, anomáliákat azonosít a kódban. A rendszer hőtésképeket készít a kód vizuális elemzéséről az MI elemzés előtt és után, hogy megmutassa a fejlesztési hatékonyság növekedését. A kutatás során az eye-tracking által generált hőtésképek összehasonlítását végezzük az MI elemzések alapján. Az Eye-Tracking eszköz a kód különböző részeire vonatkozó figyelmi pontokat azonosít. Az MI elemzés előtt a hőtésképek gyakori fixációkat mutat egy hibás kódrészleten, jelezve a fejlesztők nehézségeit.

5.1. Az integrálási vizsgálat lefolyásának körülményei

Az alábbi szituációban a tanulók egy spaghetti kódznak nevezett, bonyolult, nehezen követhető programkódot fognak elemezni. Spaghetti kód jelenség gyakran előfordul kezdők fejlesztők körében, amit más fejlesztő nehezen tud értelmezni [55]. A feladat célja, hogy a tanulók gyakorlatban tapasztalják meg a bonyolult kódok elemzésének nehézségeit, és megértsék, hogyan segítheti ezt az MI-alapú technológia. Az elemzési folyamat során az eye-tracking eszköz követi a tanulók szemmozgását, rögzítve a kód különböző részeire irányuló figyelmi mintázatokat. A szemmozgások adatai alapján hőtésképek készülnek, amelyek vizuálisan ábrázolják, hogy mely kódrészletek vonzzák a legtöbb figyelmet, és hol időznek el a legtovább az elemzők. Ezek az információk segítenek feltárni, mely részek okozzák a legnagyobb nehézséget a megértésben, illetve, hogy a kód mely pontjai tartalmazhatnak potenciális hibákat vagy logikai következetlenségeket.

A hőtésképes adatokat ezt követően egy univerzális mesterséges intelligencia modell elemzi. A modell azonosítja a figyelmi minták mögötti okokat, például a bonyolult vezérlési szerkezeteket vagy az átláthatatlan logikai kapcsolatokat, és javaslatokat tesz a kód optimalizálására.

Az alábbi alfejezetben összefoglalásra kerülnek a vizsgálati eljárásban résztvevő személyek alapvető adatai, a felhasznált eszközök és maga a tesztelési környezet.

5.1.1. Vizsgálati személyek

A spagetti kód elemzésére és a mesterséges intelligenciával támogatott rendszer tesztelésére alkalmas környezetet a Dunaújvárosi Szakképzési Centrum Rudas Technikuma és Kollégiuma és Dunaújvárosi Egyetem biztosította. Az elemzésekre önkéntes alapon jelentkeztek a tanulók. Az adatokat minden résztvevő esetében anonim módon tároltuk, biztosítva a személyes adatok védelmét. Az elemzésekben összesen 32 fő (4 nő és 28 férfi) technikumi diák vett részt, akik érettségivel rendelkeztek, nem álltak gyógyszeres kezelés vagy tudatmódosító szerek hatása alatt, életkoruk pedig 18-20 év közé esett ($M=18,88$ $SD=0,61$). A tanulók mind tanultak és alkalmazták gyakorlatban C# programozási nyelvet.

5.1.2. Vizsgálati feladat

A rögzítés során egy .NET Framework környezetben írt C# programozási nyelvű kódsort elemeztek a tanulók. A kód megjelenítése fehér háttérű felületen, a Visual Studio alapértelmezett betűtípusával és stílusával történik, teljes képernyős módban. Annak érdekében, hogy a hőtérkép értelmezését ne zavarja a görgetés, a kétoldalas kód egymás mellett kerül megjelenítésre. A kódot az **5. ábra** írja le.

```

using System;
using System.Collections.Generic;

namespace ShopSystem
{
    class Program
    {
        static void Main(string[] args)
        {
            List<double> p = new List<double> { 9.99, 19.49, 5.75, 3.20 };
            List<string> n = new List<string> { "apple", "banana", "milk", "bread" };

            double t = 0;
            Dictionary<string, int> q = new Dictionary<string, int>();

            for (int i = 0; i < n.Count; i++)
            {
                Console.WriteLine($"Add {n[i]} to cart? y/n");
                string r = Console.ReadLine();
                if (r == "y")
                {
                    if (!q.ContainsKey(n[i]))
                        q[n[i]] = 1;
                    else
                        q[n[i]]++;
                    t += p[i];
                }
                else if (r == "n")
                {
                    continue;
                }
                else
                {
                    Console.WriteLine("Invalid input. Restarting...");
                    i = -1;
                    t = 0;
                    q.Clear();
                }
            }

            Console.WriteLine("Receipt:");
            foreach (var k in q)
            {
                Console.WriteLine($"{k.Key}: {k.Value} x {p[n.IndexOf(k.Key)]}");
            }
            Console.WriteLine("Total: $" + t);
        }
    }
}

```

4. ábra: A kód kezdetleges változata, amit elemeztek a diákok.
Forrás: Saját készítésű ábra.

A tanulónak a feladat végeztével egy A4 papírra nyomtatott Likert-sálán kellett értékelniük a kódot. A kérdések így szóltak:

i. Milyen nehéz volt a kód megértése?

1. Nagyon nehéz volt megérteni
2. Inkább nehéz volt
3. Semleges
4. Inkább könnyű volt
5. Nagyon könnyű volt megérteni

ii. Mennyire érezte logikusnak a kód szerkezetét?

1. Egyáltalán nem volt logikus
2. Inkább nem volt logikus

3. Semleges, részben logikus
4. Inkább logikus volt
5. Teljesen logikus volt

iii. Mennyire tudta követni a kódban használt változók és függvények kapcsolatát?

1. Egyáltalán nem tudtam követni
2. Inkább nem tudtam követni
3. Semleges, részben követhető volt
4. Inkább követhető volt
5. Teljesen követhető volt

iv. Mennyire tartja a kódot javításra szorulónak?

1. Egyáltalán nem szorul javításra
2. Inkább nem szorul javításra
3. Semleges, részben javítandó
4. Inkább javítandó
5. Teljes mértékben javítandó.

Ezután röviden össze kellett foglalniuk a kód funkcionalitását. A résztvevők mind helyesen foglalták össze a kód gyakorlati működését.

5.2. Likert-skálán adott válaszok értékelése és elemzése

A kérdőívben 15 tanuló válasza szerepelnek négy kérdés alapján, amelyek a kód érthetőségét, logikusságát, követhetőségét és javítási igényét vizsgálták. A válaszokat a X. táblázat írja le.

1. táblázat: Likert-skálán adott válaszok részletezése

Személy	1. Kérdés	2. Kérdés	3. Kérdés	4. Kérdés
Tanuló 1.	2	1	2	5
Tanuló 2.	2	1	2	5
Tanuló 3.	4	2	2	5
Tanuló 4.	4	2	2	4
Tanuló 5.	3	2	3	4
Tanuló 6.	4	2	2	5
Tanuló 7.	2	4	2	3
Tanuló 8.	2	3	3	4
Tanuló 9.	4	2	2	3
Tanuló 10.	4	2	2	3
Tanuló 11.	1	1	3	5
Tanuló 12.	2	3	2	4
Tanuló 13.	4	3	4	3
Tanuló 14.	3	3	4	2
Tanuló 15.	1	2	3	4
Átlag:	2,80	2,20	2,53	3,93
Medián:	3	2	2	4
Szórás:	1,15	0,86	0,74	0,96

A Likert-skála kérdésekre adott válaszok elemzése alapján megállapítható, hogy a diákok számára a kód megértése jelentős kihívást jelentett. Az első kérdés eredményei alapján a válaszadók többsége úgy ítélte meg, hogy a kód megértése nehéz volt, amit az átlagos és mediánértékek is alátámasztanak. A magas szórás azonban arra utal, hogy a résztvevők véleménye nem volt teljesen egységes, és néhányan talán könnyebbnek találták a feladatot. A kód logikai szerkezete különösen problémásnak bizonyult, mivel az ezzel kapcsolatos kérdésre adott válaszok átlagosan alacsonyabb értékeket mutattak. A válaszadók többsége azt jelezte, hogy a kódot inkább nem tartotta logikusnak, és a viszonylag alacsony szórás arra utal, hogy ebben a kérdésben a vélemények egységesebbek voltak. Ez azt sugallja, hogy a kód szerkezete jelentős mértékben hozzájárult a megértés nehézségeihez. A változók és függvények kapcsolatai szintén nehezen követhetők voltak a válaszadók szerint. Bár az átlagértékek kicsit jobbak voltak, mint a logikusságot érintő kérdésnél, az eredmények továbbra is azt mutatják, hogy a résztvevők nagy része nem tudta jól követni a kódot. Az alacsony szórás itt is azt jelzi, hogy a válaszadók többsége hasonlóan negatív tapasztalatokkal rendelkezett. Az utolsó kérdésre adott válaszokból világosan látszik, hogy a résztvevők túlnyomó többsége szerint a kód javításra szorul. Az átlag és medián magas értékei alapján egyértelmű, hogy a válaszadók egyöntetűen szükségesnek tartják a kód átdolgozását. Ez összhangban áll a többi kérdés

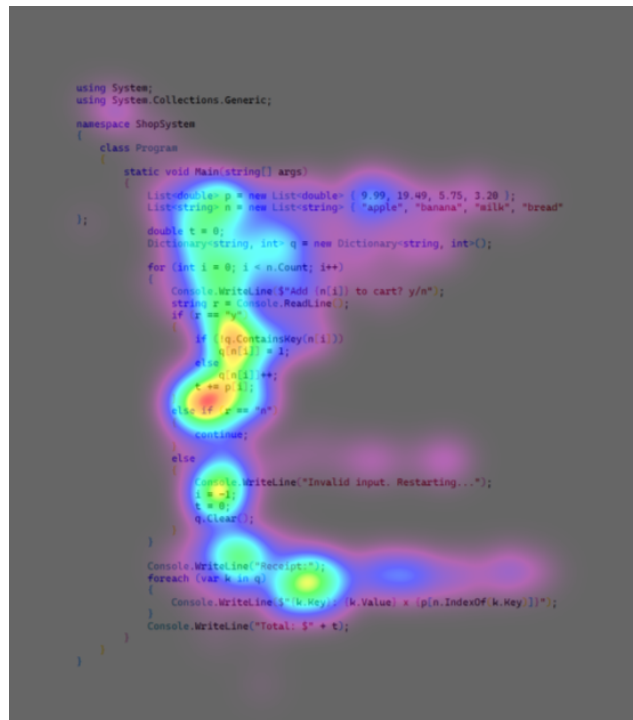
eredményeivel, amelyek már korábban is jelezték a kód szerkezetével és érthetőségével kapcsolatos problémákat.

Az elemzésből levonható következtetés, hogy a kód a tanulók számára nem volt kellően érthető és logikus, ami nagymértékben akadályozta a megértést és a követhetőséget. A válaszadók szerint a kód jelentős javításra szorul, különös tekintettel a változók elnevezésére, a logikai struktúra egyszerűsítésére és a szervezettebb kódolási gyakorlatokra. Ez arra is rávilágít, hogy a hasonló feladatok esetében fontos figyelembe venni a tanulók programozási szintjét és tapasztalatát, hogy a feladatok megfelelő kihívást nyújtsanak, de ne legyenek túl nehezek.

5.3. Hőtérkép alapú értékelés szemmozgás paraméterek alapján

A vizsgálat során a szemmozgáskövető eszköz rögzítette a tekintet mozgását, a fixációk helyét és időtartamát. A hallgatók a kódot számítógépen olvasták, miközben a szemmozgáskövető valós időben követte a szemük mozgását. A kód olvasása közben gyűjtött adatok alapján egy figyelemhőtérkép készült, amely vizuálisan ábrázolta a figyelem intenzitását a kód különböző részein. Terjedelmi korlátok miatt a jelen fejezetben csak egy hallgató hőtérképét mutatom be és elemzem részletesen. Ez az egy példa lehetővé teszi, hogy mélyebben megvizsgáljuk a szemmozgás adatok alapján készült figyelemhőtérképet, és annak összefüggéseit a kód olvasási folyamatával. Bár a többi résztvevő adatait nem részletezem, az elemzés során nyert következtetések általánosíthatóak, mivel a hallgatók figyelmi mintázatai összességében hasonlóak voltak.

A Tanuló 1. hőtérképét **5. ábra** mutatja meg, amely fixáció és szakkádok alapján lett generálva.



5. ábra: Tanuló 1. szemmozgás paraméterek alapján elkészített hőtérkép.
 Forrás: Saját készítésű ábra.

A bemutatott hőtérkép általános jellemzői alapján egyértelműen látszik, hogy a figyelmi mintázatok eltérő intenzitásúak a kód különböző részein. A hőtérkép meleg színekkel (piros, narancs) jelzi azokat a területeket, ahol a tekintet hosszabb ideig időzött, míg a hideg színek (zöld, kék, lila) a kevésbé figyelt részeket mutatják. Ez a vizuális eszköz jól tükrözi, hogy mely kódrészletek okoztak nagyobb kognitív terhelést a résztvevők számára, és melyek voltak könnyebben érthetőek.

A hőtérkép elemzése alapján kiderül, hogy a kód logikailag összetett részei, például a feltételek és a döntési struktúrák, kiemelt figyelmet kaptak. Ez a meleg színekkel jelölt részekenél figyelhető meg, amelyek a kognitív feldolgozás legintenzívebb pontjai. A meleg színek jelenléte a bonyolultabb kódrészleteknél arra utal, hogy a résztvevők kognitív erőfeszítést tettek ezek megértésére. Ez különösen igaz a döntési logikákra és a hibakezelésre, ahol a figyelmi idő hosszabb volt. A feltételkezelés és a cikluslogika erősen koncentrált figyelmet igényelt, ami azt sugallja, hogy ezek a részek nehezebben érthetőek. A deklarációk és egyszerű logikai műveletek, amelyek egyértelműek és könnyen érthetőek, kevésbé vonták el a figyelmet. Ezek a hideg színekkel ábrázolt részek, amelyek azt mutatják, hogy a résztvevők gyorsan átlépték őket.

5.4. Kódrefaktorálás szemmozgás paraméterek és az MI modell integrálása alapján

A hőtérkép mesterséges intelligencia modellként értelmezhető, mivel az emberi figyelmi mintázatokat rögzíti és vizualizálja, amelyek alapján következtetéseket vonhatunk le a kód érthetőségéről. A modell a fixációk és a szakkádok mintázatát elemzi, majd ezt színekkel ábrázolja. A modellel történő hőtérkép elemzése alapján az alábbi javaslatokat lehet tenni a kód szerkezetének és érthetőségének javítására:

1. **Beszédesebb változónevek használata:** A jelenlegi változónevek, mint például p, n és q, nem adnak támpontot arra, hogy milyen adatokat tárolnak. Ezért javasolt beszédes neveket használni, például: prices a p helyett; products az n helyett; cart a q helyett. Ez jelentősen növeli a kód olvashatóságát, és csökkenti a résztvevők kognitív terhelését.
2. **Hibakezelési logika egyszerűsítése:** Az index (i) visszaállítása -1-re és a teljes kosár törlése (q.Clear()) nem intuitív. Ehelyett javasolt egy külön metódust használni a hibakezeléshez, amely figyelmezteti a felhasználót, de nem indítja újra a teljes folyamatot.
3. **Feltételkezelés külön metódusba szervezése:** A for cikluson belüli feltételkezelési logika külön metódusba szervezése (HandleCartUpdate) segítene a kód szerkezetének egyszerűsítésében.
4. **Kommentek és dokumentáció:** Rövid, de informatív kommenteket kellene hozzáadni a kódhoz, amelyek magyarázzák a logikát, különösen a bonyolultabb részeknél. Ez segítene a tanulók számára a kód gyorsabb megértésében.
5. **Kód szerkezetének vizuális átláthatósága:** A kódot jobban strukturált blokkokba kellene szervezni, például külön szakaszokban kezelni a bemenetek fogadását, a logikát és a nyugta kiírását. Ez csökkentené a vizuális terhelést és javítaná az olvashatóságot.

5.5. Az MI-Modell visszajelzés hatékonyságának elemzése

Ebben a fejezetben tárgyalásra kerül, hogy a mesterséges intelligencia által támogatott szemmozgásparaméterek bevonása vajon hozzá tudott-e járni a szoftverfejlesztésben a kód átláthatóságának objektív értékeléséhez. Az előző fejezetben kapott visszajelzések alapján elkészített tiszta kódot bemutatom a tanulóknak, és megvizsgálom, hogy az átláthatóságot célzó módosítások valóban elősegítik-e a kód könnyebb értelmezését. A 2. sz. melléklet az elkészített kódot mutatja meg. A résztvevők figyelmi mintázatait szemmozgáskövető eszköz segítségével rögzítem, valamint Likert-skálás kérdőívvel mérem a kód értelmezhetőségéről alkotott

véleményüket. Az így gyűjtött adatokat statisztikai elemzésnek vetem alá, különös tekintettel a T-próbára, hogy megállapítsam, milyen mértékben javult a kód átláthatósága az eredeti verzióhoz képest. A Likert-skála ugyanazokkal a kérdésekkel rendelkezik, ami az eredeti kód értékelésnél volt.

Az új kód elemzése után a hallgatók a 2. táblázat alapján jelölték meg a kérdőíven.

2. táblázat: Új kód értékelése Likert skálán

Személy	1. Kérdés	2. Kérdés	3. Kérdés	4. Kérdés
Tanuló 1.	3	4	4	3
Tanuló 2.	4	4	4	2
Tanuló 3.	4	3	4	3
Tanuló 4.	4	4	4	3
Tanuló 5.	5	5	4	2
Tanuló 6.	4	5	4	2
Tanuló 7.	5	5	4	3
Tanuló 8.	3	4	3	4
Tanuló 9.	4	4	4	3
Tanuló 10.	3	5	4	4
Tanuló 11.	5	5	5	4
Tanuló 12.	4	3	4	2
Tanuló 13.	4	4	4	2
Tanuló 14.	5	5	5	3
Tanuló 15.	5	4	5	3
Átlag:	4,13	4,27	4,13	2,87
Medián:	4	4	4	3
Szórás:	0,74	0,70	0,52	0,74

A táblázat elemzése alapján az új kód értékelése láthatóan javulást mutat a korábbi verzióhoz képest, különösen az érthetőség, logikusság és követhetőség szempontjából. Az első kérdés, amely a kód megértésének nehézségére vonatkozott, 4,13-as átlagértéket kapott. Ez az érték a skála felső tartományába esik, jelezve, hogy a résztvevők többsége könnyen érthetőnek találta a kódot. A medián szintén 4-es, ami tovább erősíti azt a következtetést, hogy a válaszadók jelentős része pozitívan értékelte a kód érthetőségét. A 0,74-es szórás alacsonynak mondható, ami azt mutatja, hogy a vélemények nagyjából egységesek voltak, és nem fordultak elő szélsőséges értékelések. A második kérdés a kód logikusságát vizsgálta, és a legmagasabb átlagértéket, 4,27-et érte el. Ez az eredmény kiemelkedően pozitív, és arra utal, hogy a javasolt tiszta kód szerkezete jól átgondolt és logikus. A medián itt is 4-es, ami azt jelzi, hogy a résztvevők többsége inkább a "logikus" vagy "nagyon logikus" értékelést választotta. A 0,70-

es szórás tovább erősíti, hogy a válaszok viszonylag egységesek voltak, és a diákok egyetértettek abban, hogy a kód logikai felépítése jelentősen javult. A harmadik kérdés, amely a változók és függvények közötti kapcsolatok követhetőségét vizsgálta, szintén erősen pozitív eredményt mutatott, 4,13-as átlagértékkel. Ez azt jelenti, hogy a diákok többsége könnyen követhetőnek találta a tiszta kódot, amit a medián érték (4) is alátámaszt. A szórás itt a legalacsonyabb, 0,52, ami arra utal, hogy ezen a területen mutatkozott a legnagyobb egyetértés a válaszadók között. Ez különösen fontos eredmény, mivel a hőtérkép elemzéséből kiderült, hogy a korábbi kód ezen a ponton volt a legproblémásabb. A negyedik kérdés, amely a kód javításra szorultságát mérte, alacsonyabb átlagot kapott, 2,87-et. Ez azonban pozitív változásként értékelhető, hiszen a résztvevők többsége úgy vélte, hogy a kód kevésbé szorul javításra, mint korábban. A medián itt 3, ami azt jelzi, hogy néhány résztvevő még mindig úgy vélte, hogy van tér a fejlesztésre. A 0,74-es szórás viszonylag egységes véleményeket tükröz, bár itt a válaszok kissé szórtabbak, mint az előző kérdéseknél.

5.5.1. Az MI visszajelzés hatékonyságának elemzése szemmozgás paraméterek kiértékelésével

Ebben a fejezetben a résztvevők szemmozgási paraméterei kerülnek elemzésre, hogy objektíven megállapítható legyen, mennyiben járult hozzá a tiszta kód a kód átláthatóságának növeléséhez. A szemmozgáskövető eszköz által rögzített adatok – például a fixációk hossza, azok eloszlása, valamint a szakkádok mintázatai – a diákok figyelmi mintázatairól nyújtanak értékes információt. Ezek az adatok lehetőséget adnak arra, hogy a korábbi és az új kód között tapasztalt figyelmi különbségeket objektív módon kerüljenek vizsgálatra. A T-próba segítségével statisztikai összehasonlítást végzek a két kódbázishoz tartozó szemmozgásparaméterek között. Ez eredményül adja, hogy van-e szignifikáns eltérés a diákok kódolvasási mintázatai között, és hogy ez az eltérés alátámasztja-e a tiszta kód érthetőségének és követhetőségének javulását. Ez a megközelítés lehetővé teszi, hogy a kód értékeléséhez ne csupán szubjektív válaszokat használjunk, hanem a figyelmi mintázatok kvantitatív elemzésével objektív bizonyítékokat szerezzünk a fejlesztések hatékonyságáról.

Fixációs szám hipotézis kiértékelése eredeti és tiszta kód között

Ebben az alfejezetben az eredeti és a tiszta kód fixációs számainak statisztikai összehasonlítása kerül elvégzésre T-próba segítségével. A fixációk száma a kód olvasása közben mutatott figyelmi mintázatok egyik legfontosabb paramétere, mivel azt jelzi, hogy a diákok hányszor álltak meg egy adott ponton az információ feldolgozása érdekében. A magasabb fixációs szám

a kód érthetőségével kapcsolatos problémákra utalhat, míg az alacsonyabb szám könnyebb feldolgozhatóságot és átláthatóságot jelez. A statisztikai kiértékeléshez független mintás T-próbát alkalmaztam, mivel a két kód bázis fixációs számai két különböző mérőszorozatból származnak. A T-próba segítségével megvizsgálom, hogy van-e szignifikáns különbség a két csoport fixációs száma között. A nullhipotézis az, hogy nincs különbség a két kód fixációs számaiban, míg az alternatív hipotézis az, hogy a tiszta kód esetében kevesebb fixációra van szükség a kód olvasása során.

Az adatok alapján az alábbi értékek rendelkeznek két kód fixációs számaira vonatkozóan, amelyet 3. táblázat részletez.

3. táblázat: Rendelkezésre álló fixációs szám adatok:

	Csoport	N	Átlag	Szórás
Fixációs szám	1	15	667,933	77,04
	2	15	182,6	41,032

A T-próba eredményei alapján a fixációk száma között szignifikáns különbség mutatható ki az eredeti és a tiszta kód olvasása során. A Levene-teszt p-értéke kisebb, mint a szignifikanciaszint (0,05), így az "egyenlő szórások nem feltételezett" modell eredményeit alapján a résztvevők fixációs számai szignifikánsan alacsonyabb volt ($182,6 \pm 41,03$) mint az eredeti kódnál ($667,93 \pm 77,04$). $t(21) = 21,535$, p (két oldali) $< 0,001$.

Hipotézis kiértékelése átlagos fixációs időtartam alapján eredeti és tiszta kód között

Ebben az elfejezetben az átlagos fixációs időtartamok statisztikai összehasonlítását végzem el a az eredeti és a tiszta kód esetében. Az átlagos fixációs időtartam a szemmozgás során eltöltött idő egy adott ponton, amely a kognitív feldolgozás intenzitását jelzi. Hosszabb fixációs idő általában a nehezen érthető, összetett vagy kognitív erőfeszítést igénylő kódrészekre utal, míg rövidebb fixációs idő az átláthatóbb és könnyebben értelmezhető kódot tükrözi. A statisztikai kiértékeléshez független mintás T-próbát alkalmaztam, mivel a két kód bázis átlagos fixációs időtartama két különböző mérőszorozatból származik. A T-próba segítségével megvizsgálom, hogy van-e szignifikáns különbség a két csoport átlagos fixációs időtartama között. A nullhipotézis az, hogy nincs különbség a két kód átlagos fixációs időtartamaiban, míg az alternatív hipotézis az, hogy a tiszta kód esetében rövidebb időtartamok figyelhetők meg a kód olvasása során.

Az adatok alapján az alábbi értékek rendelkeznek két kód átlagos fixáció időtartamra vonatkozóan, amelyet 4. táblázat részletez.

4. táblázat: Rendelkezésre álló átlagos fixációs időtartamra vonatkozó elérhető adatok

	Csoport	N	Átlag	Szórás
Átlagos Fixációs időtartam	1	15	280,93	26,98
	2	15	204,34	10,69

A T-próba eredményei alapján a fixációk száma között szignifikáns különbség mutatható ki az eredeti és a tiszta kód olvasása során. A Levene-teszt p-értéke kisebb, mint a szignifikanciaszint (0,05), így az "egyenlő szórások nem feltételezett" modell eredményeit alapján a résztvevők fixációs számai szignifikánsan alacsonyabb volt ($204,34 \pm 10,69$) mint az eredeti kódnál ($280,93 \pm 26,98$). $t(28) = 10,220$, p (két oldali) $< 0,001$.

5.6. Következtetések

A Likert-skálás kérdőívek és a két T-próba eredményei alapján világosan megállapítható, hogy a tiszta kód jelentős mértékben javította a kód átláthatóságát és érthetőségét az eredeti kódhoz képest. Az elemzések objektív statisztikai bizonyítékot nyújtanak arra, hogy a kódolvasási folyamat hatékonysága növekedett, amit mind a szubjektív válaszok, mind a szemmozgásadatok alátámasztanak.

A Likert-skálás értékelések alapján a tiszta kód minden szempontból kedvezőbb visszajelzéseket kapott. A diákok szerint a kód logikussága, érthetősége és követhetősége jelentősen javult, amit a magasabb átlagértékek és az alacsonyabb szórások is tükröznek. Ezek az eredmények egyértelműen jelzik, hogy a kód struktúrájának és változóneveinek átdolgozása megkönnyítette a kód értelmezését.

A T-próbák további kvantitatív bizonyítékokkal szolgáltak a kód minőségi javulására. Az első T-próba, amely a fixációk számát hasonlította össze, kimutatta, hogy a tiszta kód olvasása során szignifikánsan kevesebb fixációra volt szükség. Ez arra utal, hogy a kód logikai szerkezete jobban támogatta a gyors és hatékony információfeldolgozást. A második T-próba, amely az átlagos fixációs időtartamokat vizsgálta, szintén szignifikáns csökkenést mutatott a tiszta kód esetében. Ez a rövidebb fixációs idők a kód könnyebb kognitív feldolgozhatóságát és átláthatóságát tükrözik. Az eredmények összességében azt igazolják, hogy a tiszta kód nemcsak a szubjektív vélemények szintjén, hanem objektív szemmozgás-paraméterek alapján is jelentős javulást mutatott az eredeti kódhoz képest. Ez megerősíti, hogy a mesterséges intelligencia és

szemmozgás alapú elemzés hatékony eszközként használható a szoftverfejlesztési gyakorlatok optimalizálásában. A kód átláthatóságának növelése nemcsak a tanulók számára könnyítette meg a kód olvasását és értelmezését, hanem hozzájárult a fejlesztési folyamatok átláthatóságának és hatékonyságának javításához is. A tiszta kód alkalmazása nemcsak oktatási, hanem ipari környezetben is releváns következtetéseket kínál, mivel az átláthatóbb és logikusabb kódstruktúrák csökkenthetik a fejlesztési hibák számát, növelhetik a karbantarthatóságot, és javíthatják a fejlesztői csapatok közötti kommunikációt. A szemmozgásparaméterek elemzése tehát nemcsak az emberi viselkedés tanulmányozásában, hanem a szoftverminőség biztosításában is hasznos eszköz lehet, amit a jelen kutatás eredményei is alátámasztanak.

6. Kutatás korlátai

A kutatás során számos tényező korlátozta a vizsgálatok kiterjedtségét és az eredmények általánosíthatóságát. Az egyik fő kihívás az eye-tracking és a mesterséges intelligencia modulok integrálásához szükséges technológiai infrastruktúra biztosítása volt. Bár a választott eszközök és szoftverek megfelelőek voltak a kitűzött célok eléréséhez, a piacon elérhető fejlettebb technológiák alkalmazása szélesebb körű elemzési lehetőségeket nyújthatott volna. Az eszközhasználat költségvonzatai is jelentősen befolyásolták a kutatás kereteit, így bizonyos technológiai megoldások alkalmazása nem volt lehetséges. További korlátot jelentett a vizsgálati mintanagyság, amely elsősorban a résztvevők számában mutatkozott meg. A kutatás során begyűjtött adatok szignifikáns eredményekkel szolgáltak ugyan, azonban a minta nagyságának növelésével pontosabb és általánosíthatóbb következtetések vonhatók le. Ezenkívül a résztvevők egyetemi hallgatók voltak, így a vizsgálatok kizárólag az ő demográfiai csoportjukra vonatkoztak, ami korlátozza az eredmények más csoportokra történő kiterjeszthetőségét. Kutatás során felhasznált mesterséges intelligencia modellek és algoritmusok szintén behatárolták az elemzések mélységét. Az alkalmazott modellek pontosak és megbízhatóak voltak, de további fejlesztések, például fejlettebb gépi tanulási technikák alkalmazása, részletesebb és prediktívebb elemzési lehetőségeket kínálhatott volna. Végül az időbeli korlátok is meghatározó szerepet játszottak a kutatás során.

A rendelkezésre álló idő nem tette lehetővé a hosszabb távú vizsgálatok és az iteratív fejlesztési ciklusok teljes körű alkalmazását, így bizonyos elemzések és optimalizációk a jövőbeli kutatások tárgyát képezhetik. Ennek ellenére a kutatás célkitűzéseit sikerült teljesíteni, és az eredmények értékes alapot nyújtanak a további fejlesztésekhez.

7. Összefoglalás

A dolgozat fő célja az agilis szoftverfejlesztés támogatásának elősegítése mesterséges intelligencia és szemmozgás-követés technológiák integrációjával. Az agilis módszertan már régóta népszerű a szoftverfejlesztési iparágban, mivel gyors, rugalmas és képes alkalmazkodni a változó piaci igényekhez. A módszertan egyik nagy előnye, hogy a fejlesztők gyakran kisebb iterációkban dolgoznak, és a felhasználói visszajelzések alapján folyamatosan javítják a terméket. Ugyanakkor az agilis fejlesztés során gyakran szembesülnek azzal a kihívással, hogy miként lehet a felhasználói élményt hatékonyan optimalizálni és hogyan lehet hatékony kommunikációt fenntartani a csapatok között.

A dolgozat részletesen bemutatja, hogyan segítheti az MI és az eye-tracking technológiák kombinációja a fejlesztési folyamatok hatékonyságát. A mesterséges intelligencia képes nagy mennyiségű adatot feldolgozni, automatikus hibakeresést végezni és segíteni a döntéshozatali folyamatokban. Az MI feladata az, hogy azonosítsa a kódban lévő problémákat, javaslatokat tegyen azok megoldására, és optimalizálja a fejlesztés lépéseit. Ezáltal csökkenthető a hibák száma, rövidülhet a fejlesztési idő, és növelhető a kód minősége. Az eye-tracking technológia lehetőséget biztosít arra, hogy nyomon kövessék a felhasználók szemmozgását, így pontos képet kaphatunk arról, hogy a felhasználók hogyan használják a fejlesztett terméket. A szemmozgás-követés lehetővé teszi, hogy azonosítsuk azokat a felhasználói interakciókat, amelyek nehézségeket okoznak, vagy ahol a felhasználók hosszabb időt töltenek, mint szükséges. Ezáltal javítható a felhasználói felületek tervezése és a felhasználói élmény. Az eye-tracking technológia különösen hasznos lehet olyan helyzetekben, amikor a vizuális interfész komplex, és fontos tudni, hogy a felhasználók hol veszítenek el navigáció közben.

A kutatás során bemutatott integrált rendszer arra törekszik, hogy mind az MI, mind az eye-tracking adatok kombinációjával olyan komplex rendszert hozzon létre, amely valós idejű információkat biztosít a fejlesztők számára. Az MI képes arra, hogy előrejelzéseket adjon a projektek kimeneteléről, valamint javaslatokat tegyen a fejlesztési folyamatok finomhangolására. Emellett az eye-tracking technológia segítségével a felhasználók valós idejű visszajelzéseket adhatnak a termék használatáról, amelyeket a fejlesztők azonnal beépíthetnek a munkafolyamatukba. Ez az integrált megközelítés elősegíti a gyors és hatékony visszacsatolást, valamint a felhasználói élmény központi szerepét a fejlesztés során. A dolgozat végső célja egy olyan elméleti modell bemutatása, amely az MI és az eye-tracking technológiák integrációján alapuló fejlesztési keretrendszert kínál. Ez a keretrendszer nem csak a fejlesztők

számára biztosít automatikus visszajelzést, hanem a felhasználói élményt is középpontba helyezi, ezáltal növelve a szoftverek minőségét és a fejlesztési folyamat hatékonyságát. A mesterséges intelligencia és az eye-tracking technológia kombinációja új lehetőségeket nyit a szoftverfejlesztés területén, és segít abban, hogy a fejlesztők gyorsabban és hatékonyabban dolgozzanak a felhasználói igények kielégítésén. Az integrált rendszer bevezetése új távlatokat nyit meg a fejlesztés során, és biztosítja, hogy a felhasználói visszajelzések közvetlenül befolyásolják a termékfejlesztési ciklusokat.

Az dolgozat második felében az elemzéseim célja az volt, hogy objektív adatokat gyűjtssek a kódolvasási mintázatokról, valamint a kód érthetőségének és követhetőségének javításáról a mesterséges intelligencia támogatásával. Ehhez először rögzítettem az eredeti kód és a tiszta kód olvasása közben keletkezett fixációs számokat és átlagos fixációs időtartamokat. A statisztikai elemzéshez T-próbát használtam, hogy megvizsgáljam a két csoport között mutatkozó különbségeket. Az eredmények azt mutatták, hogy a tiszta kód esetében mind a fixációs számok, mind az átlagos fixációs idők szignifikánsan alacsonyabbak voltak, ami egyértelműen jelezte a kód jobb érthetőségét és könnyebb feldolgozhatóságát. A kapott adatok alapján sikerült bizonyítanom, hogy a mesterséges intelligencia és a szemmozgás-követési technológiák integrálása hatékonyan hozzájárulhat a kódok minőségének javításához, és pontosabb visszajelzést nyújthat a fejlesztési folyamatok optimalizálásához.

8. Irodalomjegyzék

- [1] E. Altameem, „Impact of Agile Methodology on Software Development”, *Comput. Inf. Sci.*, köt. 8, sz. 2, o. p9, ápr. 2015, doi: 10.5539/cis.v8n2p9.
- [2] K. Kaur, M. Khurana, és Manisha, „Impact of Agile Scrum Methodology on Time to Market and Code Quality – A Case Study”, *2021 3rd Int. Conf. Adv. Comput. Commun. Control Netw. ICAC3N*, o. 1673–1678, 2021, doi: 10.1109/ICAC3N53548.2021.9725375.
- [3] S. C. Misra, V. Kumar, és U. Kumar, „Identifying some important success factors in adopting agile software development practices”, *J. Syst. Softw.*, köt. 82, sz. 11, o. 1869–1890, nov. 2009, doi: 10.1016/j.jss.2009.05.052.
- [4] M. Stadler, R. Vallon, M. Pazderka, és T. Grechenig, „AGILE DISTRIBUTED SOFTWARE DEVELOPMENT IN NINE CENTRAL EUROPEAN TEAMS: CHALLENGES, BENEFITS AND RECOMMENDATIONS”, *Int. J. Comput. Sci. Inf. Technol.*, köt. 11, sz. 01, o. 01–18, febr. 2019, doi: 10.5121/ijcsit.2019.11101.
- [5] K. Kuusinen, T. Mikkonen, és S. Pakarinen, „Agile User Experience Development in a Large Software Organization: Good Expertise but Limited Impact”, o. 94–111, 2012, doi: 10.1007/978-3-642-34347-6_6.
- [6] P. T. Robinson, „Communication Network in an Agile Distributed Software Development Team”, in *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, Montreal, QC, Canada: IEEE, máj. 2019, o. 100–104. doi: 10.1109/ICGSE.2019.00015.
- [7] K. Kuusinen, „Overcoming Challenges in Agile User Experience Work: Cross-Case Analysis of Two Large Software Organizations”, *2015 41st Euromicro Conf. Softw. Eng. Adv. Appl.*, o. 454–458, 2015, doi: 10.1109/SEAA.2015.38.
- [8] Q. Sun, G. Chen, és S. He, „Enterprise Development Decision Support System Based on Artificial Intelligence Technology”, *2022 Int. Conf. Artif. Intell. Things Crowdsensing AIoTCs*, o. 245–249, 2022, doi: 10.1109/AIoTCs58181.2022.00044.
- [9] D. J. Pedregal, „New algorithms for automatic modelling and forecasting of decision support systems”, *Decis Support Syst*, köt. 148, 2021, doi: 10.1016/J.DSS.2021.113585.

- [10] M. Mesir, „The Role of Artificial Intelligence in Decision Making in Small Businesses”, *Indian J. Sci. Technol.*, 2019, doi: 10.17485/ijst/2019/v12i18/144586.
- [11] H. Al Maqbali, F. Scholer, J. A. Thom, és M. Wu, „Using eye tracking for evaluating web search interfaces”, in *Proceedings of the 18th Australasian Document Computing Symposium*, Brisbane Queensland Australia: ACM, dec. 2013, o. 2–9. doi: 10.1145/2537734.2537747.
- [12] N. H. Mat Zain, F. H. Abdul Razak, A. Jaafar, és M. F. Zulkipli, „Eye Tracking in Educational Games Environment: Evaluating User Interface Design through Eye Tracking Patterns”, in *Visual Informatics: Sustaining Research and Innovations*, köt. 7067, H. B. Zaman, P. Robinson, M. Petrou, P. Olivier, T. K. Shih, S. Velastin, és I. Nyström, Szerk., in *Lecture Notes in Computer Science*, vol. 7067. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, o. 64–73. doi: 10.1007/978-3-642-25200-6_7.
- [13] M. Raschke, T. Blascheck, és M. Burch, „Visual Analysis of Eye Tracking Data”, o. 391–409, 2014, doi: 10.1007/978-1-4614-7485-2_15.
- [14] B. Sharif, M. Falcone, és J. I. Maletic, „An eye-tracking study on the role of scan time in finding source code defects”, in *Proceedings of the Symposium on Eye Tracking Research and Applications*, in ETRA '12. New York, NY, USA: Association for Computing Machinery, 0 2012, o. 381–384. doi: 10.1145/2168556.2168642.
- [15] K. R. Chandrika, J. Amudha, és S. D. Sudarsan, „Recognizing eye tracking traits for source code review”, in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, szept. 2017, o. 1–8. doi: 10.1109/ETFA.2017.8247637.
- [16] C. Zhang, J. Chi, Z. Zhang, X. Gao, T. Hu, és Z. Wang, „Gaze estimation in a gaze tracking system”, *Sci. China Inf. Sci.*, köt. 54, o. 2295–2306, 2011, doi: 10.1007/s11432-011-4243-6.
- [17] E. D. Guestrin és M. Eizenman, „General theory of remote gaze estimation using the pupil center and corneal reflections”, *IEEE Trans. Biomed. Eng.*, köt. 53, sz. 6, o. 1124–1133, jún. 2006, doi: 10.1109/TBME.2005.863952.
- [18] C. H. Morimoto, D. Koons, A. Amir, és M. Flickner, „Pupil detection and tracking using multiple light sources”, *Image Vis. Comput.*, köt. 18, sz. 4, o. 331–335, márc. 2000, doi: 10.1016/S0262-8856(99)00053-0.

- [19] Y. Ebisawa és S. -i. Satoh, „Effectiveness of pupil area detection technique using two light sources and image difference method”, in *Proceedings of the 15th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Diego, CA: IEEE, 1993, o. 1268–1269. doi: 10.1109/IEMBS.1993.979129.
- [20] Q. Ji és Z. Zhu, „Eye and gaze tracking for interactive graphic display”, in *Proceedings of the 2nd international symposium on Smart graphics*, Hawthorne New York USA: ACM, jún. 2002, o. 79–85. doi: 10.1145/569005.569017.
- [21] M. Mesir és Kuwait Foundation for the Advancements in Science (KFAS), Kuwait;, „The Role of Artificial Intelligence in Decision Making in Small Businesses”, *Indian J. Sci. Technol.*, köt. 12, sz. 18, o. 1–6, máj. 2019, doi: 10.17485/ijst/2019/v12i18/144586.
- [22] D. J. Pedregal, „New algorithms for automatic modelling and forecasting of decision support systems”, *Decis. Support Syst.*, köt. 148, o. 113585, szept. 2021, doi: 10.1016/j.dss.2021.113585.
- [23] V. Kaul, S. Enslin, és S. A. Gross, „History of artificial intelligence in medicine”, *Gastrointest. Endosc.*, köt. 92, sz. 4, o. 807–812, okt. 2020, doi: 10.1016/j.gie.2020.06.040.
- [24] S. E. Dreyfus, „Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure”, *J. Guid. Control Dyn.*, köt. 13, sz. 5, o. 926–928, szept. 1990, doi: 10.2514/3.25422.
- [25] H. Jaakkola, J. Henno, J. Makela, és B. Thalheim, „Artificial Intelligence Yesterday, Today and Tomorrow”, in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia: IEEE, máj. 2019, o. 860–867. doi: 10.23919/MIPRO.2019.8756913.
- [26] L. Pinto-Coelho, „How Artificial Intelligence Is Shaping Medical Imaging Technology: A Survey of Innovations and Applications”, *Bioengineering*, köt. 10, sz. 12, o. 1435, dec. 2023, doi: 10.3390/bioengineering10121435.
- [27] A. Al Kuwaiti és mtsai., „A Review of the Role of Artificial Intelligence in Healthcare”, *J. Pers. Med.*, köt. 13, sz. 6, o. 951, jún. 2023, doi: 10.3390/jpm13060951.
- [28] V. Mehta, „Artificial Intelligence in Medicine: Revolutionizing Healthcare for Improved Patient Outcomes”, *J. Med. Res. Innov.*, köt. 7, sz. 2, o. e000292, jún. 2023, doi: 10.32892/jmri.292.

- [29] Rita Jain, „Role of artificial intelligence in banking and finance”, *J. Manag. Sci.*, köt. 13, sz. 3, o. 1–4, szept. 2023, doi: 10.26524/jms.13.27.
- [30] P. Li, „Application of Artificial Intelligence Technology in Internet Finance and Analysis of Security Risks”, in *2023 IEEE International Conference on Integrated Circuits and Communication Systems (ICICACS)*, Raichur, India: IEEE, febr. 2023, o. 1–6. doi: 10.1109/ICICACS57338.2023.10099863.
- [31] K. Muhammad, A. Ullah, J. Lloret, J. D. Ser, és V. H. C. De Albuquerque, „Deep Learning for Safe Autonomous Driving: Current Challenges and Future Directions”, *IEEE Trans. Intell. Transp. Syst.*, köt. 22, sz. 7, o. 4316–4336, júl. 2021, doi: 10.1109/TITS.2020.3032227.
- [32] Y. Ma, Z. Wang, H. Yang, és L. Yang, „Artificial intelligence applications in the development of autonomous vehicles: a survey”, *IEEECAA J. Autom. Sin.*, köt. 7, sz. 2, o. 315–329, márc. 2020, doi: 10.1109/JAS.2020.1003021.
- [33] C.-F. Chien, S. Dauzère-Pérès, W. T. Huh, Y. J. Jang, és J. R. Morrison, „Artificial intelligence in manufacturing and logistics systems: algorithms, applications, and case studies”, *Int. J. Prod. Res.*, köt. 58, sz. 9, o. 2730–2731, máj. 2020, doi: 10.1080/00207543.2020.1752488.
- [34] D. Loske és M. Klumpp, „Human-AI collaboration in route planning: An empirical efficiency-based analysis in retail logistics”, *Int. J. Prod. Econ.*, köt. 241, o. 108236, nov. 2021, doi: 10.1016/j.ijpe.2021.108236.
- [35] N. D. Q. Bui, Y. Wang, és S. Hoi, „Detect-Localize-Repair: A Unified Framework for Learning to Debug with CodeT5”, 2022, *arXiv*. doi: 10.48550/ARXIV.2211.14875.
- [36] Y. Bajaj és M. K. Samal, „Accelerating Software Quality: Unleashing the Power of Generative AI for Automated Test-Case Generation and Bug Identification”, *Int. J. Res. Appl. Sci. Eng. Technol.*, köt. 11, sz. 7, o. 345–350, júl. 2023, doi: 10.22214/ijraset.2023.54628.
- [37] A. Rasheed és mtsai., „Requirement Engineering Challenges in Agile Software Development”, *Math. Probl. Eng.*, köt. 2021, o. 1–18, máj. 2021, doi: 10.1155/2021/6696695.
- [38] K. Kuusinen, T. Mikkonen, és S. Pakarinen, „Agile User Experience Development in a Large Software Organization: Good Expertise but Limited Impact”, in *Human-Centered Software Engineering*, köt. 7623, M. Winckler, P. Forbrig, és R. Bernhaupt, Szerk., in Lecture

Notes in Computer Science, vol. 7623. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, o. 94–111. doi: 10.1007/978-3-642-34347-6_6.

[39] J. Pasuksmit és mtsai., „Improving Agile Planning for Reliable Software Delivery”, in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, Melbourne, Australia: IEEE, máj. 2023, o. 25–26. doi: 10.1109/MSR59073.2023.00017.

[40] M. L. Bernardi, G. Canfora, G. A. Di Lucca, M. Di Penta, és D. Distanté, „The relation between developers’ communication and fix-Inducing changes: An empirical study”, *J. Syst. Softw.*, köt. 140, o. 111–125, jún. 2018, doi: 10.1016/j.jss.2018.02.065.

[41] R. Vieira, A. Da Silva, L. Rocha, és J. P. Gomes, „From Reports to Bug-Fix Commits: A 10 Years Dataset of Bug-Fixing Activity from 55 Apache’s Open Source Projects”, in *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*, Recife Brazil: ACM, szept. 2019, o. 80–89. doi: 10.1145/3345629.3345639.

[42] P. Behnamghader, R. Alfayez, K. Srisopha, és B. Boehm, „Towards Better Understanding of Software Quality Evolution through Commit-Impact Analysis”, in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Prague, Czech Republic: IEEE, júl. 2017, o. 251–262. doi: 10.1109/QRS.2017.36.

[43] C. A. Thompson, „Investigating completeness and consistency of links between issues and commits”, 2017, doi: 10.14288/1.0354567.

[44] Y. Huang, N. Jia, H.-J. Zhou, X.-P. Chen, Z.-B. Zheng, és M.-D. Tang, „Learning Human-Written Commit Messages to Document Code Changes”, *J. Comput. Sci. Technol.*, köt. 35, sz. 6, o. 1258–1277, nov. 2020, doi: 10.1007/s11390-020-0496-0.

[45] Hong Men, Yujie Wu, Yanchun Gao, Xiaoying Li, és Shanrang Yang, „Application of support vector machine to pattern classification”, in *2008 9th International Conference on Signal Processing*, Beijing, China: IEEE, okt. 2008, o. 1612–1615. doi: 10.1109/ICOSP.2008.4697444.

[46] J. A. Gualtieri és R. F. Crompt, „Support vector machines for hyperspectral remote sensing classification”, előadás The 27th AIPR Workshop: Advances in Computer-Assisted Recognition, R. J. Mericsko, Szerk., Washington, DC, jan. 1999, o. 221–232. doi: 10.1117/12.339824.

- [47] A. Acien, A. Morales, J. Fierrez, R. Vera-Rodriguez, és J. Hernandez-Ortega, „Active detection of age groups based on touch interaction”, *IET Biom.*, köt. 8, sz. 1, o. 101–108, jan. 2019, doi: 10.1049/iet-bmt.2018.5003.
- [48] R. Bednarik, S. Eivazi, és H. Vrzakova, „A Computational Approach for Prediction of Problem-Solving Behavior Using Support Vector Machines and Eye-Tracking Data”, in *Eye Gaze in Intelligent User Interfaces*, Y. I. Nakano, C. Conati, és T. Bader, Szerk., London: Springer London, 2013, o. 111–134. doi: 10.1007/978-1-4471-4784-8_7.
- [49] M. Ahmed, R. Seraj, és S. M. S. Islam, „The k-means Algorithm: A Comprehensive Survey and Performance Evaluation”, *Electronics*, köt. 9, sz. 8, o. 1295, aug. 2020, doi: 10.3390/electronics9081295.
- [50] K. Rajeswari, O. Acharya, M. Sharma, M. Kopnar, és K. Karandikar, „Improvement in k-Means Clustering Algorithm Using Data Clustering”, in *2015 International Conference on Computing Communication Control and Automation*, Pune, India: IEEE, febr. 2015, o. 367–369. doi: 10.1109/ICCUBEA.2015.205.
- [51] Y. Zhao és X. Zhou, „K-means Clustering Algorithm and Its Improvement Research”, *J. Phys. Conf. Ser.*, köt. 1873, sz. 1, o. 012074, ápr. 2021, doi: 10.1088/1742-6596/1873/1/012074.
- [52] M. Breskvar, D. Kocev, és S. Džeroski, „Ensembles for multi-target regression with random output selections”, *Mach. Learn.*, köt. 107, sz. 11, o. 1673–1709, nov. 2018, doi: 10.1007/s10994-018-5744-y.
- [53] J. L. Speiser, M. E. Miller, J. Tooze, és E. Ip, „A comparison of random forest variable selection methods for classification prediction modeling”, *Expert Syst. Appl.*, köt. 134, o. 93–101, nov. 2019, doi: 10.1016/j.eswa.2019.05.028.
- [54] K. Dedja, F. K. Nakano, K. Pliakos, és C. Vens, „BELLATREX: Building Explanations through a LocaLly AccuraTe Rule EXtractor”, 2022, *arXiv*. doi: 10.48550/ARXIV.2203.15511.
- [55] T. Lee, J. B. Lee, és H. P. In, „A Study of Different Coding Styles Affecting Code Readability”, *Int. J. Softw. Eng. Its Appl.*, köt. 7, sz. 5, o. 413–422, szept. 2013, doi: 10.14257/ijseia.2013.7.5.36.

9. Mellékletek

1. számú melléklet

Új, MI javaslatai által refaktorált kód.

```
using System;
using System.Collections.Generic;

namespace ShopSystem
{
    class Program
    {
        static void Main(string[] args)
        {
            // Products and Their Prices
            List<Product> products = new List<Product>
            {
                new Product("Apple", 9.99),
                new Product("Banana", 19.49),
                new Product("Milk", 5.75),
                new Product("Bread", 3.20)
            };

            ShoppingCart cart = new ShoppingCart();

            Console.WriteLine("Welcome to the shop! Add items to your cart.\n");

            // Adding items to the cart
            foreach (var product in products)
            {
                Console.WriteLine($"Add {product.Name} to cart for
                ${product.Price}? (y/n)");
                string input = Console.ReadLine()?.ToLower();

                if (input == "y")
                    cart.AddProduct(product);
                else if (input == "n")
                    continue;
                else
                    Console.WriteLine("Invalid input. Please enter 'y' or 'n'.");
            }

            // After adding all items, print the receipt
            cart.PrintReceipt();
        }
    }

    public class Product
    {
        public string Name { get; }
        public double Price { get; }

        public Product(string name, double price)
        {
            Name = name;
            Price = price;
        }
    }
}
```

```
public class ShoppingCart
{
    private readonly Dictionary<string, CartItem> _items;
    private double _totalPrice;

    public ShoppingCart()
    {
        _items = new Dictionary<string, CartItem>();
        _totalPrice = 0;
    }

    public void AddProduct(Product product)
    {
        if (_items.ContainsKey(product.Name))
        {
            _items[product.Name].Quantity++;
        }
        else
        {
            _items[product.Name] = new CartItem(product);
        }

        _totalPrice += product.Price;
    }

    public void PrintReceipt()
    {
        Console.WriteLine("\nReceipt:");

        foreach (var item in _items.Values)
        {
            double itemTotal = item.Product.Price * item.Quantity;
            Console.WriteLine($"{item.Product.Name}: {item.Quantity} x
            ${item.Product.Price} = ${itemTotal:F2}");
        }

        Console.WriteLine($"Total: ${_totalPrice:F2}");
    }
}

public class CartItem
{
    public Product Product { get; }
    public int Quantity { get; set; }

    public CartItem(Product product)
    {
        Product = product;
        Quantity = 1;
    }
}
```

2. számú melléklet

Gazepoint 3 szemmozgás követő eszköz.

Forrás: <https://www.gazept.com/product/gazepoint-gp3-eye-tracker/>



3. számú melléklet

Smart Eye's Aurora szemmozgás követő eszköz.

Forrás: <https://www.smarteye.se/aurora/setup/>



4. számú melléklet

Tobii Eye X szemmozgás követő eszköz.

Forrás: <https://eyecomtec.com/3118-Tobii-EyeX>



5. számú melléklet

The Eye Tribe szemmozgás követő eszköz.

Forrás: <https://theeyetribe.com/>



6. számú melléklet

EyeLink 1000 Plus szemmozgás követő eszköz.

Forrás: <https://www.sr-research.com/eyelink-1000-plus/>

