

TDK-dolgozat

Szerző:

Varga József Márk

Debreceni Egyetem

Informatikai kar, Információ Technológia Tanszék

Optikai lemez vizuális reprezentációján alapuló adatminőség meghatározása

Témavezetők:

Dr. Jeszenszky Péter

Adjunktus

Dr. Ispány Márton

Egyetemi tanár

Készítette:

Varga József Márk

Programtervező informatikus MSc

1. évfolyam

Tartalomjegyzék

1. Bevezetés	3
2. Felhasznált szoftverek	6
2.1. Integrált fejlesztői környezet	6
2.2. Anaconda	6
2.3. Python csomagok	6
2.4. QPxTool	8
2.5. További szoftverek	8
3. Hardver	9
4. Optikai lemezek adatminősége	10
4.1. Adatminőségi metrikák	10
4.2. Az adathalmaz megalkotása	11
4.2.1. Adatok begyűjtése	11
4.2.2. Adathalmaz előkészítése	12
4.2.3. Előzetes klaszterezés	14
4.3. Eredmények	15
5. A lemezek vizuális reprezentációja	18
5.1. Eszközök és környezet leírása	18
5.2. Lemez fotózás	19
5.2.1. Karcolások kinyerése	20
6. Augmentálás	24
6.1. Képi adatok augmentálása	24
6.2. Adatminőségi halmaz augmentálása	26
6.3. Augmentált adatok asszociálása és új adathalmaz megalkotása	28
6.4. A végső klaszterek megállapítása	30
7. Osztályozás	32
7.1. A modell felépítése és szerkezete	32

7.2. Teljesítmény	34
7.3. Példák	34
8. Összefoglalás	37
8.1. Adatminőségen alapuló következtetések	37
8.2. Vizualizáción alapuló következtetések	38
8.3. Eredmények értékelése	40
8.4. Továbbfejlesztési lehetőségek	40
Irodalomjegyzék	41
Köszönetnyilvánítások	42

1. Bevezetés

Az elmúlt időszakban volt szerencsém részt venni több olyan egyetemi kurzuson, melyek témái a gépi tanulás és a mélytanulás voltak. Ezek adták számomra a motivációt és az érdeklődést a gépi tanulás és az adattudomány felé.

Emellett középiskolás korom óta játszom konzolos játékokkal, amik optikai lemezeken vannak tárolva. Mivel újonnan túlságosan drágák voltak ezek a lemezek, ezért a használt piac felé vettem az irányt. Ezzel viszont az volt a gond, hogy nagy volt a rizikója annak, hogy egy használt lemez a karcolások miatt használhatatlan. Ez adta az ötletet, hogy készítssek egy olyan szoftvert, mely gépi tanuló modellek felhasználásával képes felismerni a rossz lemezeket.

A fő probléma az optikai lemezekkel az, hogy hamar tönkre tudnak menni. A dolgozatban bemutatásra kerülő lemez-szűrő algoritmusommal meg lehet állapítani, hogy egy lemez használható vagy használhatatlan, mindehhez elegendő néhány fénykép a lemez alsó oldaláról.

Egy ilyen szoftvernek számos különböző potenciális célközönsége van. Egyrészt hasznos lehet optikai lemezek adásvételével foglalkozó felhasználók (akár magánszemélyek, akár kereskedők) számára. Másrészt haszonnal alkalmazhatják olyan intézmények is, melyek napi rendszerességgel használnak optikai lemezeket, mint például a könyvtárak és egészségügyi intézmények (kórházak, laborok).

Meg kell jegyezni, hogy a dolgozatban bemutatásra kerülő munkához hasonló eredményeket – melyekben a lemezek felületéről készült fényképeket vizsgálnák – nem találtam a tudományos szakirodalomban.

A fentieket összefoglalva, a kutatásomban egy olyan modell kerül bemutatásra, mely az optikai lemezeket abból a szempontból képes osztályozni, hogy használhatóak-e vagy sem. Ezen gépi tanuló modell képes közel 98% pontossággal megállapítani, hogy az adott lemez jó-e vagy rossz, amihez elegendő a lemez felületi sérüléseiről egy átfogó képsorozat. Sikerült elérnem továbbá azt is, hogy a modell jól teljesítsen korábban nem látott, új adatokon is.

A kutatáshoz a Debreceni Egyetem Egyetemi és Nemzeti Könyvtára biztosított számomra vizsgálható DVD lemezeket, a munkát ezek lefotózásával kezdtem. Mivel

rengeteg szempontnak kellett megfelelnie a képeknek, egy általam egyedileg elkészített szerkezetet/környezetet használtam, ami alkalmas volt megfelelő képek készítéséhez. A lemezeket egy forgó platformra helyeztem, amire egy előre meghatározott szögben bedöntött kamera és fényforrás nézett. Ennek köszönhetően a lemezek felületéről készült fotókon jól kivehetően látszódnak a karcolások.

A képeken fellelhető karcolások (karctérképek) kinyerésére a HED¹ élkinyerő konvolúciós neurálist hálót használtam, mivel erre a célra ez bizonyult az optimálisnak.

A karctérképek kinyerése után a lemezek adatminőségét nyertem ki, mégpedig a QPx-Tool nevű nyílt forrású szoftverrel. Ez lehetővé tette különböző, a lemez adatminőségére vonatkozó leíró statisztikák kinyerését, mint például a *Parity Inner Error* (PIE).

Ezen lépések elvégzése után az adathalmazom a karctérképekből és az adatminőségre vonatkozó metrikák értékeiből állt.

Ahhoz, hogy egy gépi tanuló modell optimálisan teljesítsen, elengedhetetlen, hogy az adathalmaz megfelelő minőségű legyen, illetve hogy elegendően sok adat álljon rendelkezésre. Esetemben csak 100 lemez állt rendelkezésemre, így hát valamilyen módon ki kellett bővítenem az adathalmazomat. Ezt a bővítést célzott módon tettem meg, amit a következő paragrafusokban fogok vázolni.

A lemezeket két csoportra osztottam, hogy lássam az adatok egy kezdetleges eloszlását. Ehhez egy klaszterező modellt választottam, mely a lemezek adatminőségét leíró metrikák értékeit használta bemenetként. Sajnos nem kaptam kielégítő eredményt, hiszen a kezdetleges kategóriák aránya 20%-80% volt, amiről mondanom sem kell, hogy közel sem optimális.

Mivel az előzetes klaszterezés eredményeképpen rendelkezésemre állt a kategóriák aránya, ezért úgy tudtam bővíteni az adathalmazt, hogy kiegyenlítsem ezt az arányt. Az augmentálásra (bővítésre) egy több célváltozós regressziós modellt használtam, melynek a bemenetei a karctérképek voltak, a kimenetei pedig az adatminőségi változók. Így ezt a modellt fel tudtam a későbbiekben használni az adatminőségi adatok generálására.

A következő lépés a képi adathalmaz bővítése volt, amihez egy erre a célra kifejlesztett generáló algoritmust használtam, amely bizonyos transzformációkat elvégezve új képeket generált. Az előbbiekben kifejtett előzetes csoportosítást használtam fel ahhoz, hogy az

¹*Holistically-Nested Edge Detection*

adott kategóriák képeiből pontosan annyit generáltassak, hogy kiegyenlítődjenek a csoportok. A szintetikus képek előállítását után a regressziós modell felhasználásával rendeltem hozzájuk az adaminőségi metrikák értékei.

Eredményül rendelkezésemre állt egy közel 500 elemből álló adathalmaz, amit szintén két csoportra osztottam, amihez ismételtén egy klaszterező modellt használtam fel. Így már megvoltak a karctérképek és a hozzájuk tartozó bináris kategóriák is (vagyis hogy egy lemez az adaminőséget tekintve jó vagy rossz).

Végezetül ezen adathalmazt felhasználva készítettem egy kompozit modellt, amely tartalmazza magát az osztályozó modellt illetve a jellemző tanulásra (*feature learning*) használt konvolúciós rétegeket. Ezeknek köszönhetően létrehoztam egy rendkívül pontos és általános modellt, amely kategorizálni tudja a lemezeket, vagyis meg tudja mondani, hogy egy lemez jó vagy rossz adaminőségű.

2. Felhasznált szoftverek

A dolgozatban bemutatásra kerülő munkához számos szoftvert használtam, melyek ebben a fejezetben kerülnek részletesen felsorolásra.

2.1. Integrált fejlesztői környezet

Integrált fejlesztői környezetként a Visual Studio Code-ot választottam, mely támogatja a Python programozási nyelvet, melyben jelen munka kódjai íródtak.

2.2. Anaconda

A megfelelő Python fejlesztői környezetet kialakításában az Anaconda² segített, mely egy tudományos célú Python és R disztribúció. Egy saját csomagkezelővel (conda) rendelkezik, mely révén több ezer szoftvercsomag telepíthető egyszerűen. A conda egyben egy környezetkezelő szoftver is, mely lehetővé teszi csomagok (beleértve magát a Python-t is) különböző verzióinak telepítését és az ezek közötti váltást. [?].

A munkámhoz szükséges csomagokat egy olyan conda környezetbe telepítettem, melybe a Python 3.10.0 számú verziója került.

2.3. Python csomagok

A kutatás során az alábbi Python csomagokat használtam:

- **tensorflow/Keras** (verziószám: **2.11.0**): Ezt a csomagot a lemezek bináris osztályozásánál használtam, azon belül is a konvolúciós rétegek létrehozásánál. Továbbá ezen csomag fel volt még használva a képi adatok kiterjesztésénél, generálásánál.
- **scikit-learn** (verziószám: **1.2.1**): A kutatás algoritmusainak javarésze ezen csomag segítségével készült el. Eszköztárából felhasználtam a különböző modellek tanítására szánt osztályokat (regresszió, osztályozás), az előfeldolgozásra használt részeit (tanító és validációs/teszt adatok szétválasztása), illetve a különböző leíró

²<https://www.anaconda.com/>

metrikákat kiszámoló függvényeit (pontosság, *loss* érték, mint például MAE, bináris keresztentropia és egyebek).

- **opencv-python** (verziószám: **4.7.0.68**): Mivel képi adatokat használnak a modellek, ezért elengedhetetlen az ezek feldolgozására és betöltésére szánt csomag használata, ami esetemben az OpenCV. Azért esett erre a választásom, mivel rendkívül széles képmanipuláló eszköztárral rendelkezik, mint például a képek transzformálása vagy más színtérbe történő alakítása.
- **pandas** (verziószám: **1.5.3**): Tekintve, hogy gépi tanuló modelleket használok, melyek bemenetei nem csak képek, hanem numerikus adatok is (itt a lemez adatminőségét leíró adatokra gondolok), azok kezelésére szükségem volt a jól ismert *pandas* csomagra. Ezt egyrészt a CSV fájlok elkészítéséhez, betöltésére és manipulálására (például az oszlopok módosítására és törlésére) használtam, másrészt pedig leíró statisztikák (például átlag, legkisebb és legnagyobb érték, percentilisek) értékeinek meghatározásához.
- **pickle**: Ez a Python egy szabványos modulja, melyet a *scikit-learn* csomagban, általam létrehozott és felparaméterezett modellek objektumainak mentésére használtam.
- **matplotlib** (verziószám: **3.7.0**): A különböző leíró ábrák létrehozására került felhasználásra.
- **seaborn** (verziószám: **0.12.2**): Bizonyos ábrák (például *pairplot*-ok) létrehozására volt használva.
- **numpy** (verziószám: **1.24.2**): Bizonyos matematikai műveletek (mátrixműveletek, vektor műveletek és egyebek) elvégzésére alkalmaztam.
- **beautifulsoup4** (verziószám: **4.11.2**): Ennek a csomagnak a segítségével nyertem ki az adatokat a nyílt forrású, adatminőséget kinyerő szoftver által generált HTML fájlból.
- **os, re**: Telepítésük szintén nem szükséges, hiszen ezek standard modulok.

2.4. QPxTool

Az adatminőségre és integritásra vonatkozó adathalmaz megalkotásához a szabad és nyílt forrású QPxTool³ szoftvert használtam, mely lehetővé tette ezen minőségre vonatkozó adatok kinyerését.

2.5. További szoftverek

Mindezekon kívül a felhasznált szoftverek közé tartozik még az operációs rendszer is, amely a Windows 10 Pro volt. Ezenfelül a programok elkészítéséhez *Jupyter Notebook*⁴ jegyzetfüzeteket használtam, melyek egy interaktív környezetet biztosítottak számomra, egyben egy flexibilis interfészt is nyújtottak a fájlok rendezésére.

³<https://qpxtool.sourceforge.io/>

⁴<https://jupyter.org/>

3. Hardver

A munkához a következő specifikációkkal rendelkező hardvereszközöket használtam:

- **Processzor:** Intel Core i9 10980XE @ 3.0GHz
- **Memória:** 128GB DDR4 @ 1198MHz
- **Videókártya:** NVIDIA GeForce RTX 3080
- **Háttértár:** 953GB Samsung SSD 970 PRO 1TB + HDD

4. Optikai lemezek adatminősége

A kutatás ezen részében az optikai lemezek adatminőségére és integritására vonatkozó adatok kinyerését és feldolgozását fogom részletezni. Szó fog esni a felhasznált szoftvekről, úgymint a gépi tanuló modellek implementálására szolgáló segédsoftver és maga az adatminőség-kinyerő szoftver.

Továbbá be lesznek mutatva azok a módszerek, amelyeket az adatok előfeldolgozására használtam, hogy azok olyan állapotba kerüljenek, hogy átláthatóak és értelmezhetőek legyenek. Ilyen módszer például a kiugró értékek eltávolítása, a dimenziócsökkentés és a skálázás.

4.1. Adatminőségi metrikák

A kutatásban DVD lemezeket vizsgáltam, melyek minőségének méréséhez többek között a következő metrikákat használják:

- **PIE** (*Parity Inner Error*): Egy DVD lemezen az adatok egy kétdimenziós tömbben vannak tárolva, melyekhez úgynevezett paritásellenőrző bitek (*parity check bits*) vannak fűzve. Minden ilyen kétdimenziós tömböt hibajavító kódblokknak (*error correction code block*) neveznek. A PIE metrika a kétdimenziós adattömbben előforduló azon hibás sorokat számolja, melyeket javítani lehet. Ezek a hibák minden DVD lemezen jelen vannak. Normál esetben a meghajtó hibajavító szoftvere javítani tudja ezen hibákat. [1]
- **PIF** (*Parity Inner Failure*): A PIE-hoz hasonlóan számított mennyiség, mely a kétdimenziós adattömbben előforduló azon hibás sorokat számolja, melyeket nem lehet javítani.

Ezen metrikák mérésére tökéletesen alkalmas volt a QPxTool. Meg kell jegyezni, hogy a dolgozatban bemutatásra kerülő eljárásokat CD lemezekre is alkalmazni lehetne mindössze annyi eltéréssel, hogy a PIE és PIF helyett CD-specifikus metrikákat kellene használni, mint például a BLER (*Block Error Rate*).

Az 1. táblázatban láthatóak a QPxTool által meghatározott hozzávetőleges határértékek, melyek alapján megkérdőjelezhető a DVD lemezek adatminősége.

Minimum (PIE)	Maximum (PIE)	Kategória
0	20	tökéletes
21	99	nagyon jó
100	280	jó
281	1664	rossz

1. Táblázat. A QPxTool által meghatározott kategória intervallumok

4.2. Az adathalmaz megalkotása

4.2.1. Adatok begyűjtése

A QPxTool alkalmazásban lehetőség van a lemez minőségét vizsgálni egy külön menüpontban. Az adatminőségi metrikák (PIE és PIF) értékei egy állományba exportálhatók.

A szoftver exportálási opciói között szerepelnek a HTML, PDF illetve a szoftver specifikus mentési módja, a QPX. Jómagam a HTML mentési mód mellett döntöttem, mégpedig az okból kifolyólag, mivel ez egy struktúrált dokumentum, melyből egyszerűen ki lehet nyerni a számomra érdekes és a kutatás számára releváns információkat.

Az adatok kinyeréséhez a BeautifulSoup⁵ Python csomagot használtam, mely adatok kinyerésére szolgál HTML illetve XML fájlokból. Lehetővé tette számomra az adatok kinyerését és azok keresését is a QPxTool-ból kiexportált dokumentumokból.

A feldolgozáshoz egy függvényt írtam, ami megkeresi a kinyerni kívánt adatokat tartalmazó táblázatot, abból pedig kinyeri a különböző `td` elemekben található leíró statisztikákat. Ennek köszönhetően minden egyes lemezből sikerült kinyernem a következő információkat:

- PIE (összes, maximum és átlag).
- PIF (összes, maximum és átlag).

A lemezek adatminőségéből kinyert statisztikai adatokat a 2. és a 3. táblázatban lehet megtekinteni.

⁵<https://www.crummy.com/software/BeautifulSoup/>

	PIE_total	PIE_max	PIE_average
count	100.00	100.00	100.00
mean	657394.43	95.99	5.63
std	1127448.89	75.36	9.71
min	211.00	15.00	0.00
25%	127671.25	32.75	1.44
50%	310809.50	59.00	2.92
75%	695277.25	165.50	5.75
max	9220364.00	231.00	64.86

2. Táblázat. A PIE értékek leíró statisztikái

	PIF_total	PIF_max	PIF_average
count	100.00	100.00	100.00
mean	99735.76	63.19	0.75
std	336446.44	78.05	2.37
min	0.00	0.00	0.00
25%	12938.75	21.00	0.11
50%	36835.00	26.00	0.30
75%	101910.75	49.50	0.82
max	3348419.00	230.00	23.56

3. Táblázat. A PIF értékek leíró statisztikái

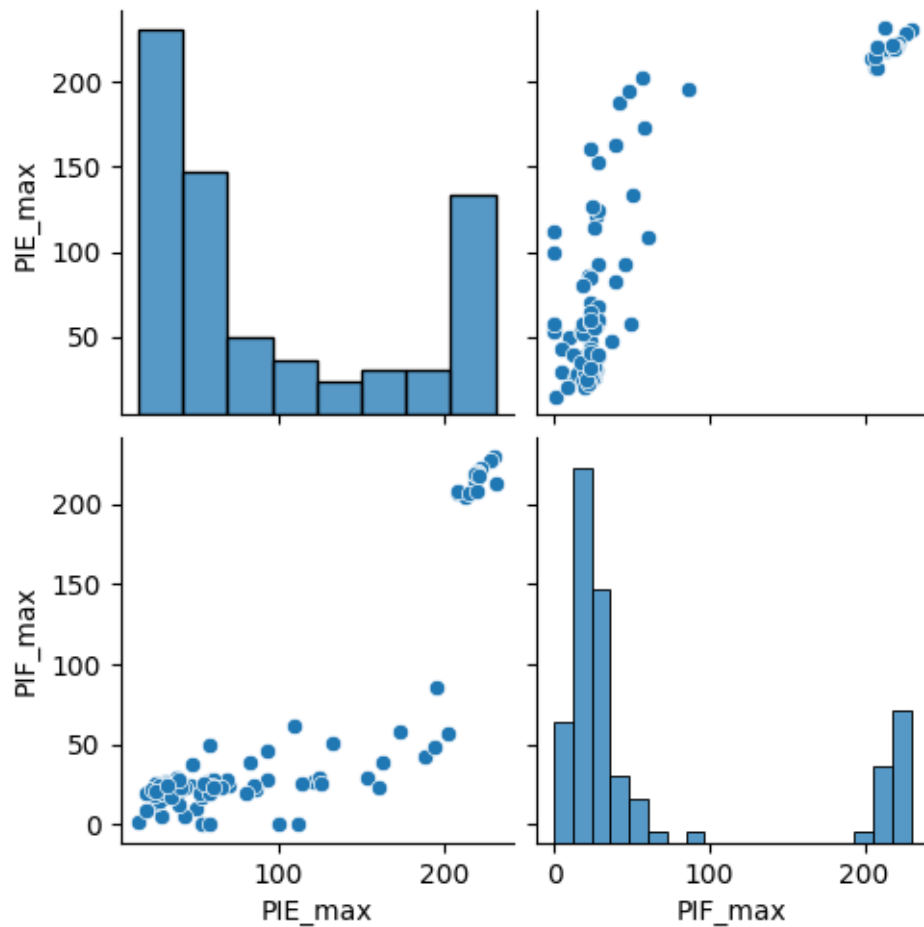
4.2.2. Adathalmaz előkészítése

Mivel a QPxTool számos leíró statisztikát készít az adott lemez alapján, ezért dimenziócsökkentésre van szükség. Nem csupán egyszerűsítési okokból kifolyólag, hanem mert nem feltétlenül releváns minden célváltozó. Azt is figyelembe kell venni, hogy az adatok megfelelőképpen legyenek elosztva, s mindezt optimálisan lehessen majd klaszterezni, hogy a klaszterezés értékelése és teljesítménye a legjobb legyen (ezen értékelési módokról a későbbiekben fogok szót ejteni). Továbbá, mint az a QPxTool dokumentációjából is kiderül, egy adott blokkban a maximális hibaértékek a fontosak, mivel a PIE és PIF metrikák is blokkonként mérendőek. Ezen okból kifolyólag tartottam meg csak a **PIE_max** és **PIF_max** oszlopokat, mivel ezek a blokkonkénti maximumokat mérik.

Amint azt az 1. ábrán (ami egy *pairplot*) is látható, még a dimenziócsökkentett adathalmaz is, vagyis a PIE_max és PIF_max metrika-párt tartalmazó adathalmaz, is rengeteg

oda nem illő kiugró értéket tartalmaz, amik megnehezítik a helyes klaszterezést.

Első lépésként a kiugró adatok eltávolítása előtt a teljes adathalmazt ugyanarra az intervallumra skáláztam. A táblázatból illetve az 1. ábráról jól látható, hogy míg a **PIE_max** értékei a $[15, 231]$ intervallumon mozognak, addig a **PIF_max** értékek a $[0, 230]$ intervallumon. Ez okból kifolyólag használtam az sklearn csomag által biztosított `MinMaxScaler` osztályt, mellyel minden adatot a $[0, 1]$ intervallumra skáláztam át.



1. ábra. A nyers adathalmaz páronkénti kapcsolatai

Ezután az adatokat meg kellett tisztítani ezektől a kiugró értékektől. Ezen értékek megtalálásához az úgynevezett **z-pontszám**on (*z-score*) alapuló adatszűrést alkalmaztam a teljes adathalmazon. A *z*-pontszám egy statisztikai mérőszám, amely az adott érték kapcsolatát írja le a csoportátlagához képest, melynek értéke lehet pozitív vagy negatív is. [2]

A *z*-pontszám által elvárt küszöbérték megállapításához az 1. ábrát használtam

segítségül. Mindaddig csökkentettem ezt a küszöbértéket, amíg az ábra jobb felső sarkában lévő kiugró értékek el nem tűntek. Ez az érték az 1,4 lett. A szűrés eredményeként az adathalmaz mérete 100-ról 79-re csökkent, és ezek az adatok már szinte tökéletes eloszlással rendelkeztek egy klaszterező modell számára.

Mivel egy lemezhez nem csak annak minőségére vonatkozó adatai tartoznak, hanem képi adatok is (ezekről később lesz szó), törölni kellett a kiugró minőségi adatokkal rendelkező lemezek képi adatait is az azonosítóik alapján (minden lemez kapott egy azonosítót a digitalizálás során).

4.2.3. Előzetes klaszterezés

Mivel nem állt rendelkezésemre elegendően sok adat, mindössze 79 darab, ezért az adathalmaz szintetikus bővítésére, augmentálására volt szükség. Viszont ehhez előbb fontos volt látni a már meglévő 79 adat csoportosulását, tendenciáját, illetve azt, hogy mely hiperparaméterekkel sikerül kihozni a legnagyobb leíró statisztikákat.

A klaszterező modell(ek) teljesítményének értékelésére a következő értékelési módokat használtam:

- **Silhouette pontszám:** Mivel kevesebb mint 100 adat állt rendelkezésemre, nem lett volna bölcs döntés, ha mindezt még jobban lecsökkentettem volna például egy tanító-validációs halmazra való bontással. Itt jött képbe a *Silhouette score*, hiszen míg a legtöbb teljesítményt vizsgáló módszernek szüksége van tanító adathalmazra, addig ennek kiszámításához nincs szükség tanító adathalmazra a klaszterező modell eredményeinek értékeléséhez. [3] Tulajdonképpen minél magasabb ez a pontszám, annál jobb az adott modell.
- **Calinski-Harabasz pontszám:** másnéven *Variance Ratio Criterion*, a klaszterek közötti szórás és a belső klaszter szórás összegének az aránya. [4] Irányadásképpen, minél nagyobb ez a pontszám, annál jobb a klaszterező modell teljesítménye. [4]
- **Davies-Bouldin pontszám:** Ez az átlagos „hasonlóságot” jelzi a klaszterek között, ahol a hasonlóság a klaszterek közötti különbségben méretik. [4] Ezen értékelő módszernél evidens, hogy minél kisebb a kapott szám, a modell annál jobban teljesít.

A modell felépítése előtt ismét a $[0, 1]$ intervallumra skáláztam az adatokat. Alapvetően az a probléma a többi klaszterező modellel, amelyek a klaszterszám előre megadását igénylik, hogy – bár előre meg van határozva, hogy 2 csoportba osztom őket – hiába rendelkeznek a legmagasabb leíró statisztikákkal, a kategóriák eloszlása nagyon rossz (a tesztelések során egy adott kategória túlnyomó többségben volt, közel 80% arányban).

Ez okból kifolyólag a csoportok kialakítására a **DBSCAN** klaszterező modellt választottam. Ennek köszönhetően egyenletesen tudtam felosztani az adathalmazt két csoportra.

4.3. Eredmények

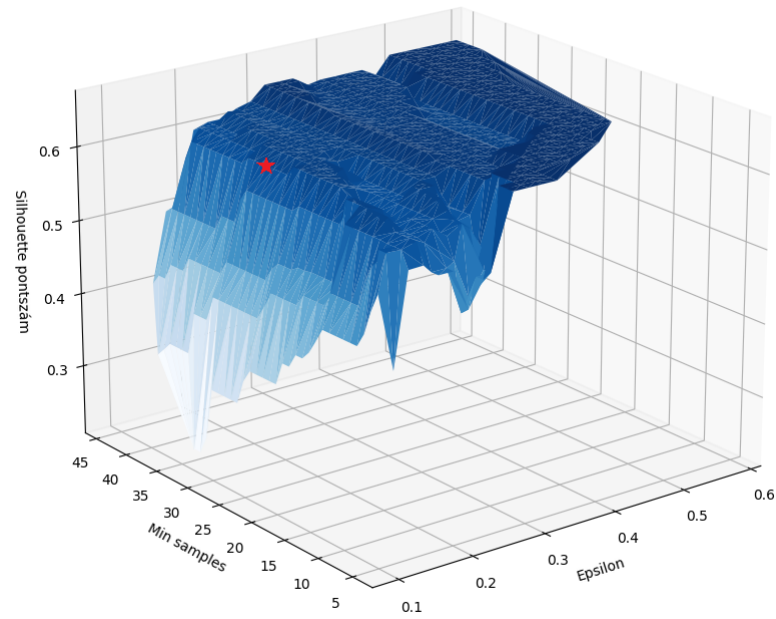
A megfelelő hiperparaméterek kiválasztásakor próbáltam figyelembe venni mindhárom statisztikai változót, vagyis hogy a lehető legjobb értékeket vegyék fel egyszerre. A választásom az **epsilon = 0,18** és **min_samples = 30** paraméter-párra esett, mely pontot szemléltetésképpen be is jelöltem egy piros csillaggal a 2., 3. és 4. ábrákon. Mint azt látni is lehet, a 30 az pont egy optimális szám, hiszen 79 adatom van, s ezáltal a lehető legarányosabb eloszlást sikerült elérni. Ezeket az értékeket használva a modell teljesítménye (két tizedes pontosságot használva) a következő:

- Silhouette pontszám: 0,61 (lásd a 2. ábrát)
- Calinski-Harabasz pontszám: 105,14 (lásd a 3. ábrát)
- Davies-Bouldin pontszám: 0,74 (lásd a 4. ábrát)

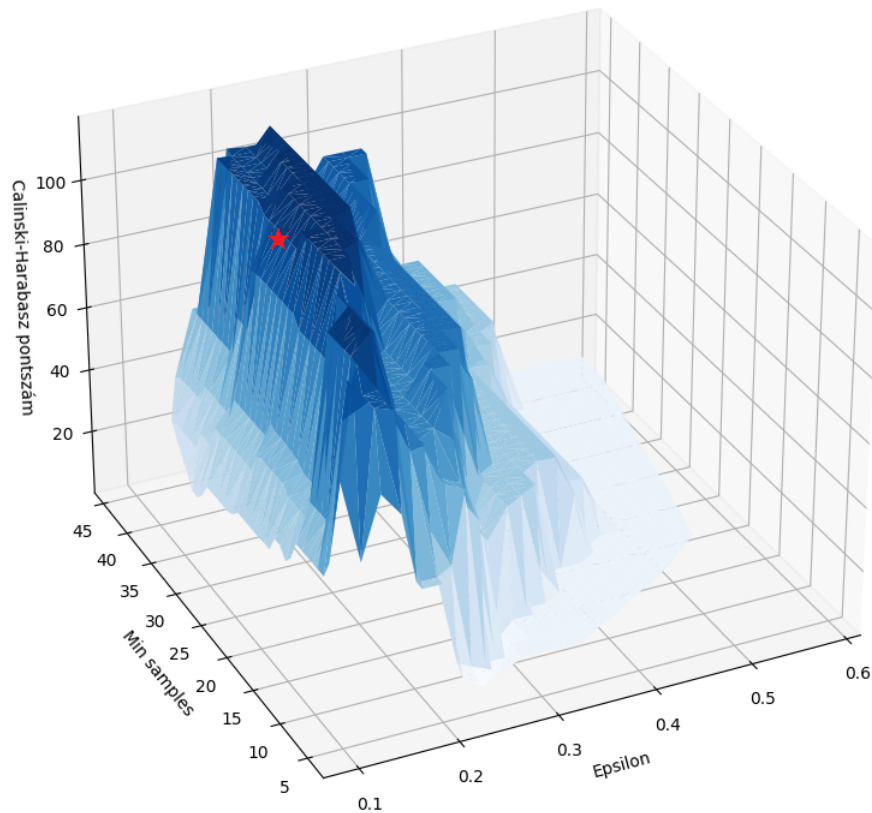
A fenti számok nem tűnhetnek optimálisnak és helytállónak, viszont pontosan ezért lesznek az adatok célirányosan kiterjesztve, hogy az osztályozó modell minél nagyobb pontosságot érhesen el, illetve hogy a csoportok eloszlása kiegyenlített legyen. A model által kialakított kategóriák eloszlását az 5. ábrán lehet megtekinteni.

A DBSCAN által kialakított csoportok eloszlását pedig a 6. ábrán lehet szemügyre venni, ahol a klaszterek középpontjait egy fekete csillaggal jelöltem.

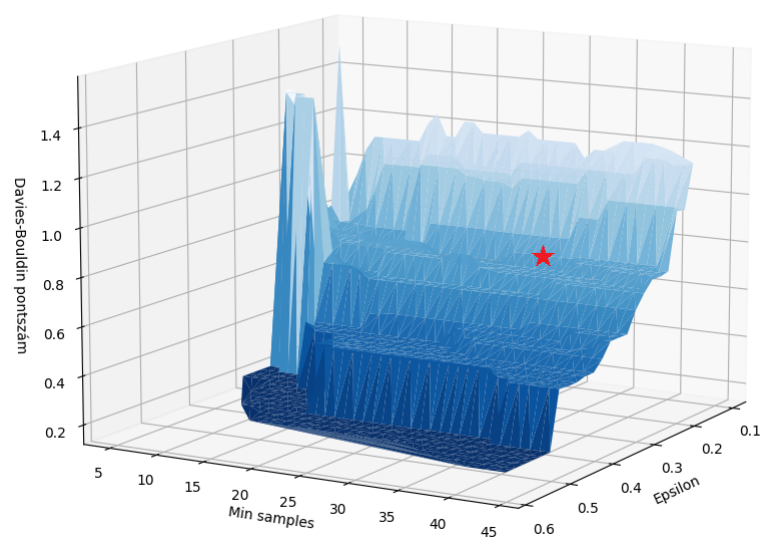
Végezetül a modell által meghatározott kategóriákat hozzáfűztem a letisztított adathalmazhoz (ami 79 elemet tartalmaz), amit végül kimentettem egy CSV fájlba.



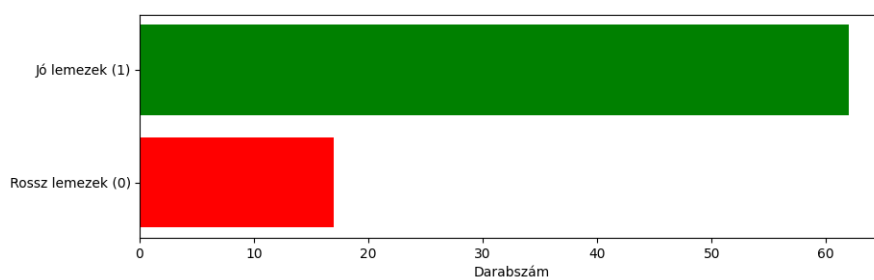
2. ábra. A *Silhouette* pontszám értéke az **epsilon** és **min_samples** paraméterek függvényében



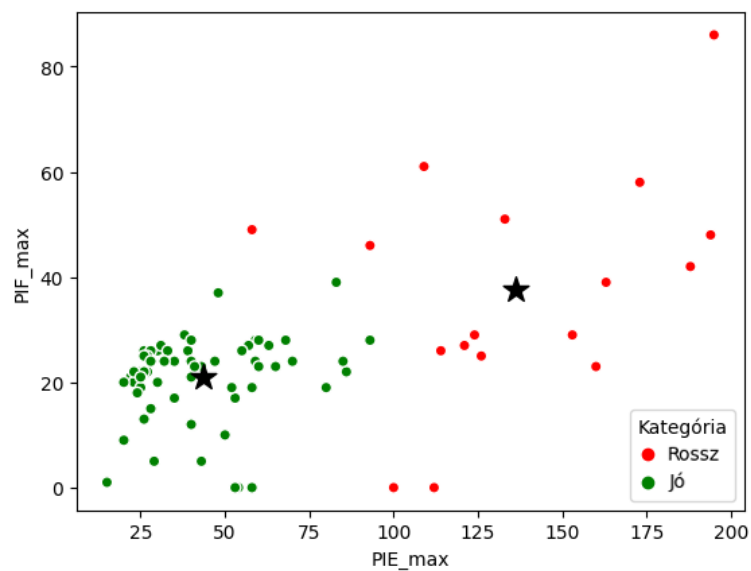
3. ábra. A Calinski-Harabasz pontszám értéke az **epsilon** és **min_samples** paraméterek függvényében



4. ábra. A Davies-Bouldin pontszám értéke az **epsilon** és **min_samples** paraméterek függvényében



5. ábra. Az előzetes klaszterezés eredményeként kapott kategóriák aránya

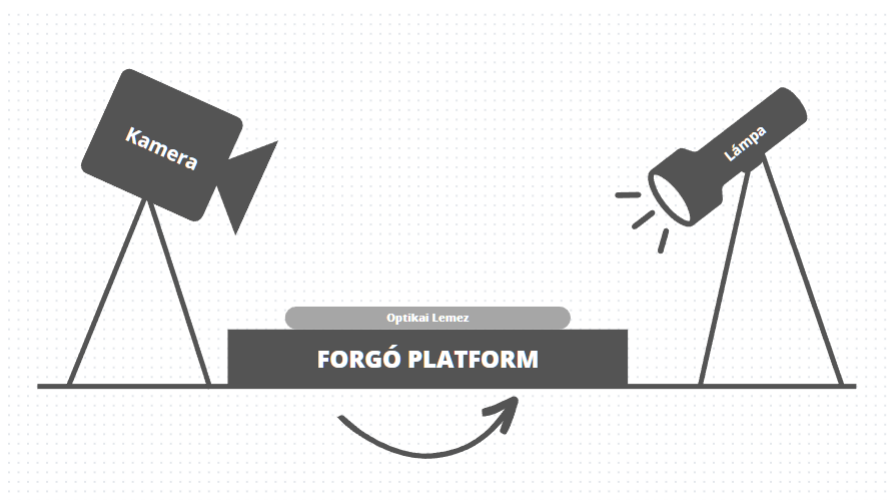


6. ábra. A célváltozók páronkénti kapcsolata az előzetes klaszterezés eredményeként kapott csoportoknál

5. A lemezek vizuális reprezentációja

A dolgozat ezen részében arról esik szó, hogy hogyan kerültek létrehozásra karctérképek a lemezek alsó oldaláról készült fényképek felhasználásával.

5.1. Eszközök és környezet leírása



7. ábra. A digitalizálásra használt környezet és eszközök

A lemezek felületének lefotózásához a 7. ábrán látható „szerkezetet” készítettem el. A szerkezet a következő komponenseket tartalmazza:

- **Platform:** egy 3 cm magasságú, kör alakú platform, melyet 360 fokban lehet körbeforgatni.
- **Kamera állvány:** egy hétköznapi kamera állvány, melynek magassága 14 cm.
- **Lámpa állvány:** egy fényképezésre használt mini állvány, magassága 24 cm.
- **Kamera:** erre a célra egy Honor 8X telefont használtam az alábbi kamera beállításokkal:
 - **Felbontás:** 20MP (5120×3840)
 - **F-stop:** 1,8
 - **Expozíciós idő:** 1/33s

– **Érzékenység (ISO):** ISO-200

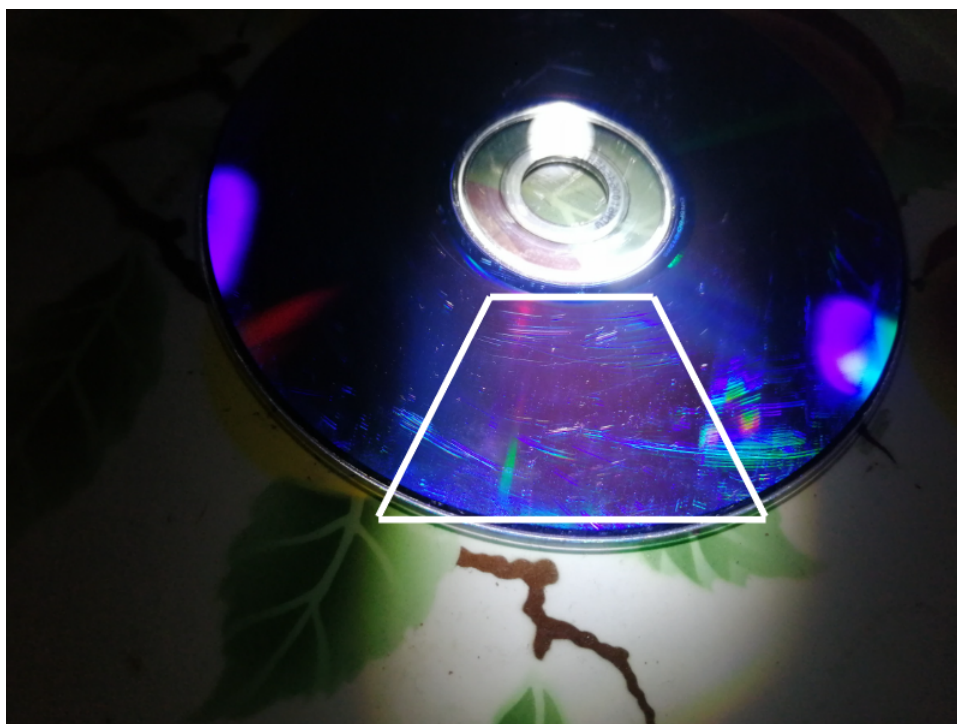
– **Fókusz távolság:** 4mm

A kamera szöge 50 fok volt, távolsága a lemez középpontjától pedig 15 cm.

- **Fényforrás:** fényforrásként egy ledes zseblámpát használtam, melynek szöge az platformhoz képest 40 fokra volt beállítva, illetve a lemez középpontjától számítva 17 cm-re volt helyezve.

5.2. Lemez fotózás

Tapasztalataim szerint nagyon nehéz a lemezek felületéről olyan fotót készíteni, melyen látszódnak a karcolások. A karcok láthatóságához a lemezek felületét egy bizonyos szögből kell megvilágítani, és a fotót is egy megfelelő szögből kell készíteni. A 8. ábrán látható egy olyan fotó, melyen jól kivehetők a karcok. Szembetűnő, hogy a fotón mindössze a lemez egy kb. 45 foknyi része látható. Ezért minden lemezeről összesen 8 darab képet kellett készítenem ($8 \times 45^\circ = 360^\circ$). A nyolc fotó készítéséhez a platformot 45 fokonként forgattam el.



8. ábra. Nyers lemezkép

A fotók mindegyikéből ki kellett vágnom a lemez egy 45 fokos szeletének megfelelő részt, melyet a 8. ábrán egy fehér trapéz ábrázol. Így nyolc kisebb kép keletkezett, melyek mindegyike a lemez karctérkép egy részleteként tekinthető. A nyolc képet egy nagy képpé fűztem össze, ezáltal egy teljes karctérkép reprezentációt kaptam (lásd a 9. ábrát).



9. ábra. Egy teljes lemezkép-panoráma

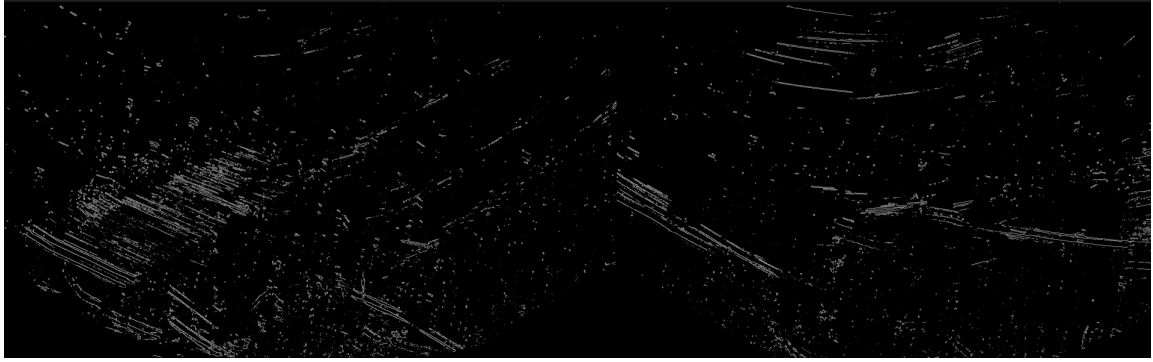
5.2.1. Karcolások kinyerése

A munka egy következő fontos pontja az volt, hogy a karcolásokat teljes egészükben, jól kivehető módon tudjam kinyerni. A képfeldolgozásban számos lehetőséget lehet találni az élek kinyerésére, többek közt az egyik leghíresebb a Canny módszer, melyet John F. Canny fejlesztett ki. Ezen elgoritmus a következő lépéseket foglalja magába [5]:

- **Zajcsökkentés:** egy 5×5 méretű Gauss-szűrővel történik.
- **A kép intenzitás gradienseinek megkeresése:** a képre egy Sobel-szűrő kerül alkalmazásra függőlegesen és vízszintesen is.
- **Nem-maximális elnyomás:** a pixeleken való áthaladást jelenti, mely eltávolítja a nem kívánt pixeleket, melyek nem számítanak élnek.
- **Hiszterézis küszöbérték:** ebben a fázisban lesz eldöntve, hogy mely élek bizonyulnak igazi élnek.

Elméletileg ez az algoritmus megfelelő megoldást adthat a problémára, viszont úgy találtam, hogy egy másik módszer (HED) sokkal tisztább éleket tud kinyerni egy lemezképből.

A Canny algoritmust alkalmazva egy képen a 10. ábrán látható eredményt kaptam.



10. ábra. Canny karctérkép

Mint az látható is, a Canny algoritmus alkalmazása egy nem túl részletgazdag képet eredményezett, mely a HED által készített karctérképhez képest mindössze nagyjából tizedannyi információt tartalmaz, amely a modell pontatlanságához vezetne. Ennek bizonyításához betanítottam az osztályozó modellt az eredeti, még nem kiterjesztett adathalmazzal, mely révén a 4. táblázatban látható eredményeket kaptam.

	Modell pontossága	Bináris kereszt-entrópia	Fehér-fekete pixelek aránya
Canny módszer	95%	0,19	0,8%
HED módszer	100%	0,04	8,4%

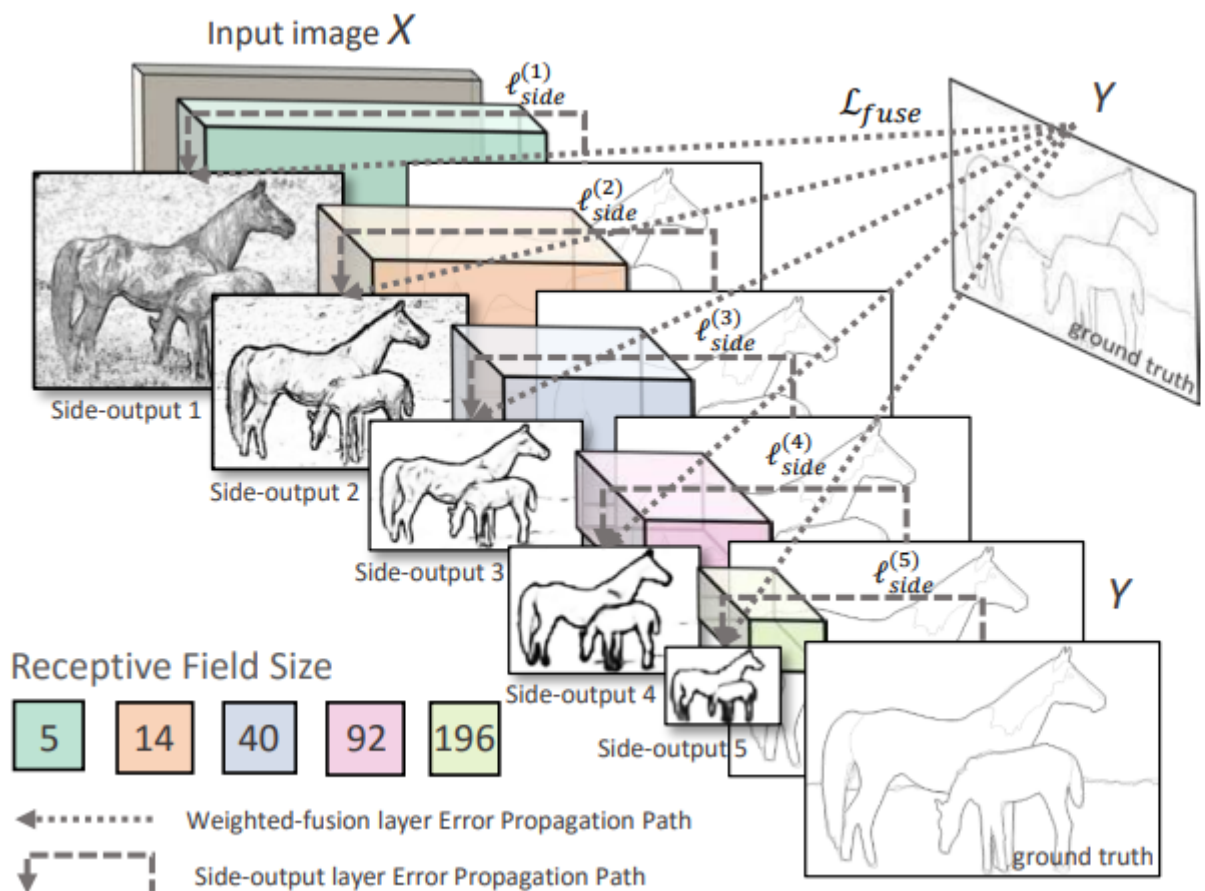
4. Táblázat. Canny vs. HED módszer

A táblázatban lévő adatok alapján kijelenthető, hogy bár a modell pontosságát nem sokban javítja a HED háló használata, viszont a modell általánosító képességét nagyban javítja (ezt jelzi a bináris kereszt-entrópia). Minél kisebb a kereszt-entrópia, annál kevésbé függ a modell a tanító adathalmaztól, ezáltal jobban teljesít új bemeneti adatok esetén. Végezetül pedig az is látható, hogy a HED háló sokkal (a Canny algoritmushoz képest nagyjából tízszer) több részletet tudott kinyerni egy lemezképből.

Ezen motivációból kiindulva döntöttem végül a **holisztikusan beágyazott** (*holistically-nested*) élkinyerő konvolúciós neurális háló mellett (röviden HED), mely sokkal szebb és részletgazdagabb képet tudott generálni.

Mivel egy olyan neurális hálóról van szó, mely konvolúciós rétegeket is tartalmaz, illetve a bemenetként használt képek is nagyok, azt gondolná az ember, hogy nagy a modell erőforrásigénye. Szerencsére ez egy előre definiált és betanított modell, mely szabadon elérhető és használható.

A HED modell célja, hogy olyan módon tanulja meg a jellemzőket, hogy ennek alapján egy, a valósághoz közeli éltérképet lehessen készíteni. [6] Továbbá, mint azt egyes munkák igazolták is, az előre betanított neurális hálók finomhangolása rendkívül hasznosnak bizonyult az alacsony szintű élkinyerő feladatokhoz is. [6] Éppen ezért a HED szerkezetének szerves részét képezi a VGGNet architektúrájának egy része, a következő változtatásokkal: a VGGNet minden fázisának utolsó konvolúciós rétegéhez van hozzákötve a fejlesztők által kifejlesztett egyedi kimenet, és így alakul az élkinyert kép fázisról fázisra.[6] Ezt az architektúrát a 11. ábra szemlélteti.



11. ábra. A HED háló architektúrája [6]

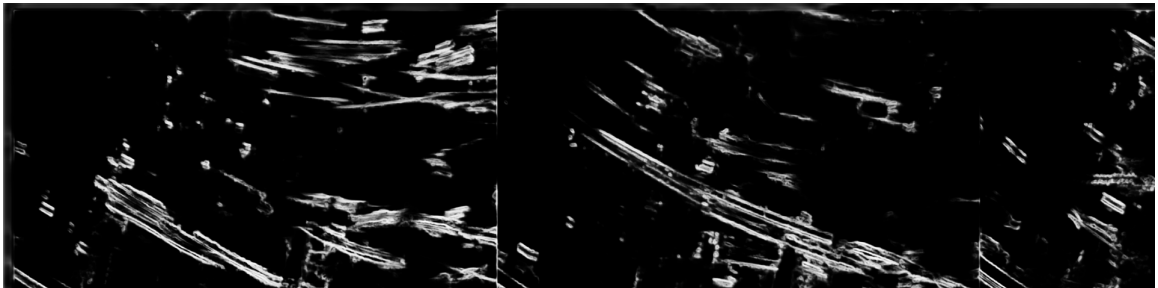
A modell használatához szükség volt a `deploy.prototxt`⁶ fájlra, mely tartalmazta a háló teljes szerkezetét, illetve egy előre betanított `caffee` fájlra⁷. Ezek kombinációjával, illetve az OpenCV `dnn` csomagjában lévő `readNetFromCaffe()` metódust használva inici-

⁶<https://github.com/s9xie/hed/blob/master/examples/hed/deploy.prototxt>

⁷https://github.com/ashukid/hed-edge-detector/blob/master/hed_pretrained_bsds.caffemodel

alizáltam a hálót, majd némi előfeldolgozás után (ez alatt a blobok elkészítését, a neurális háló bemenetének beállítását illetve a háló előre léptetését értem) kinyertem a karcolásokat a képekből. Ennek a modellnek a segítségével a 12. ábrán látható eredményt sikerült elérnem.

Viszonyításképpen ez ugyanannak a lemezképnek egy kivonata, amelyet a fentiekben bemutatott Canny módszernél is látni lehet. Tisztán látható, hogy ez a módszer sokkal jobb eredményt produkált, mint az előbbiekben vizsgált Canny algoritmus.



12. ábra. HED karctérkép

6. Augmentálás

A kutatás ezen pontján rendelkezésemre állt egy 79 elemből álló adathalmaz, amely tartalmazta a minőségre vonatkozó kettős célváltozókat (PIE és PIF), illetve a karctérképeket. Az osztályozó modell megfelelő működéséhez elengedhetetlen egy nagy adatállomány, ezért ki kellett bővítenem az adathalmazt szintetikus generált adatokkal. A képeket osztályozó modelleknél ajánlott továbbá a képi adatállomány augmentálása, hogy a modell minél több fajta képet lásson. Ennek köszönhetően a modell elég jól fog tudni általánosítani korábban még nem látott bemenetekre.

Az általam készített szerkezettel akár 15-30 percig is eltartott sajnos egy lemez digitalizálása. Nyilvánvaló, hogy nem járható út ezzel a módszerrel nagyobb mennyiségű lemez digitalizálása. Továbbá az sem elhanyagolandó szempont, hogy az egyik osztály nagyobb számban volt reprezentált.

Ezen probléma kiküszöbölésére két megoldást lehet alkalmazni: csökkenteni a többségi osztályban az adatok mennyiségét vagy növelni a kisebbségi osztályban lévők mennyiségét. Az első opció választása nem lett volna bölcs döntés, mivel eleve kevés volt a rendelkezésemre álló adatmennyiség, s ha ezekből az adatokból még töröltem volna, akkor még rosszabb helyzetbe kerültem volna adat-diverzitás tekintetében.

Ezen okokból kifolyólag jött az az ötlet, hogy arányosan kiterjeszthetném mind a képi adatokat, mind a minőségre vonatkozó adathalmazt, hogy minden kategóriájú lemezből ugyanannyi legyen.

6.1. Képi adatok augmentálása

Mivel az előzetes klaszterezésnél az algoritmus által meghatározott bináris célváltozót egy külön oszlopként fűztem hozzá a minőségi adatokhoz, ahol szerepeltek a lemezek azonosítói is, be tudtam azonosítani a lemezeket és a hozzájuk tartozó kategóriát. Továbbá a generált karctérképek fájlneveiben is szerepeltek az azonosítók.

Miután betöltöttem az összes adatot, vagyis a klasztert jelző illetve a karctérkép adatokat, elkezdtem augmentálni a képi adatokat.

Az algoritmus elindítása előtt előre definiáltam, hogy mennyi képet szeretnék generálni osztályonként (erre gondoltam a célzott kiterjesztés fogalom alatt a fentiekben), hogy

eredményül minden kategóriából ugyanannyi kép álljon rendelkezésemre (ez esetben 300 kép volt osztályonként, vagyis összesen 600 darab).

A képi adatok kiterjesztésére a Keras API `preprocessing.image` csomagjában lévő `ImageDataGenerator` osztályt használtam fel. Noha ezt a mélytanuló modelleknél szokták alkalmazni, alkalmas képgenerálásra is.

A paraméterezés meghatározásakor figyelemmel követtem az algoritmus által generált képek milyenségét, hogy azok reálisan nézzenek ki, tehát hogy minél közelebb álljanak egy valós karctérképhez. A paraméterek és értékeik az alábbi kódban láthatók:

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    width_shift_range=0.05,
    zoom_range=0.01,
    shear_range=0.1,
    height_shift_range=0.0,
    vertical_flip=False,
    horizontal_flip=True,
    fill_mode="constant",
    cval=0
)
```

Az első négy numerikus paraméter értéke a következőképpen értendő: mindegyik egy $[0,1]$ intervallumbeli érték, amely a teljes képhez való viszonyt jelzi. Tehát a `width_shift_range` paraméter esetében egy $\pm 5\%$ -os eltolást fog majd elvégezni a generátor.

Két másik fontos paraméter a `fill_mode` és a `c_val`. Ezek az értékpárok fogják meghatározni, hogy az eltolás helyénél milyen értékkel és milyen módon legyenek kitöltve az üres pixelek. A karctérképek bináris színcsatornával rendelkeznek, ezért evidens volt, hogy egy konstans kitöltésmódot (`fill_mode="constant"`) választok, s mindezt a fekete színnel (`cval=0`).

A fenti generátor által előállított képek egyike a 13. ábrán látható.



13. ábra. Szintetikus karctékép

6.2. Adatminőségi halmaz augmentálása

Az adatok minőségét tartalmazó adathalmaznál már nem volt ilyen egyszerű dolog, mint az előbbieken, ahol csak egy generátort kellett alkalmaznom, hiszen itt nem volt ilyenre lehetőség. Lehetett volna valamilyen zajt adni az adatokhoz, és ezáltal növelni azok mennyiségét, viszont mindez nem lenne reális.

A problémát egy gépi tanuló modellel oldottam meg, melynél a bemenet az eredeti, még nem kiterjesztett (79 elemet tartalmazó) képi adathalmaz volt, a kimenet pedig a két célváltozó, a **PIE_max** és a **PIF_max**.

Meg kell jegyezni, hogy nagy adathalmaz esetén a neurális hálók – akár konvolúciós rétegekkel, akár nélkülük – sokkal pontosabb eredményt tudnak adni egy olyan egyszerűbb gépi tanuló algoritmushoz képest, mint például a logisztikus regresszió. Viszont esetemben ez nem állt fenn, mivel csupán 79 elemből áll az adathalmazom, ami túl kevés egy mélytanuló algoritmus használatához. Ezért egy olyan köztes megoldást kellett alkalmaznom, amely alkalmas volt erre a feladatra.

A regressziós modellhez 20%-os kicsinyítést alkalmazva töltöttem be a képi adatokat (mivel ez is bőven elegendő volt a modell számára), betöltöttem továbbá a célváltozó párokat is. Hogy meg tudjak bizonyosodni a modell teljesítményéről, az adathalmazt ezután egy tanító és egy validációs halmazra osztottam fel, melynek aránya 80%–20% volt. Majd újraskáláztam a képi adatokat oly módon, hogy a pixel értékeket elosztottam 255-tel, ezáltal átskálázva őket a $[0, 255]$ intervallumról a $[0, 1]$ intervallumra. Azért volt erre szükség, mivel így a modell pontosabb és gyorsabb is lett.

Mivel képi adatokról van szó, a neurális hálóknál alkalmazott konvolúciós rétegek jellemző tanulást (*feature learning*) tesznek lehetővé. Ennek köszönhetően a képek jel-

lemezőinek manuális kinyerése nélkül érhető el a jellemző tanulás. Ezért alkalmaztam a betöltött képeken az említett konvolúciós rétegeket, melynek köszönhetően több mint kétszeres jellemzőszám növekedést sikerült elérnem, ami a lehetővé tette egy elég pontos regressziós modell felépítését és betanítását.

Az alábbi rétegeket használtam:

- **Conv2D**: paraméterei `filters = 8`, `kernel_size = (3, 3)`, `padding = 'same'`, `activation = 'relu'` és `input_shape = (180, 2100, 1)` (utóbbi a bemeneti képek méretét hivatott jelölni).
- **MaxPooling2D**: paraméterei `pool_size = (2, 2)` és `strides = 6`.
- **Flatten**: ez a réteg felel a 180×2100 -as méretű képek egy dimenziós vektorra történő alakításáért.

A modell kiválasztásánál ügyeltem arra, hogy gyorsaság tekintetében is optimális legyen, a pontossága is aránylag jó legyen, illetve hogy a hiba értéke is a lehető legkisebb legyen. Végül az `scikit-learn` könyvtár `MultiOutputRegressor` osztályára esett a választásom, tekintve hogy két célváltozóval dolgoztam. Ezenkívül szükségem volt még egy belső becselő objektumra is, amely a fentiek alapján a `LinearRegression` osztály egy példánya lett.

A legmegfelelőbb hiperparaméterek meghatározásához az `sklearn` csomag `GridSearchCV` osztályát használtam, mely az adott modellt az összes paraméterpárra kipróbálta és értékelte. Az általam használt paraméter-rács illetve a `GridSearchCV` paraméterei a következők voltak:

```
param_grid = {  
    'estimator__fit_intercept': [True, False],  
    'estimator__positive': [True, False]  
}  
estimator = LinearRegression()  
  
grid_search = GridSearchCV(MultiOutputRegressor(estimator),
```

```
param_grid=param_grid,  
scoring='neg_mean_absolute_error')
```

Mint az látható is, a pontozásra a negatív átlagolt abszolút hiba értékét használtam fel, mivel szerettem volna, hogy a modell minél jobban általánosítson, vagyis hogy jobb eredményt adjon új, még nem látott adatok esetén is.

A fenti rácskeresést lefuttatva a következő paramétereket és pontszámot kaptam:

- **Negatív MAE:** $-22,56$.
- **Optimális paraméterek:** `fit_intercept: True, positive: False`.

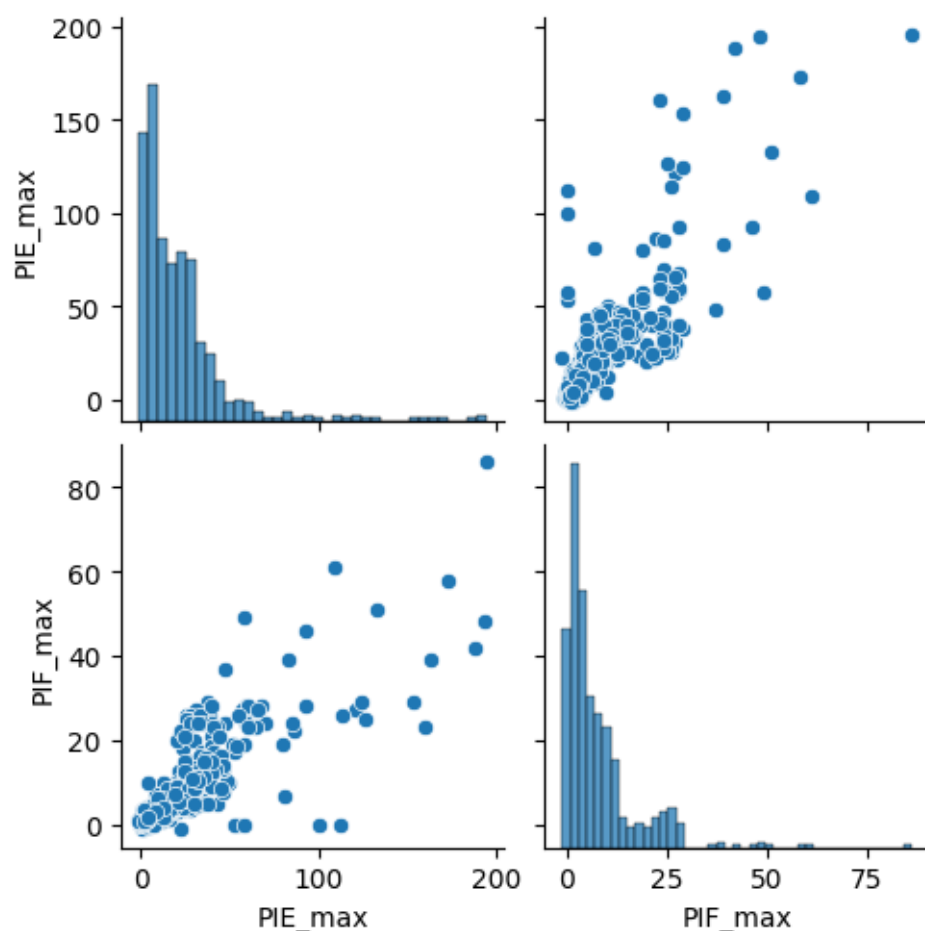
Ezt a modellt használva a fenti hiperparaméterekkel sikerült elérnem egy átlagolt, kb. $\pm 34\%$ -os hibaküszöböt, mely célváltozóként a következőket jelentette: kb. $\pm 36\%$ hibaküszöb a **PIE_max** esetében és $\pm 32\%$ küszöbérték a **PIF_max** esetében. Ezek a számok a **MAE** (*Mean Absolute Error*) teljesítményt mérő statisztikai változót használva lettek kiszámolva.

Mindazonáltal ezek magas értékeknek bizonyulhatnak, viszont (mint azt majd a későbbiekben megmutatom) az adathalmaz eloszlása nem változott sokat, sőt még jobban el lettek osztva az adatok ezen augmentációnak köszönhetően.

6.3. Augmentált adatok asszociálása és új adathalmaz megalkotása

A fenti cím alatt az augmentált képi adatokon alapuló új adatminőségi adatok generálását értem, mégpedig az előzőleg bemutatott lineáris regresszióval.

Az összes kép – beleértve az eredeti adathalmazt és az augmentáltat is – és az elkészített modell betöltése után prediktáltam az adatintegritást leíró két célváltozó-pár értékét. Annak érdekében, hogy a végleges adathalmaz tartalmazza az eredeti adathalmaz elemeit is, az augmentált adathalmazhoz hozzáfűztem az eredeti, 79 elemből álló adatállományt is. Ezáltal sikerült elérnem, hogy a kiterjesztett adathalmaz kb. 16% -a az eredeti adathalmazból származzon. Ezen adatelemek eloszlása a 14. ábrán látható.



14. ábra. A kiterjesztett adathalmaz páronkénti kapcsolatai

Ezek után mentettem az augmentált adatokat, illetve azonosítókat is fűztem hozzájuk és az aktuális képekhez a későbbi beazonosítás megkönnyítésének érdekében.

A fenti lépések után rendelkezésemre állt egy 533 elemből álló adathalmaz, mely tartalmazta a szintetikusan kialakított adatokat is. Azért nem tartalmazott az adatállomány pontosan 600 darab elemet, mivel a generálandó képmennyiség nem mindig volt egész szám. Ha például az egyik kategóriából 200 darab szintetikus képre volt szükség és a halmazban eredetileg 34 ilyen kategóriájú kép volt, akkor ezen számokat elosztva 5,88 képet kellene generálni elemenként, ami természetesen nem lehetséges. A nem egész számokat lefelé kerekítettem, tehát például 5,82 kép esetén az algoritmus 5 képet generált.

6.4. A végső klaszterek megállapítása

Az augmentáció utolsó lépése a végső kategóriák kialakítása volt, ehhez szinte ugyanazok a módszerek kerültek alkalmazásra, mint az előzetes klaszterezésnél.

Mint azt a 14. ábrán is lehet látni, ismételten keletkeztek kiugró értékek. Ezen értékek kiküszöbölésére szintén a fentiekben is alkalmazott z -pontszámon alapuló szűrést alkalmaztam, viszont most a küszöbértéket 3-ra állítottam be, melyet a gyakorlatban általában használnak. Meg kell említeni, hogy az adatokat a szűrés előtt a $[0, 1]$ intervallumra normalizáltam, megkönnyítve ezzel az adathalmazzal kapcsolatos további teendőket.

A gépi tanuló modell tekintetében szintén egy nem felügyelt tanításon alapuló modellt használtam, mégpedig a DBSCAN-t, ahol fel tudtam konfigurálni a fontos hiperparamétereket, a `min_samples`-t és az `epsilon`-t. Ebben az esetben is arra próbáltam figyelni, hogy a leíró statisztikák mellett a címkék aránya is megfelelően alakuljon.

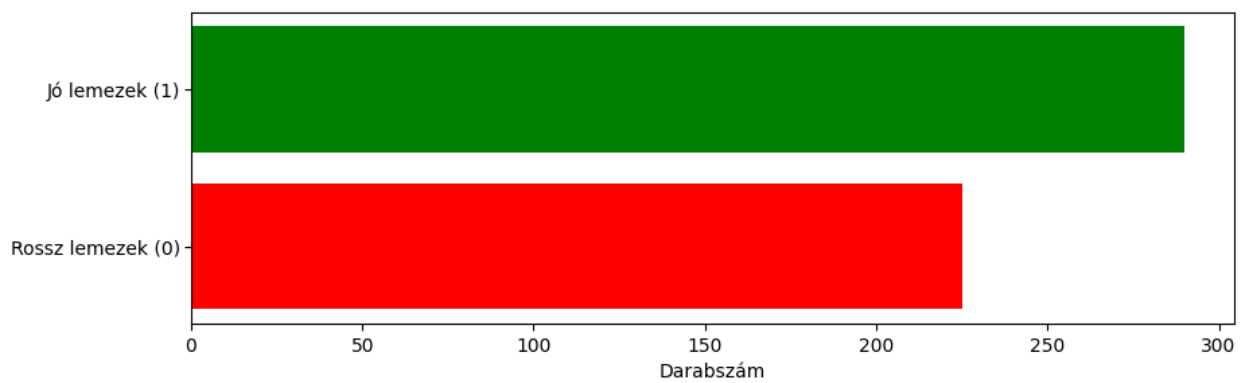
Az `epsilon` paraméter értékeként a 0,05 értéket használtam, a `min_samples` értékeként pedig 250-et, mivel ez lefedi nagyjából a teljes adathalmaz felét.

Ezeket a paramétereket használva végül a következő teljesítmény-különbségeket figyelem meg:

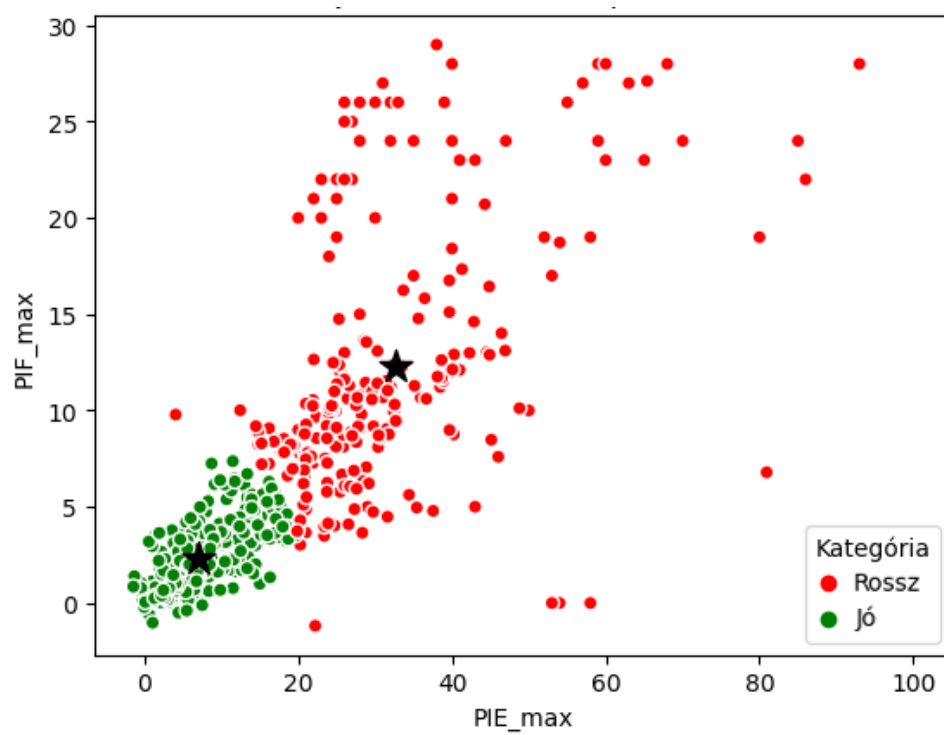
- **Silhouette pontszám:** 0,54
- **Calinski-Harabasz pontszám:** 647,30
- **Davies-Bouldin pontszám:** 0,69

Bár a Silhouette pontszám 13%-kal rosszabb lett, addig a Calinski-Harabasz pontszám közel 600%-kal javult, a Davies-Bouldin pontszám pedig 7%-kal nőtt, nem beszélve a kategóriák eloszlásáról, amely sokkal jobb lett, amint az a 15. ábrán is látható.

A 16. ábra pontdiagramján az újonnan kiterjesztett adathalmaz pontjainak a csoportosulását lehet megtekinteni. A csoportok középpontjait most is egy fekete csillaggal jelöltem.



15. ábra. A kiterjesztett adathalmaz kategóriáinak eloszlása



16. ábra. A kiterjesztett és előfeldolgozott adathalmaz csoportosulása

7. Osztályozás

Végezetül szeretnék szót ejteni a kutatás fő eleméről, az osztályozó modelről. Eleinte úgy gondoltam, hogy mivel képeket kell majd osztályozni, jól jöhet egy konvolúciós neurális háló, viszont ehhez túl kevés a közel 500 elemű adathalmaz. A másik opció sem túl jó, vagyis hogy csak egyszerű logisztikus regresszió legyen használva önmagában. Hiszen éppen azért szokás konvolúciós neurális hálókat használni, mert azoknál nem kell kinyerni a jellemzőket manuálisan, hanem a háló maga tanulja meg őket (*feature learning*) eltérően a logisztikus regressziótól, melynél kézzel szükséges kinyerni a jellemzőket (*feature engineering*).

Erre a problémára azt a megoldást találtam ki, hogy miért is ne kombináljam e kettőt? Vagyis hogy tartsam meg a jellemző tanulást biztosító konvolúciós rétegeket és a kevés adattal is jól beérő lineáris modellt. Éppen ezért is készítettem egy olyan hibrid modellt, amely e kettőt ötvözi, és amelynek a következő alfejezetben fogom leírni a szerkezetét és felépítését.

7.1. A modell felépítése és szerkezete

Az osztályozó modell felépítésénél fontos szempont volt a megfelelő rétegek használata és azok optimális felparaméterezése. Ezen szempontok azért voltak fontosak, mert nagyon vékony a határ egy modell túltanítása (*overfitting*) és alultanítása (*underfitting*) között.

A modellt egy sor – pontosabban három – réteggel kezdtem felépíteni, melyek segítségével a képekből sokkal gördülékenyebben ki lehetett nyerni a jellemzőket. A három konvolúciós réteg:

- **Conv2D:** paraméterei `filters = 8`, `kernel_size = (3, 3)`, `padding = 'same'`, `activation = 'relu'` és `input_shape = (180, 2100, 1)` (utóbbi a bemeneti képek méretét hivatott jelölni).
- **MaxPooling2D:** paraméterei `pool_size = (2, 2)` és `strides = 6`.
- **Flatten:** ez a réteg felel a 180×2100 -as méretű képek egy dimenziós vektorra történő alakításáért.

Ezeket követi a logisztikus regresszió alkalmazása, mivel ez adta a legjobb eredményt az összes többi osztályozó modellel összehasonlítva. A tanítás előtt a képek át lettek méretezve eredeti méretük 20%-ára, illetve a pixel értékek el lettek osztva 255-tel a $[0, 1]$ intervallumra való skálázás céljából.

A megfelelő hiperparaméterek kiválasztásához itt is a `GridSearchCV` osztályt használtam, amely minden lehetséges paraméter kombinációval betanítja és értékeli az általam kiválasztott modellt. A rács-kereső algoritmus és annak paraméterei:

```
logistic_regression = LogisticRegression()
param_grid = {
    'C': [0.01, 0.1, 1, 10],
    'penalty': ['l1', 'l2', 'elasticnet'],
    'solver': ['liblinear', 'lbfgs'],
    'fit_intercept': [False, True],
    'class_weight': [None, 'balanced']
}

grid_search = GridSearchCV(
    logistic_regression,
    param_grid,
    cv=5,
    error_score=-9999,
    scoring=('f1', 'accuracy'),
    refit=False
)
```

A fenti paramétereket úgy próbáltam megválasztani, hogy a lehető legátfogóbbak legyenek, illetve hogy kompatibilisek legyenek a megoldandó problémával. Mindazonáltal, mivel így is kialakultak inkompatibilis paraméter párosítások, a -9999 hiba pontszámot állítottam be az eljárás számára, vagyis ha nem kompatibilisek a paraméterek és emiatt hibára futna, akkor a pontszámnak ezt az értéket állítja be.

Ennél a rácskeresésnél nem csak egy pontozási metrikát állítottam be, hanem kettőt

is, mégpedig az $F1$ illetve a pontosság metrikákat. Azért esett erre a metrika-párra a választásom, mivel bináris osztályozási feladatoknál ezek a leghelyénvalóbbak, mivel biztosítják a modell pontosságát és azt is, hogy jól teljesítsen új, korábban nem látott adatokon is. A rácskereső algoritmus a következő paramétereket találta optimálisaknak:

```
{'C': 0.1, 'class_weight': 'balanced', 'fit_intercept': True,
'penalty': 'l2', 'solver': 'lbfgs'}
```

Szélsőséges esetekben a két metrika két különböző paraméterhalmazt határoz meg, viszont esetemben mindkét pontozási metrika a fenti paramétereket eredményezte optimális megoldásként, melyeket alkalmazva a következő pontszámokat produkálta: $\text{mean_test_f1} = 0.94$ és $\text{mean_test_accuracy} = 0.93$.

7.2. Teljesítmény

A modell teljesítményénél több tényezőt is figyelembe vettem, mint például a pontosság, a bináris kereszt-entrópia, a ROC görbe⁸ és az AUC⁹ érték. A pontosságra vonatkozó eredményeket az 5. táblázatban lehet megvizsgálni.

	Validációs adathalmazon	Teljes adathalmazon
Pontosság	92%	98%
Bináris kereszt-entrópia	0,18	0,06
AUC	0,92	0,98

5. Táblázat. Az osztályozó modell teljesítménye

Ebben az esetben is szétválasztottam a teljes adathalmazt tanító és validációs halmazokra, melyek aránya ismételten 80%–20% volt, vagyis az adatok 80%-a a tanításra volt felhasználva, a maradék 20% pedig a modell teljesítményének mérésére.

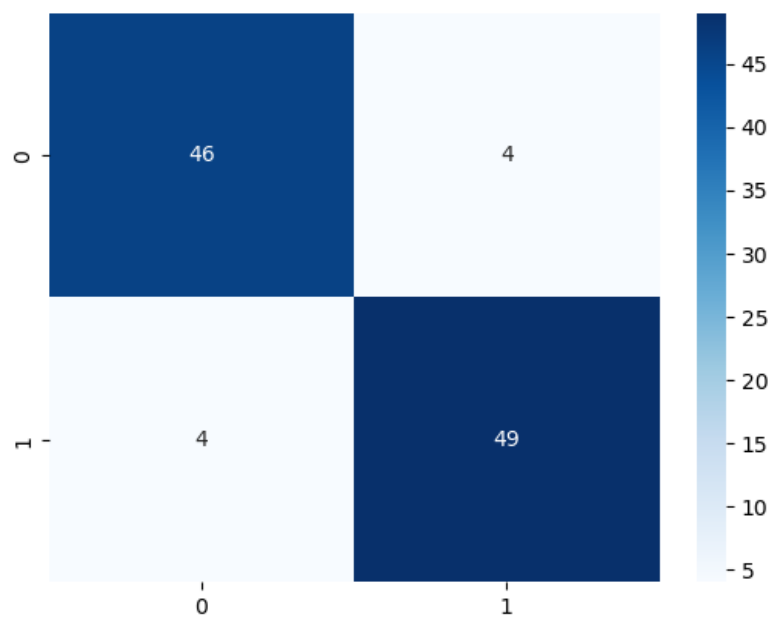
Tévesztési mátrixot is készítettem a validációs adatok felhasználásával, mely a 17. ábrán látható. A validációs adatokon alapuló ROC görbe pedig a 18. ábrán látható.

7.3. Példák

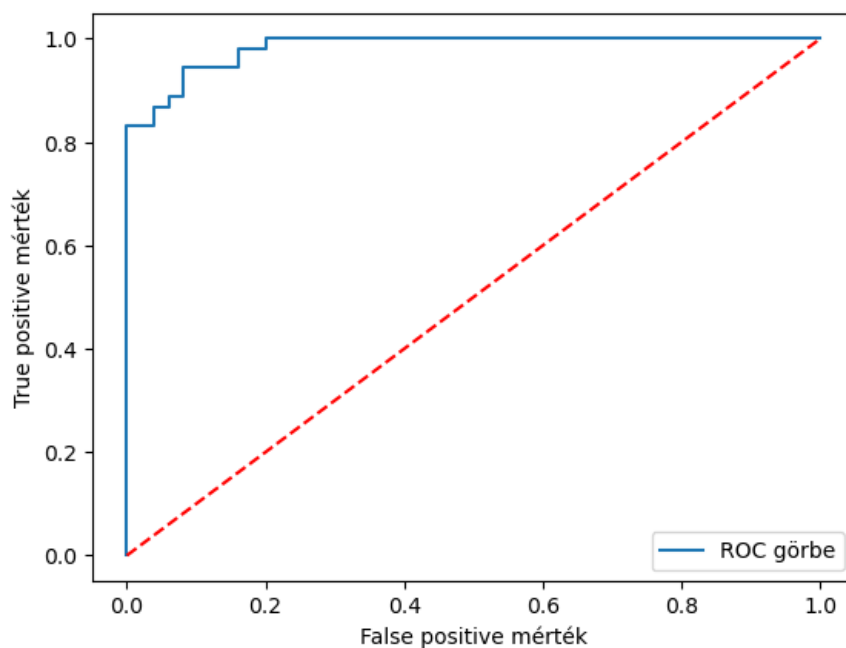
Ebben az alfejezetben pár képpel szeretném illusztrálni mindkét kategóriát, vagyis egy adatminőségében jó lemezt és egy rosszat is, hogy összevethetőek legyenek egymással.

⁸Receiver Operating Characteristic Curve

⁹Area Under the ROC Curve



17. ábra. A validációs adatokon alapuló tévesztési mátrix



18. ábra. Az osztályozó modell ROC görbéje

A 19. ábrán a jó és rossz adatminőségű kategóriákba eső lemezek karctérképei láthatók egymás mellett. Mint látható, a jó minőségű lemezen alig van karcolás, míg a rossz minőségű tele van felületi sérülésekkel.



(a) Jó minőségű lemez



(b) Rossz minőségű lemez

19. ábra. Jó és rossz adatminőségű lemezek kartérképei

8. Összefoglalás

A kutatás záró gondolataként szeretnék szót ejteni néhány érdekes és fontos következtetésről, melyeket az alábbiakban fogok tárgyalni. Ezen következtetések kialakításához a teljes adathalmazt használtam fel.

Két oldalról fogom továbbá vizsgálni az optikai lemezek és azok állapota közötti kapcsolatot: az **adatminőséget** és a **vizuális reprezentációt** figyelembe véve.

Az adaminőségre vonatkozó következtetések megalkotásához a kiterjesztett adathalmazt használtam fel kiindulásként, s így nyertem ki a 6. táblázatban olvasható megfigyeléseket.

A vizuális különbségek vizsgálatához pedig egy egyszerű eljárás-sorozatot implementáltam, amely az összes karctérkép feldolgozásával a következő adatokat nyeri ki:

- **Maximális, minimális és átlagos karcterület(ek):** a lemezeken található egyes karcolások területének maximuma, minimuma és átlaga.
- **Maximális, minimális és átlagos karckerület(ek):** a lemezeken található egyes karcolások kerületének maximuma, minimuma és átlaga.
- **Karcolások száma:** átlagosan hány darab karcolás van a lemezeken.
- **Karcolások lefedettsége:** egy $[0, 1]$ intervallumbeli szám, amely a karcolások összegzett területének és a lemez teljes területének arányát jelenti.

Fontos hozzátennem, hogy ez csak metrikák kinyerésére lett elkészítve, az osztályozó modellel nem használható, mivel nagyon rossz eredményt produkálna. Ennek oka az, hogy a jellemzők manuális kinyerése közel nem olyan optimális, mint a konvolúciós rétegekkel történő kinyerés.

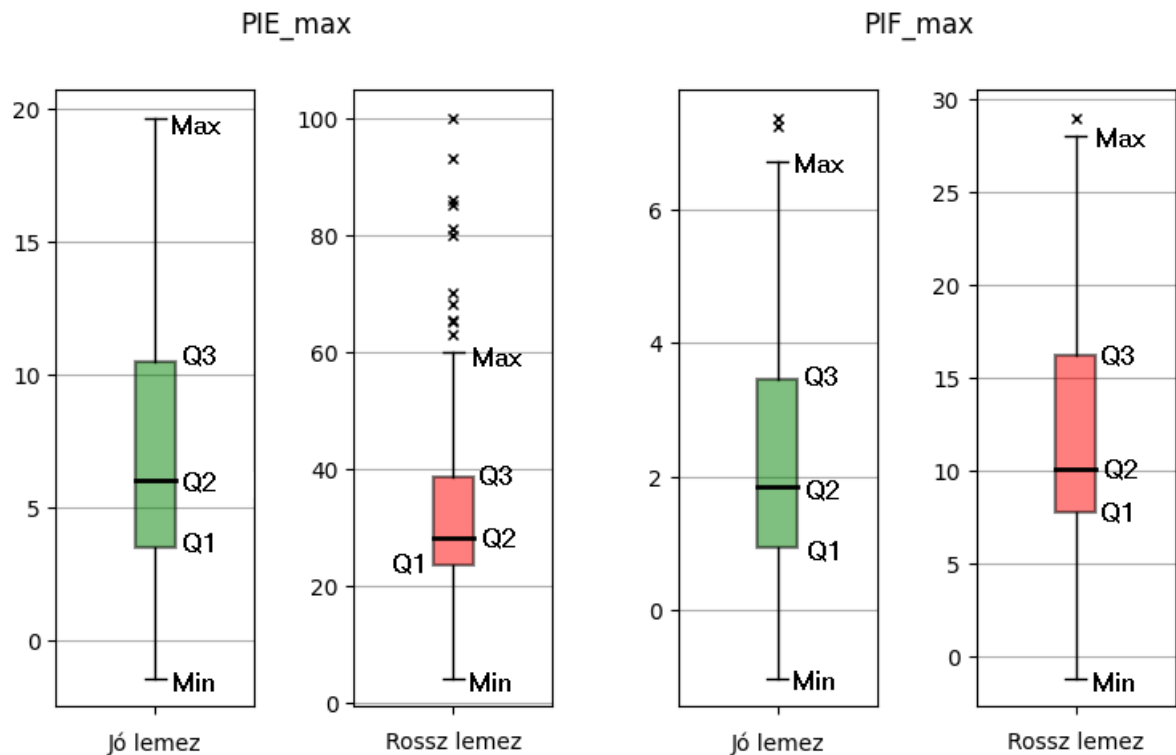
8.1. Adatminőségen alapuló következtetések

Összefoglalásként először szeretnék szót ejteni az adatminőségi halmaz intervallumairól, tehát hogy mely intervallumba tartoznak a jó illetve rossz lemezek. Intervallumok tekin-

tetében mindkét célváltozót figyelembe vettem, vagyis a **PIE_max**-ot és a **PIF_max**-ot is. Ezen statisztikák értékeit a 6. táblázat tartalmazza.

	PIE max intervallum	PIF max intervallum
Rossz lemezek	[4.06, 93.0]	[3.02, 29.0]
Jó lemezek	[0.1, 19.65]	[0.04, 7.37]

6. Táblázat. Adatminőségi határok



20. ábra. A **PIE_max** és **PIF_max** dobozábrája

A fenti értékeket használva meg lehet állapítani, hogy egy lemez adatminőségét tekintve mennyire van jó avagy rossz állapotban. Fontos kiemelni azt is, hogy ezen értékek az általam mért tendenciákat hivatottak reprezentálni, hiszen minden esetben előfordulhatnak szélsőséges értékek.

8.2. Vizualizáción alapuló következtetések

A karcolások tulajdonságai (például méret, terület) és a lemez minősége (jó vagy rossz) közötti kapcsolatot szemlélteti a 7. táblázat.

Fontos megjegyezni, hogy a táblázatban látható értékek az utolsó sor kivételével pixelben értendőek. Az utolsó előtti sor értékei darabszámok, az utolsó soréi pedig arányok ($0,6 = 6\%$).

	Jó lemez	Rossz lemez	Formula
Maximális karcterület	96650,8	260806,32	$\max(T_1^k, \dots, T_{n_k}^k)$
Minimális karcterület	0,55	0,51	$\min(T_1^k, \dots, T_{n_k}^k)$
Átlagos karcterület	2029,39	2789,42	$\frac{1}{n_k} \sum_{i=1}^{n_k} T_i^k$
Maximális karckerület	4367,09	10497,71	$\max(K_1^k, \dots, K_{n_k}^k)$
Minimális karckerület	3,61	3,45	$\min(K_1^k, \dots, K_{n_k}^k)$
Átlagos karckerület	181,09	207,27	$\frac{1}{n_k} \sum_{i=1}^{n_k} K_i^k$
Karcolások száma	291,22	473,13	n_k
Karcolások lefedettsége	0,06	0,13	$(\sum_{i=1}^{n_k} T_i^k)/T$

7. Táblázat. A karcolások statisztikái

Adott a jó lemezek halmaza, melyet jelöljön D_1 , illetve a rossz elemek halmaza, melyet pedig jelöljön D_0 , legyenek továbbá $n_1 = |D_1|$ (a jó lemezek száma) és $n_0 = |D_0|$ (a rossz lemezek száma).

A táblázat formuláiban T_i^k és K_i^k az i -edik jó ($k = 1$)/rossz ($k = 0$) lemezen lévő karcolások területének illetve kerületének összegét jelentik, ahol $k \in \{0, 1\}$ és $i = 1, \dots, n_k$, T pedig egy lemez teljes területe.

Mint azt látni is lehet (lásd a piros számokat a táblázatban), a jó lemezek és rossz lemezek között a területnél, kerületnél, a karcolások számánál és lefedettségénél vannak nagyobb eltérések az adatokban. Így azt a következtetést lehet levonni, hogy leginkább a **karcolások területe, kerülete, mennyisége és lefedettsége** számít az adatminőség eldöntésekor.

8.3. Eredmények értékelése

Kijelenthetem, hogy a kutatást rendkívüli eredményekkel zárhattam, mivel az általam készített kompozit modell közel 98%-os pontossággal képes meghatározni egy adott lemezről annak milyenségét (használható vagy használhatatlan). Továbbá sikerült azt is elérnem, hogy a modell jó eredményt adjon korábban nem látott adatokon is. Ezt a teljesítményt hivatott illusztrálni a bináris kereszt-entrópia, melynek értéke a validációs adathalmazon 0,18.

8.4. Továbbfejlesztési lehetőségek

A továbbfejlesztési lehetőségek között szerepel a nagyobb adathalmaz megalkotása és a modell újratanítása. Mint azt említettem is, minnél nagyobb és dúsabb az adathalmaz, a modell annál pontosabb is. Az adathalmazt növelve lehetőség lenne konvolúciós neurális háló használatára is, mivel már lenne elég adatom ahhoz, hogy ez jobb eredményt adjon a dolgozatban bemutatott modellhez képest.

Véleményem szerint egy jóval nagyobb adathalmaz megalkotásához elengedhetetlen lenne egy olyan szerkezet, mely képes lenne gyorsabban digitalizálni (befotózni) a lemezeket, mint ahogy azt jelenleg manuálisan meg tudom csinálni. Ehhez lehetne használni 3D nyomtatott és Arduino alkatrészeket (kamera, motor, mikro-kontroller).

Irodalomjegyzék

- [1] O. Slattery, R. Lu, J. Zheng, F. Byers, and X. Tang, „*Stability Comparison of Recordable Optical Discs-A Study of Error Rates in Harsh Conditions*,” *J Res Natl Inst Stand Technol*, vol. 109, pp. 517–524, Oct 2004.
- [2] C. Andrade, „*Z Scores, Standard Scores, and Composite Test Scores Explained*,” *Indian Journal of Psychological Medicine*, vol. 43, no. 6, pp. 555–557, 2021. PMID: 35210687.
- [3] M. Shutaywi and N. Kachouie, „*Silhouette Analysis for Performance Evaluation in Machine Learning with Applications to Clustering*,” *Entropy*, vol. 23, p. 759, 06 2021.
- [4] H. Wei, „*How to measure clustering performances when there are no ground truth?*” <https://medium.com/@haataa/how-to-measure-clustering-performances-when-there-are-no-ground-truth-db027e9a871c> [Letöltve: 2023. március].
- [5] OpenCV, „*Canny Edge Detection*.” https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html [Letöltve: 2023. március].
- [6] S. Xie and Z. Tu, „*Holistically-Nested Edge Detection*,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [8] F. Chollet, *Deep Learning with Python*. Manning, 2 ed., October 2021.
- [9] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O’Reilly Media, Inc., 3 ed., November 2022.

Köszönetnyilvánítások

A kutatás létrejöttében rengeteg ember segédkezett nekem, legyen az akár egy apró megjegyzés, tanács vagy sok dolgot átölelő segítségnyújtás.

Először is köszönetet szeretnék mondani a témavezető tanárainknak, **Dr. Jeszenszky Péter** tanár úrnak és **Dr. Ispány Márton** tanár úrnak, akik nélkül nem jöhetett volna létre ez a kutatás. Elképesztően sokat segítettek az ötletelésben, a gondolatmenet kialakításában és elindításában. Az sem elfelejtendő, hogy ők illetve az Információ Technológia Tanszék egy rendkívül erős számítógépet is biztosítottak a kutatási munkára, ami nélkül nem tudott volna megvalósulni ez a dolgozat.

Szeretném megköszönni továbbá a **Debreceni Egyetem Egyetemi és Nemzeti Könyvtárának**, hogy rendelkezésemre bocsátottak olyan DVD lemezeket, melyek a kutatásban felhasznált adatállományok alapjául szolgáltak. Személy szerint meg szeretném köszönni **Jeszenszky Zsuzsannának** a lemezek kölcsönzése kapcsán nyújtott segítségét.

Végül, de nem utolsósorban szeretnék köszönetet mondani a családomnak és barátaimnak, hogy támogattak a mindennapokban és segítették ezen munkámat is.