Gépészmérnöki Kar

Gyártástudomány és -technológia Tanszék

Többlépcsős gépi tanulással támogatott eljárás a gyártási selejtek szűrésére és elemzésére

Giczi Alexandra Laura

Konzulens: Dr. Kovács Edith Alice

2023

Tartalomjegyzék

1. Bevezetés

Napjainkban a gyártás területén már nem csak a nagyvállalatok, hanem a kisebb cégek körében is egyre jobban terjednek el az Ipar 4.0 alapelvei, melyhez köthetőek a digitalizációs, illetve különböző mesterséges intelligencia alapú megoldások, melynek segítségével többek között igyekeznek csökkenteni a selejtek termelésének mennyiségét. Egyre nő a tárolt bemeneti adatok mennyisége, illetve több szenzor kerül fel a gyártósorokra - ezzel megsokszorozva a gyűjtött adatok számát, melyek komplex információkat hordoznak, ugyanakkor ezek rejtve vannak a nagy adathalmazokban.

A gyártásban megjelenő adattudományi feladatok kezelésének során a legjobban elterjedt módszertan a CRISP-DM (Cross-industry standard process for data mining), ennek lépéseit veszik alapul a projektek tervezésének és kialakításának folyamán. Ezt alkalmazva a programozás tekintetében a különböző előrejelző modellek felállítását egy adatelőkészítési lépés előzi meg. Ez többnyire az egész projekt idejének több mint felét veszi igénybe, ugyanis számos adattisztítási feladatra (például hiányzó értékek, kiugró értékek kezelése) kell megfelelő megoldást keresni. Az adattudományi modellek fékezhetetlen terjedésének köszönhetően meghatároztak már számos általános lépést, melyek mentén az adattisztítást szokás végezni, azonban ritkán foglalkoznak azzal az információval, ami az előkészítés során elvész.

Dolgozatom elején ezeknek az információknak a kiaknázásával foglalkozom. Ezután ezt is felhasználva olyan többlépcsős klasszifikációs eljárást vezetek be, aminek eredményeképp lehetőség nyílik selejtes termékek előzetes kiszűrésére, már a végső klasszifikációt megelőzően. A bevezetett eljárást egy döntéstámogató folyamatba építem bele, amely információkat szolgáltat a felhasználónak, aki a visszajelzésekből fel tudja ismerni nem csak a hibás termékeket, hanem fel tudja térképezni a hiba lehetséges okait is. Erre a célra, az itt bevezetett döntéstámogató rendszerbe beépítésre került egy könnyen értelmezhető vizualizációs eszköztár is. Jelen munkám statisztikai és gépi tanulási eszköztárak ötvözésére épül. Az eljárást valós gyártási adatokra tesztelem és bemutatom a beépített lépcsők fontosságát.

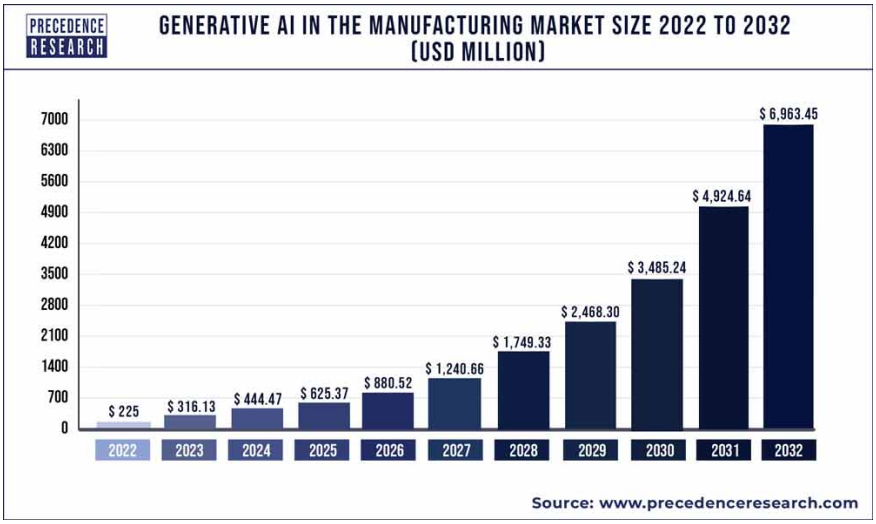
2. Gépi tanulás megjelenése a gyártásban

Az elmúlt évszázadokban óriási fejlődésen ment keresztül a világon a termékek gyártása. Már az első ipari forradalom idején, a 19. században használtak gőzgépeket, a 20. század elején pedig az elektromos energia lendítette meg a fejlődést a tömeggyártás lehetővé tételével a második ipari forradalom során. A harmadikban - a 70-es években - a gyártósorokat már automatizáltság jellemezte, PLC és különböző IT struktúrák használatával, míg elértünk a negyedik ipari forradalomhoz, melyet általában röviden Ipar 4.0-nak hívnak. Ennek 9 alappillérét szokták meghatározni, ez látható az 1. ábrán. [1]

1. ábra. Ipar 4.0 alappillérei. [2]

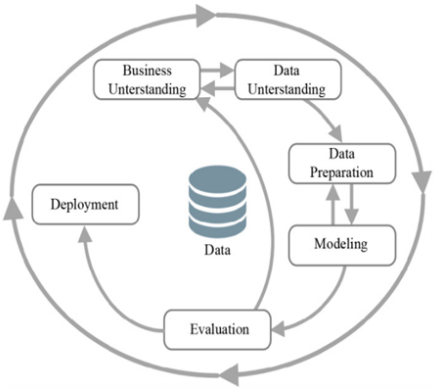
Jelen dolgozat szempontjából fontos az egyik húzó ágazat, a Big Data pillér, melynek során nagy mennyiségű, gyorsan keletkező, változatos adatok elemzésével oldanak meg feladatokat. Jól mutatja a 2. ábra, hogy mennyire fontos már napjainkban is a különböző mesterséges intelligencia alapú megoldások használata, és ennek a területnek óriási növekedést jósolnak a közeljövőben. Ez nem csoda, hiszen az adattudományi feladatok segítségével a gyártást gyorsabbá, rugalmasabbá, fenntarthatóbbá és hatékonyabbá tudják tenni azáltal, hogy magasabb minőségű termékek gyártása válik elérhetővé kevesebb költséggel. Számos irodalom mutat be olyan eseteket, ahol valamilyen, adatokat feldolgozó, majd elemző gépi tanulásos megoldás jelenik meg a gyárban. Ilyenek például a tervezett karbantartást segítő megoldások, amellyel a gépeket hatékonyan lehet használni a meghatározott élettartamukig, csökkentve a karbantartási célú nem tervezett leállások idejét,

illetve a meghibásodások okozta többletköltséget. Fontos felhasználási terület még a hibás termékek (előzetes) felismerése, ezzel biztosítva a megfelelő minőséget, illetve a gyártási paraméterek optimalizációja, mely akár leválthatja a nagy erőforrásigényű és költséges fizikai kísérleteket. A legegyszerűbben megvalósítható feladat pedig egy egyszerű monitorozó rendszer, ahol minden paramétert nyomon követve lehet azonnal beavatkozni, amennyiben valamilyen abnormális eset történik a gyártósoron. [3] [4] [5] [6]



2. ábra. Az AI gyártásban történő alkalmazásának piaci mérete a közeljövőben. [7]

Az Ipar 4.0 széles körben történő elterjedésének köszönhetően manapság számos kifejlesztett keretrendszer teszi lehetővé az adattudományi feladatok lépéseinek végrehajtását. A projektmenedzsmenthez kapcsolódóan kidolgoztak olyan metódusokat, melyekkel könnyebben lehet definiálni az egyes lépéseket, szerepköröket, ezek közül a gyártás területén a legjobban elterjedt a CRISP-DM.



3. ábra. A CRISP-DM módszertan fázisai. [8]

A CRISP-DM (Cross-industry standard process for data mining) módszertan a 3. ábrán is látható módon 6 lépésből áll. Ezek és jelentőségük a következő:

- Business understanding: projekt elindítása, üzleti információk megértése, gyártósor és folyamatok megismerése
- Data understanding: adatok megismerése és megértése
- Data preparation: adatok előkészítése az algoritmus számára szükséges formára
- Modeling: modellezés, megfelelő algoritmus(ok) kiválasztása
- Evaluation: eredmények kiértékelése
- Deployment: algoritmus bevezetése, dokumentációk elkészítése és projekt lezárása

Fontos megjegyezni, hogy ezek nem minden esetben lineárisan következnek egymás után, hanem sokszor 1-1 lépés után új információk derülnek ki és máshogy, optimalizáltabban lehet elvégezni korábbi fázisokat.

Jelen munkám középpontjában az ismertetett fázisok közül az adatelőkészítés, a modellezés és a kiértékelés lesz fókuszban. A következő fejezetben bemutatom az adatelőkészítéshez használt általános lépéseket, melyet minden feladat esetén elvégeznek, utána pedig következik a dolgozat kutatási eleme, melyben bemutatom, hogy a tradicionális előkészítés során milyen információk vesznek el, melyet nem szoktak vizsgálni, illetve, hogy hogyan lehet ezek információtartalmát is felhasználni, mely módszerrel különböző módon erőforrást lehet spórolni. Ismertetem a többlépcsős keretrendszerem klasszifikációs problémamegoldás kezelésére, illetve az egészet összefoglaló döntéstámogató rendszert.

3. Adattisztítási feladatok

3.1. Bevezetés

A projektek során általánosan kimondható, hogy az egyes lépések közül az adatelőkészítés részére megy el a legtöbb idő. Ennek pontos értéke több mindentől függ, a szakirodalomban is eltérőek szerepelnek, az IBM szerint 50-70%-ig terjed, míg az Anaconda nagyszabású éves Data Science riportja szerint a százalékos érték 2020-hoz képest csökkenő tendenciát mutat, és jelenleg 38%. Ez a tendencia a manapság elérhető programozási csomagok segítségével magyarázható, azonban mivel minden adathalmaz egyedi és a bejövő adat általában nagyon változatos hibákat rejt, így egy bizonyos mértéknél nagyobb csökkenés nem várható. [9] [10] [11]

Az adatelőkészítés stádiuma is tovább bontható alfeladatokra. Először is szükséges az adatok összegyűjtése, sokszor többféle módon tárolt halmazok összevonásával. Ezután következik az adattisztítás, mely a következő eseteket foglalja magába:

- duplikált sorok
- hiányzó értékek
- kiugró értékek, anomáliák

A következő lépés a különböző transzformációk elvégzése, mely adattípusok szerint szétválogatva végzendő:

- numerikus adatok normalizálása/standardizálása
- kategorikus adatok kódolása

Az utolsó lépésnek pedig a nagy adathalmazok esetén van jelentősége, de érdemes a kisebbek esetén is elvégezni, ez pedig a redukálás:

- fontos szerepet játszó változók kiválasztása (feature selection)
- sok rekord esetén mintavételezés

Ezután következik a modellezés fázisa és a megfelelő algoritmus kiválasztása. Vannak olyan előbb említett eljárások, melyeket mindenképpen végre kell hajtani a legtöbb algoritmus használatához, azonban vannak olyanok is, melyek csak a szükséges számítási időt és erőforrást csökkentik le, mely ugyanúgy fontos szerepet játszik. Ahhoz, hogy meg lehessen állapítani, melyik lépésnél történhet információ vesztés, először szeretném összegezni, hogy az egyes említett lépések során milyen módszerekkel szokás az adott feladatot megoldani.

3.2. Adattisztítási módszerek lépéseinek megoldási módszerei

Az első lépés a duplikált adatokkal való feladat deifiniálása. Ezen elemek kezelése nagyon fontos, főleg abban az esetben, ha indexként szerepel a duplikáció, hiszen ez nem megengedett. Ezeket a sorokat általában automatikusan törlik, azonban a legtöbb esetben van annak valamilyen oka, hogy duplikációk keletkeznek. Lehet ez egyszerűen valamilyen információs hiba is, de az is előfordulhat, hogy valamilyen többletinformációt hordoz a duplikátumok száma. Emiatt ez egy olyan információ, mely a tradicionális feldolgozás során elvész, ezt érdemes vizsgálni az adatelőkészítés során.

A következő lépés a hiányzó értékekkel kapcsolatos. A legtöbb algoritmus fontos bemeneti követelménye, hogy ne legyen olyan mező az adathalmazban, amely NaN (not a number) vagy NULL értéket tartalmaz (ez indikálja a hiányzó értéket), ezért minden esetben az adatelőkészítés részét képezi a hiányzó értékek kezelése, nagyon kevés esetben térnek el ettől. Erre már többféle módszer is létezik, melyek a következőek: [12] [13]

- hiányzó értéket tartalmazó sor törlése
- hiányzó értékek feltöltése, azaz adatimputáció:
  - mediánnal
  - átlaggal
  - leggyakrabban előforduló elemmel
  - előtte/utána szereplő elemmel
  - egyéb interpolációs technikával

Az első esetben, amennyiben nagy mennyiségű sort érint, a törlés nem lehet opció, hiszen azzal túl sok adat vész el. A második eset alkalmazásakor általában a legnehezebb feladatot az szokta okozni, hogy meghatározzuk, pontosan milyen értékkel is legyen feltöltve a hiányzó adatot tartalmazó mező. Fontos előre tisztázni, hogy annak a hiányzó értéknek nincs-e jelentősége (például két mérőállomás van és mindig a kettő közül csak az egyiken halad keresztül minden termék), hiszen akkor más módon kell eljárni (változók összevonása). Ha nincs előre tulajdonított szerepe, akkor sem megoldás egy egyszerű 0 behelyettesítése egy számérték esetén, hiszen az azt jelentené, hogy egy adott paraméter esetén a mérési érték ennyi. Összegezve tehát ez ismét egy olyan lépés, melynek során információt veszítünk, hiszen az adatimputáció után már nem meghatározható, mely értékek hiányoztak az eredeti adathalmazban.

Ezután következik a kiugró értékek, különböző anomáliák kezelése. Ezeknek a detektálása már nem olyan egyértelmű, mint a fentebb említetteknek, ugyanis többféle megközelítés alapján lehet egy értéket kiugrónak – outliernek – tekinteni. Talán az egyik legelterjedtebbnek mondható az interkvartilisek használata, mivel ez a módszer a nem Gauss-eloszlású attribútumok esetén is működik. Ennek alapja a boxplot ábrázolás, mely esetben a 25 és 75%-os alsó és felső kvartilishez képest 1,5 vagy 3 interkvartilist (a kettő különbségét) kell kivonni/hozzáadni és az ezen kívül esők tekinthetőek kiugró értéknek. Az ehhez tartozó vizuális ábrázolás látható a 4. ábrán, itt pedig az értékek számolásának módja: [12] [13]

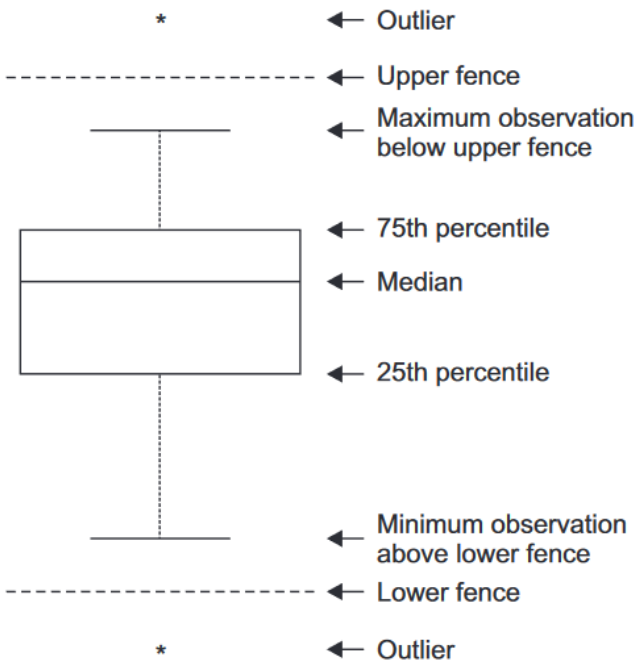
$$Q_1 = \text{quantile}(0.25) \tag{1}$$

$$Q_3 = \text{quantile}(0.75) \tag{2}$$

$$IQR = Q_3 - Q1 \tag{3}$$

$$alsó\_határ = Q_1 - 1.5 \cdot IQR \tag{4}$$

$$fenti\_határ = Q_3 + 1.5 \cdot IQR \tag{5}$$



4. ábra. Kiugró értékek elhelyezkedése. [12]

Miután megállapítottuk, hogy mely sorban mely értékek kiugróak, a következő kezelési módszerek általánosak: [12] [13]

- kiugró értéket tartalmazó sor törlése
- kiugró érték helyettesítése például a legmagasabb/legkisebb nem kiugró értékkel
- más, robosztus módszerrel kapott értékkel történő helyettesítés

Hasonlóan a hiányzó értékeknél említettekkel, az első esetben itt is előfordulhat, hogy túl sok adatot kellene törölni, ami ahhoz vezetne, hogy nehezebb betanítani a modellt, a másik két esetben pedig valamilyen értékkel történő feltöltés után elveszik, hogy melyik mezők tartalmaztak nem megfelelő értéket. Emiatt ezen lépés során is olyan információt vesztünk, melyet érdemes megvizsgálni.

Ezután következnek a transzformációk. A numerikus értékek esetén legfőképpen akkor van szükség erre, ha az egyes változók értéktartományai nagyon eltérőek, ugyanis az hatással van a modell futási idejére. Normalizálás esetén 0-1 közötti értékre alakítjuk át a következő módon: [13]

$$x_{norm} = \frac{x - min}{max - min} \tag{6}$$

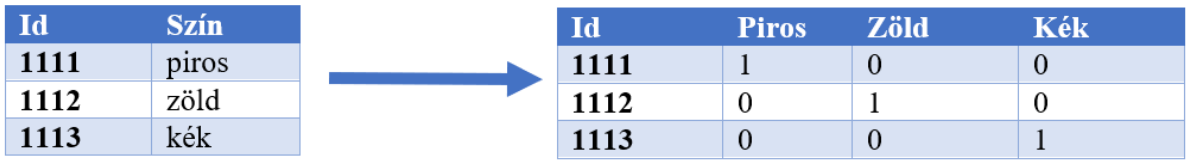
Standardizálás esetén a végső attribútum eloszlására érvényes lesz, hogy a várható értéke nulla, szórása egy legyen. Ez a következő számolással végezhető el: [13]

$$x_{st} = \frac{x - átlag}{szórás} \tag{7}$$

Fontos megjegyezni, hogy ezek alkalmazása érzékeny a kiugró értékekre, hiszen a statisztikai jellemzőket, illetve minimum-maximum értékeket nagyban befolyásolja, ezért az azokkal való műveletet mindenképpen el kell végezni (kivéve, ha más, robosztus technikát alkalmazunk). [13]

A kategorikus adatok kódolására a legtöbb algoritmusnak szüksége van, anélkül hibát fognak dobni és nem fognak lefutni. Ezen kódolások során mindig numerikus értéket képzünk belőlük, de ennek több módja is van. A két legelterjedtebb az ordinal encoding és a one hot encoding. Az első esetben az oszlopban minden egyedi elemhez egy egész szám rendelődik (pl. színek esetén piros-1, zöld-2, kék-3). A negatív oldala az algoritmusnak, hogy a későbbi modell azt feltételezheti, hogy bizonyos rangsorolás van az értékek között (ez igaz a számértékekre, hiszen 2 nagyobb mint 1,

de sok esetben nem igaz a hozzá társított kategorikus adatokra). Ezt a hibát küszöböli ki a one hot encoding. Ekkor az adott oszlop eltávolításra kerül és minden, az ott fellelhető egyedi érték számára létrejön egy új oszlop, melyben csak ott szerepel 1-es érték, ahol az adott jellemző igaz a rekordra. Ezt a műveletet szemlélteti az 5. ábra. [13]



Id	Szín
1111	piros
1112	zöld
1113	kék

Id	Piros	Zöld	Kék
1111	1	0	0
1112	0	1	0
1113	0	0	1

5. ábra. One hot encoding vizualizációja.

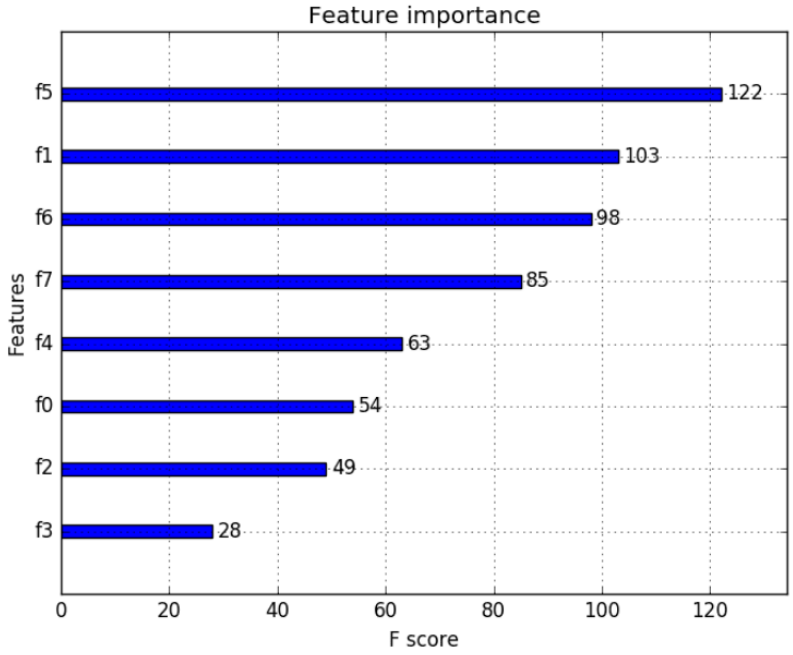
Ahogy látható itt információt nem vesztünk, hiszen minden elem benne marad az adatbázisban csak más formában, az értékek közötti távolság pedig megmarad, csak más intervallumon belül.

Az utolsó lépés pedig az adatok és attribútumok redukciója. A mintavételezésre egyre kevesebb szükség van, hiszen már megjelennek olyan programok, melyekkel a felhőben/távoli szerveren lehet a számításokat elvégezni, így csupán olyan sorok szűrését érdemes végezni, melyek egyértelmű hibából adódnak (gépi hiba miatt történt esetek, esetleges teszt jellegű adatok). Ez a lépés ugyan információvesztéssel jár, de lehet a mintavételezést úgy elvégezni, hogy bizonyos számú sorra futtatjuk a modellt és $n+1$ vagy $n+i$ eltolással újra képzünk adathalmazokat, melyekre ismét megnézzük az algoritmus eredményét.

A fontos szerepet játszó attribútumok kiválasztása is számos módszerrel történhet. A legegyszerűbb a korrelációk vizsgálata és azok kiszűrése, melyek bizonyos szintél (95%) jobban korrelálnak, hiszen ezek nem hordozna többletinformációt. Ennél bonyolultabb módon működnek a következő eljárások:

- Attribútumok fontosságának (feature importance) ábrázolása, mely segítségével el lehet dönteni, mely magyarázó változók járulnak hozzá az eredményhez a legjobban. Ilyet lehet létrehozni például döntési fákra épülő algoritmusok segítségével, ezt mutatja a 6. ábra.
- Kisebb dimenziós térbe való transzformálás, amely a változók közötti redundanciát kezelni tudja, például a PCA (főkomponens analízis).

Ezekben az esetekben is vesztünk információt, de előzetesen meg van szűrve, hogy csak olyat veszítsünk, mely nem járul hozzá szignifikánsan a kimenet jóslásához, maximum nagyon kismértékben a többi változóhoz képest.



6. ábra. Feature importance. [14]

Ezt a részt az alábbi következtetéssel zárom: a szokványos eljárások a fent említett lépésekkel foglalkoznak és nem veszik figyelembe az adatelőkészítés során azokat a rejtett információkat, amelyek ezáltal könnyen elvesznek az alábbi esetekben:

- duplikált sorok
- hiányzó értékek
- kiugró értékek

Dolgozatom egyik fontos eleme, hogy az általam bevezetett eljárásba beépítek olyan kritériumokat, amelyek alapján bizonyos összefüggéseket kiszűrök még mielőtt az adattisztítás szokványos lépéseit követnénk.

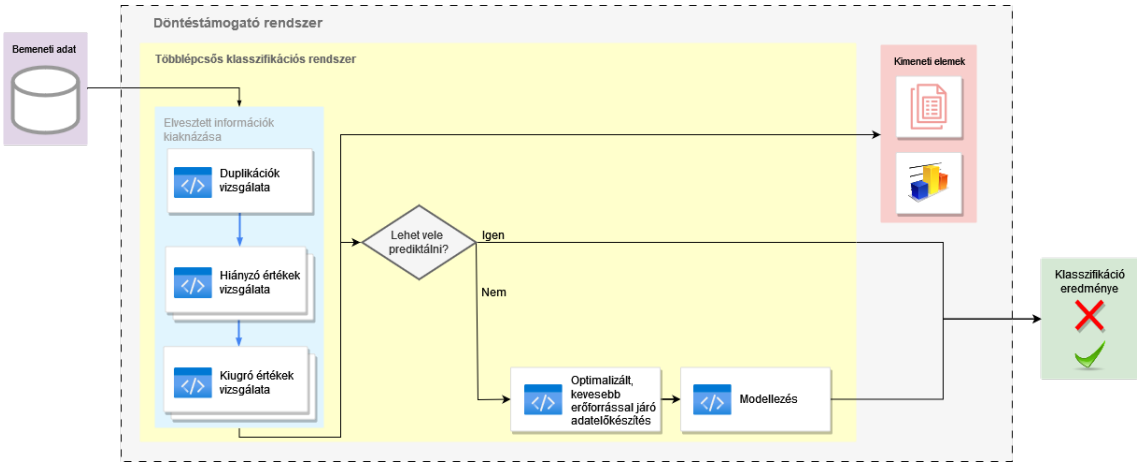
4. A megvalósított rendszer

4.1. Bevezetés

Az általam megvalósított rendszer és annak elemeinek létrehozása során minden függvény úgy készült, hogy általánosítható legyen, azaz bármilyen, excelből beolvasott adathalmaz esetén, amennyiben bináris klasszifikációt szeretnénk megvalósítani, a módszerem használható. Azért bináris klasszifikációra készítettem el a keretrendszert, mivel a gyártásban általános problémára keresem azt az erőforrásokra jobban optimalizált megoldást, ami nem más, mint a selejtek csökkentése. Egy termék vagy megfelel vagy selejtes, ez a klasszifikáció végső kimenete, melyet szeretnénk előre megjósolni. A rendszer felügyelt tanulási csoportba sorolható, ugyanis a tanító és teszt adatok címkézettek, az adathalmaz mindegyik eleméről tudjuk, hogy hibás vagy sem.

Az adatok bemeneti követelménye csupán a következő: az indexen kívül, mely egy egyedi azonosító csak azon numerikus, illetve kategorikus adatok szerepeljenek az oszlopok között, amelyeket szeretnénk belevenni a modellbe, tehát az olyan paraméterek, ahol az értékek például hosszú, nem egyedi szövegek, azok legyenek előre kiszűrve. Ezen paraméterek legyenek átnevezve úgy, hogy a típusától függően *numerical_sorszám* vagy *categorical_sorszám* legyen. Legyen mindenképp egy olyan oszlop, mely tartalmaz időbélyeget *timestamp_0* néven. A klasszifikáció eredményeként jelzett változó esetében pedig szükséges a két csoportot 0 és 1 számmal helyettesíteni, ahol 1 jelöli a selejtes, 0 a hibátlan terméket.

A megvalósított rendszer folyamatábrája látható a 7. ábrán, ennek elemeit fogom bemutatni a következő alfejezetekben.



7. ábra. A teljes megvalósított rendszer.

4.2. Adatelőkészítés rejtett információk kiaknázásával

4.2.1. Duplikációk, hiányzó értékek és kiugró értékek száma mindegyik termék esetén

A következőkben tárgyalt módszereket egymás után végrehajtva valósul meg a többlépcsős rendszer első pár lépése. Az analízis során különböző statisztikai és gépi tanulási eszközöket alkalmazok. Ezek eredményét több csoportra lehet majd osztani a végeredmény alapján, melynek segítségével meg lehet "jósolni" az eredményt, jelzőrendszerként lehet használni a rendszerben vagy pedig a későbbi adatelőkészítési lépésekkel kapcsolatban kaphatunk segítő információt, mellyel erőforrást tudunk csökkenteni.

Az analízis során használni fogom a Cramer-mutatót, így ezt előre szeretném ismertetni. Ez egy olyan statisztikai mutató, mely két változó közötti korrelációt vizsgálja. Az eredménye mindig a $[0,1]$ halmazból vett érték, 0 amennyiben a két változó között semmilyen asszociációs kapcsolat nincs, és 1, amennyiben a két változó teljes mértékben korrelál, de bármely értéket felvehet a kettő között. A számolások alapját a χ^2 próba adja és a következő egyenlettel számolható: [15]

$$V = \sqrt{\frac{\chi^2}{N \cdot min(k-1, r-1)}}, \tag{8}$$

ahol N a megfigyelések, k a kontingenciatáblázat oszlopainak és r a sorainak száma. Ez megfelel az adott feladatnak, ugyanis diszkrét, illetve kategorikus változók kapcsolatát szeretnénk elemezni, és ennél nem okoz gondot, hogy több sor van, mint oszlop (a minimumot veszi a nevezőben).

A rendszer legelső lépcsőjeként azt vizsgálom, hogy a duplikációk száma, tehát amikor adott azonosítóval több sor is szerepel, esetleg előjelzi-e a selejtes termékeket. A második és harmadik lépcsőként a hiányzó és kiugró értékek (IQR módon detektálva) esetében már kétféle módszer lesz alkalmazva. Az elsőben, a duplikációk számához hasonlóan azt fogom elemezni, hogy a hiányzó és a kiugró értékek száma megfelel-e ugyanerre a predikciós követelményre. Mivel ezekben az esetekben hasonló módon történik a vizsgálat, így létrehoztam függvényeket, melyeket mindegyik esetben használni tudok. A lépéseket összefoglalóan mutatja be az Algoritmus 1.-ben leírt pszeudokód. Már az elején külön válogatva szerepelnek a bináris klasszifikáció eredményei (selejt, nem selejt), és mindegyikre lefuttatjuk a különböző függvényeket. Ezek minta eredményei láthatóak a magyarázatokkal együtt a következő ábrákon.

Algoritmus 1: Duplikációk, hiányzó és kiugró értékek számának vizsgálata

módszerek = [duplikációk, hiányzó értékek, kiugró értékek]

kimenet = [selejt, nem selejt]

foreach $i \in$ *módszerek* **do**

  foreach $j \in$ *kimenet* **do**

    i módszerbeli összegzés minden j-beli egyedi azonosítóra;

    dictionary_1, dictionary_2, norm_dictionary inicializálása;

    Function *Előfeldolgozás I. :*

      dictionary_1: keys $\rightarrow$ j egyedi azonosító, values $\rightarrow$ összegzés eredménye;

    end

    Function *Előfeldolgozás II. :*

      dictionary_2: keys $\rightarrow$ j összegértékei, values $\rightarrow$ hány db. szerepelt belőlük;

    end

    Function *Normalizálás :*

      norm_dictionary dictionary_2 normalizálása az összes elemmel;

    end

  end

  Function *Ábrázolás :*

    plot(két normalizált dictionary ábrázolása hisztogramon i módszerhez);

  end

  dataframe létrehozása két oszloppal: összeg és kimeneti változó;

  Function *Cramer :*

    Cramer-mutató kiszámítása;

  end

  Function *Kimenet :*

    if *Cramer-mutató > 0.95* **then**

      return Prediktálhatunk belőle;

    end

    else if *Cramer-mutató <= 0.95 and Cramer-mutató > 0.5* **then**

      return Figyelmeztető jelként beépíthető;

    end

    else return Nincs releváns összefüggés;

  end

end

Az algoritmust a következő minta adathalmazon mutatom be a hiányzó értékek esetére érvényesen, de ugyanígy történik a másik két eset kezelése:

uniquepart_id	numerical_1	numerical_2	categorical_1	categorical_2	y
111	1.0	12.0	A	AA	0
112	2.0	10.8	B	BB	0
113	3.0	11.1	NaN	NaN	1
114	NaN	14.8	NaN	NaN	1

8. ábra. A kiindulási adathalmaz.

Az első lépés a hiányzó értékek összesítése minden sorban külön-külön a két csoportra, majd olyan adatstruktúra létrehozása, melyben minden egyedi azonosító mellett szerepel a hiányzó értékek száma. Hasonló adatstruktúrában összegyűjteni az előző második oszlopából az egyedi értékeket, és hogy melyikből mennyi szerepelt. Ezután ezeket az összes elem mennyiségével normáljuk, hogy összehasonlítható legyen az általános esetben kevesebb selejt és a több megfelelő termék aránya.

```
1 dict_scrap_mv_count
{2: 1, 3: 1}
```

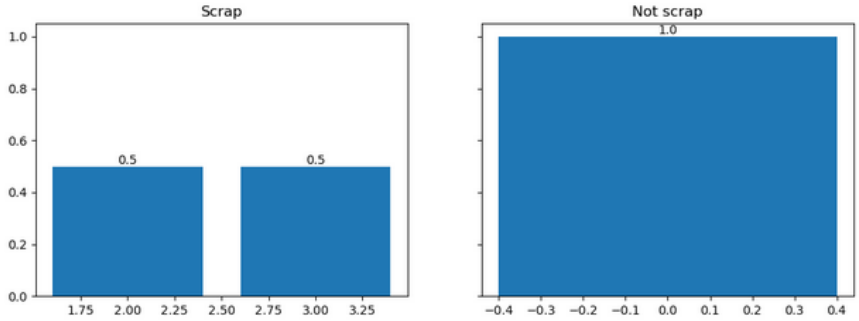
```
1 dict_scrap_mv_count_norm
{2: 0.5, 3: 0.5}
```

```
1 dict_not_scrap_mv_count
{0: 2}
```

```
1 dict_not_scrap_mv_count_norm
{0: 1.0}
```

9. ábra. Egyedi hiányzó elemszámok és azok mennyisége, illetve a normált lista.

Ezen normált értékek ábrázolásával kirajzolhatunk két hisztogramot, melyek vizuálisan is visszajelzést adnak az eloszlásról:



10. ábra. Hisztogram a normált értékekkel.

Miután ezt elvégeztük a selejtekre és nem selejtekre, a két adatstruktúrát egybe összesítve elvégezzük rajta a Cramer-mutató számolását, hogy megtudjuk, van-e összefüggés a hiányzó értékek száma és kimenet között. Itt meg lehet állapítani egy szintet, amely felett megfelelőnek tekintjük a két változó korrelációját. Ebben a rendszerben ez 95%-nál szerepel, ekkor el lehet végezni a predikciót, ennek pszeudokódja látható az Algoritmus 2. részben. Ez minden új elem esetén megnézi, hogy a hiányzó értékek száma szerepel-e a már ismert értékek között a selejtes vagy nem selejtes elemeknél, és abba a csoportba osztja be, melyhez képest a távolsága minimális. Amennyiben mindkét csoporttól való távolsága azonos, automatikusan selejtnek lesz besorolva. Ha a mutató értéke a 95%-ot nem éri el, de egy bizonyos szintnél (legyen 50%) továbbra is magasabb, akkor érdemes lehet figyelembe venni a korrelációt és a gyártósorba beépíthető egy jelzőrendszer, mely minden elem esetén real-time vizsgálja a hiányzó elemek számát és jelezhet, ha megadott mennyiségben kerül az egyedi azonosítóval rendelkező termék regisztrálásra az adatbázisban. Ha ezek egyike sem teljesül, akkor kijelenthető, hogy nincs összefüggés a két változó között.

Az utolsó esetben duplikációk esetén nem vesztünk információt, ha töröljük a duplikált sorokat. A másik két módszernél azt jelenthetné, hogy bármilyen, 3. fejezetben említett módon lehetne kezelni a hiányzó és kiugró értékeket, hiszen itt is ugyanaz jellemzi a selejtes és nem selejtes termékeket, azonban fontos még figyelembe venni, hogy annak is van információtartalma, hogy melyik oszlopban találhatóak meg ezek az értékek, így érkezünk el a már korábban említett második lépcsőhöz ezen két adatelőkészítési lépés esetén. Ezt a két allépcsős rendszert jelezte a 7. ábrán a dupla doboz.

Algoritmus 2: Prediktálás a duplikációk, hiányzó vagy kiugró értékek számából

Function *Minimum távolság számítás :*
min_1 a selejtekhez tartozó összegértékektől való minimum távolság;
min_2 a nem selejtekhez tartozó összegértékektől való minimum távolság;
if *min_1<min_2* **then**
return selejt;
end
else if *min_1>min_2* **then**
return nem selejt;
end
else return selejt;
end

4.2.2. Hiányzó és kiugró értékek elhelyezkedése az adathalmazban

Ehhez a lépéshez át kell alakítani az adathalmazunkat. Bevezetek bináris változókat a következőképpen: a hiányzó értékek esetén a pozitív azt jelenti, hogy nincs az adott mezőben bevitt érték, negatív, hogy van. A kiugró értékek esetén pozitív, ha kiugró az érték (ugyancsak IQR metódussal számolva), negatív, ha nem. Az új adat kialakításának módja:

- minden olyan mező értéke, mely pozitív, lecserélésre kerül 1-es értékre
- minden olyan mező értéke, mely negatív, lecserélésre kerül 0-ás értékre

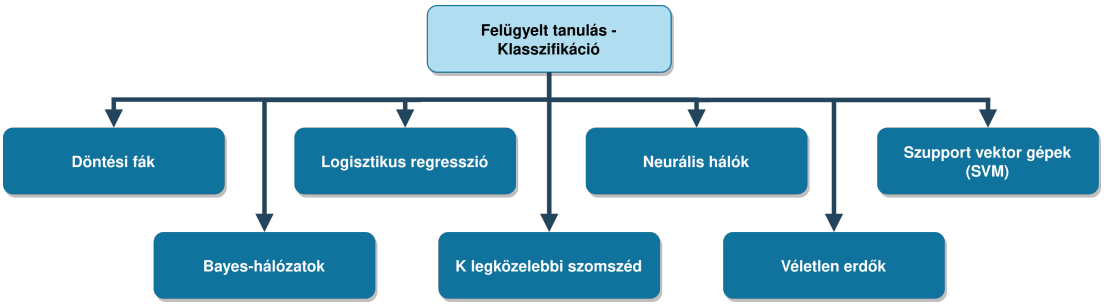
Az előző példában bemutatott minta adathalmaz esetén ez a hiányzó értékekre a következőképpen néz ki:

uniquepart_id	numerical_1	numerical_2	categorical_1	categorical_2	y
111	0.0	0.0	0	0	0
112	0.0	0.0	0	0	0
113	0.0	0.0	1	1	1
114	1.0	0.0	1	1	1

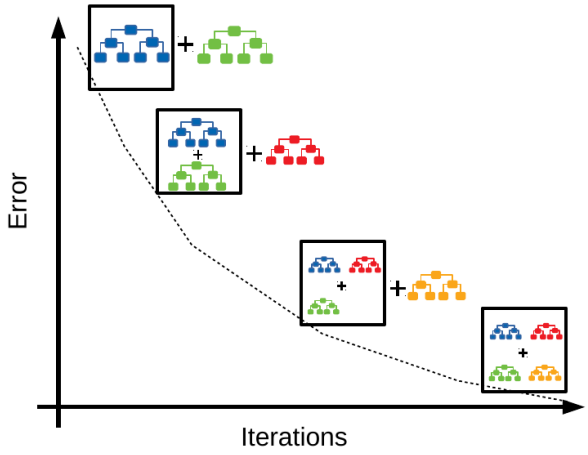
11. ábra. Értékek cseréje után kapott adathalmaz.

Erre már alkalmazhatunk általános klasszifikációs gépi tanulási algoritmusokat, ahol a bemeneti információ a numerikus és kategorikus oszlopok halmaza, a kimenet pedig a predikció eredménye. Ilyen algoritmusok láthatóak a 12. ábrán. Ezekből került kiválasztásra általam 3, amelyeknek eredményeit összehasonlítva a legjobban teljesítőt alkalmazom a predikcióra. Ez a három modell a következő: döntési fa alapú eljárások, támasztó vektor gép (SVM) és K legközelebbi szomszéd (KNN). Az elsőhöz a Catboost nevű csomagot használom a másik kettőhöz pedig a Scikit-learn csomag függvényeit, kiértékelő parancsait.

A Catboost egy olyan nyílt forráskódú könyvtár, mely döntési fákon végez gradiens boosting eljárást, mely iteratív módon több döntési fát generál le, ezzel csökkentve a hibákat. Mindegyik külön becslést végez és a későbbiek az előzőek hibáit próbálják javítani. Ez látható a 13. ábrán. Az egyik legnagyobb előnye a többi algoritmushoz képest, hogy egyedüliként kezeli a kategorikus változókat úgy, hogy semmilyen előfeldolgozásra nincs szükség (nem kell őket numerikus értékekké konvertálni). A függvény paramétereinek optimalizálása nélkül is jó minőségű eredményt ad a többi gradiens boosting algoritmushoz képest is és a futása is gyors. [16] [17]

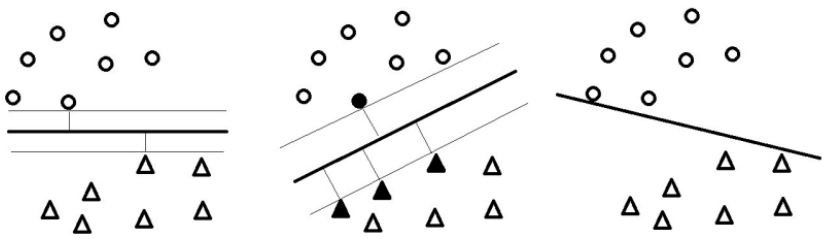


12. ábra. Felügyelt tanulási algoritmusok klasszifikációra.



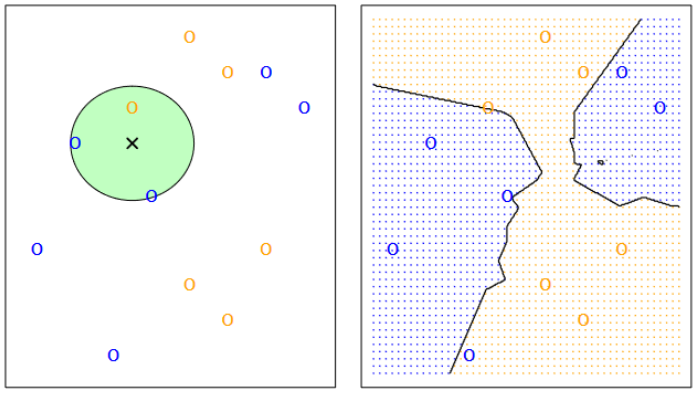
13. ábra. Gradiens boosting algoritmus. [17]

Az SVM-ek esetén a kategorikus string típusú adatokat már szükséges kódolni, ehhez a már említett ordinal encoding módszert választottam, hogy ne legyen az adathalmaz túl nagy a sok kategorikus adat esetén. A megoldás során az algoritmus szeparációt végez, azaz hipersíkkal választja el a kimeneti két osztály elemeit úgy, hogy az elválasztás mértéke a lehető legnagyobb legyen, ahogy ez látható a következő ábrán két dimenzióra. A cél tehát a két vékony vonal közti távolság maximalizálása, így az ábrák közül a középső lesz az SVM megoldása a feladatra. [18] [19]



14. ábra. Szeparáció elvégzése többféle módon, középső ábra az eredménye az SVM-nek. [18]

A KNN algoritmus azon alapszik, hogy az új elemeknél megnézi, hogy a bemeneti paraméterként adott k legközelebbi (mérték például az euklideszi távolság) elem között mely kimeneti csoport van többségben és abba fogja sorolni. A $k = 3$ számra látható ez vizuálisan a 15. ábra bal oldalán, ahol a fekete x-szel jelölt új elem esetén a két kék és egy sárga elem vizsgálatával állapítódik meg, hogy ez a kék csoportjába fog kerülni, hiszen azok vannak többségben. A jobb oldalon pedig az látható, hogy mely területeken milyen besorolást kapna egy új elem. Ezen algoritmus sem tudja a kategorikus string típusú adatokat kezelni, így itt is ordinal encoding módszerrel történik ezek előfeldolgozása. Az algoritmus előnye, hogy könnyen implementálható és megérthető a mögötte lévő logika, azonban érzékeny a k paraméter megválasztására, ezen paramétert érdemes hangolni a futások során. [18] [19]

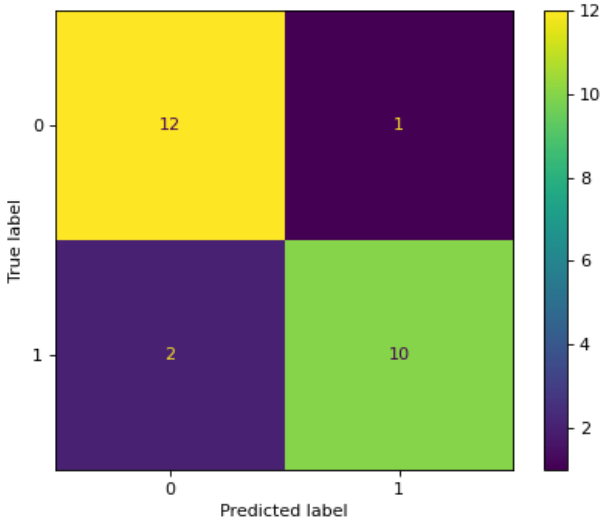


15. ábra. KNN algoritmus $k = 3$ szomszédra, illetve a területi eloszlások új adat esetén. [19]

Mindegyik modell alkalmazásához azonos lépéseket kell elvégezni, ezek pszeudokódja látható az Algoritmus 3. részben. Elsőként az adathalmazt kettéválasztjuk X bemeneti és y kimeneti részre. Ezután mindkét változót szétválasztjuk (X_{OK}, X_{NOK} és y_{OK}, y_{NOK}), hogy a tanító és teszt adatok között is azonos arányban szerepeljenek a kimeneti változók. Amennyiben a selejtek és nem selejtek száma nagyon eltérő, akkor ezt érdemes orvosolni oly módon, hogy a nagyobb méretű halmazt redukáljuk a kisebb méretűnek maximum tízszeresére. A további lépéseket pedig partícionálással végezzük el, hogy minden adat részt vegyen a predikcióban. Ezen egyéni intervallumok eredményeit átlagolva adódik ki majd a végső eredmény. Ha megtörtént a szétválasztás, mindegyikből képzünk tanító és teszt adatokat 80-20% arányban, majd újra egyesítve az OK és NOK változókat és az esetlegesen szükséges ordinal encoding transzformációt elvégezve hattatjuk a kiválasztott algoritmust a tanító adatokra, az eredményül kapott modell segítségével pedig a validáljuk a modell működését a teszt adatokon.

A modellek kiértékelésére többféle módszer terjedt el. A rendszeremben a tévesztési mátrixot használom, illetve az abból számolható metrikákat, melyek a következőkben láthatóak. A Scikit-learn által függvényként használható mátrix látható a 16. ábrán. Ennek előnye, hogy jelzi, melyik mező milyen valódi és milyen prediktált értékkel rendelkezik, illetve az egyes elemek színei összefüggésben állnak a bennük elhelyezkedő számértékkel, ezzel segítve a vizuális visszajelzést is az eredményről. A metrikák esetén az egyes mezők rövidített neve jelenik meg, ezek a következők:

- 0 valós, 0 prediktált kimenet: valódi negatív - True Negative (TN)
- 1 valós, 0 prediktált kimenet: fals negatív - False Negative (FN)
- 0 valós, 1 prediktált kimenet: fals pozitív - False Positive (FP)
- 1 valós, 1 prediktált kimenet: valódi pozitív - True Positive (TP)



16. ábra. Tévesztési mátrix példa. [20]

$$precizitás = \frac{TP}{TP+FP} \tag{9}$$

$$szenzitivitás = \frac{TP}{TP+FN} \tag{10}$$

$$pontosság = \frac{TP+TN}{TP+FN+TN+FP} \tag{11}$$

$$specificitás = \frac{TN}{TN+FP} \tag{12}$$

$$F1\ érték = \frac{2 \cdot precizitás \cdot szenzitivitás}{precizitás + szenzitivitás} \tag{13}$$

Ebben a típusú feladatban, melyet megvalósítottam, a legfontosabb a selejtes termékek (pozitív) megfelelő csoportosítása, hiszen lényegesebb a selejtes termékek megfelelőnek minősítése, mint a megfelelőek selejtesnek határozása minőségbiztosítási okokból. Az ehhez tartozó mérőszám a szenzitivitás. Azonban nem elég csupán ennek vizsgálata, hiszen, ha pontos selejtbesorolás mellett a jó termékeket is mind selejtesnek jelez, akkor az eredmény összességében ugyancsak nem helyes. Emiatt a következőképpen számolt metrikát alkalmazom a megfelelő végeredmény kiszámítására:

$$metrika = 0.8 \cdot szenzitivitás + 0.2 \cdot specificitás \tag{14}$$

A végső eredményként a 3 algoritmus közül annak nézzük a kimenetét, mely esetében ez a metrika a legnagyobb értékű. Itt is érvényes a már említett hármas kimeneti eredmény: vagy prediktálunk végeredményt (>75%), jelzőrendszerként beépíthető modellt kapunk (40-75%) vagy pedig a modell azt adja ki, hogy nem jósolható meg a kimenet (<40%).

Amennyiben az első eset áll fenn, akkor egyből jó eredménnyel tudjuk megjósolni a selejtes termékeket és ezzel a feladat el van végezve, nem szükséges további adatelőkészítési transzformációkat, redukciókat és modellezési lépéseket elvégezni. Ezzel felgyorsítható a projekt és számítási erőforrások tekintetében is kevesebbre van szükség.

Amennyiben a hiányzó és kiugró értékek esetén mindkét hozzájuk tartozó lépcső eredménye az, hogy nincs információtartalma annak, hogy hány elem tartozik bele és ezek hol helyezkednek el, akkor kijelenthető, hogy nincs különbség a selejtek és nem selejtek között, azaz ugyanolyan technikával történő kezelésük során nem vétünk hibát. Ezzel fel lehet gyorsítani az adatelőkészítés menetét, ugyanis könnyebben lehet választani módszereket, nem szükséges azok széles körben történő próbálgatása, a legjobb megtalálása.

Algoritmus 3: Algoritmusok alkalmazása

algoritmusok = [Catboost, SVM, KNN]

Function *Algoritmus :*

X = bemeneti változók;

y = kimeneti változó;

hibás = ahol a kimenet 1 értékű;

partíciók száma = 1;

if *nem selejtek száma » selejtek száma* **then**

szorzótényező = 10·selejtek száma;

partíciók száma = 10;

end

for $i = 1$ **to** *partíciók száma* **do**

X_{OK} = azon i. partícióban lévő szorzótényező db. X, melyek nem selejtes termékek;

y_{OK} = azon i. partícióban lévő szorzótényező db. y, melyek nem selejtes termékek;

X_{NOK} = azon X-ek, melyek selejtes termékek;

y_{NOK} = azon y-ok, melyek selejtes termékek;

$X0_{train}, X0_{test}, y0_{train}, y0_{test}$ = X_{OK} és y_{OK} felosztása tanító és teszt halmazra;

$X1_{train}, X1_{test}, y1_{train}, y1_{test}$ = X_{NOK} és y_{NOK} felosztása tanító és teszt halmazra;

X_{train} = $X0_{train}$ és $X1_{train}$ összekapcsolása;

X_{test} = $X0_{test}$ és $X1_{test}$ összekapcsolása;

y_{train} = $y0_{train}$ és $y1_{train}$ összekapcsolása;

y_{test} = $y0_{test}$ és $y1_{test}$ összekapcsolása;

if *kategorikus változók kódolása szükséges* **then**

Ordinal Encoding a kategorikus oszlopokra;

end

modell futtatása;

y_{pred} = a teszt adatok prediktálása;

tévesztési mátrix kirajzolása;

szenzitivitás számolása;

specificitás számolása;

kimeneti metrika számolása;

end

return *kimeneti metrikák átlaga;*

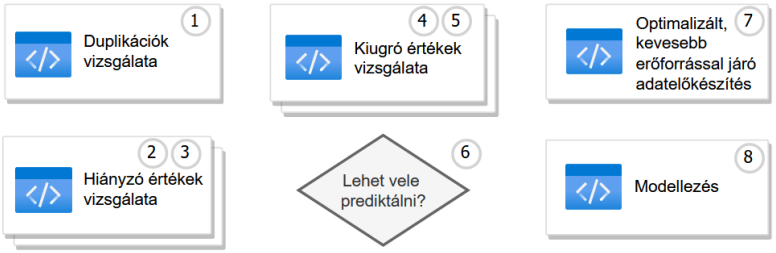
end

4.3. Többlépcsős rendszer felépítése

A 7. ábrán volt látható a teljes rendszer, melynek első alegységét mutattam be az előző fejezetben. A többlépcsős klasszifikációs rendszer, amennyiben az előző lépcsők segítségével a végső osztályozást el lehet végezni, csupán ezekből az elemekből állt, ahogy azt jelzi a kép is.

Amennyiben ezek eredménye nem megfelelő, úgy lehetőség van a klasszikus értelemben vett lépések következő sorozatainak elvégzésére, tehát az adatok előkészítése a 3.2. fejezet alapján, majd a megfelelő gépi tanulási algoritmus alkalmazása, melyek legismertebb típusai a 12. ábrán voltak láthatóak. A végső kimenete ezeknek a lépéseknek a klasszifikáció eredménye, melynek segítségével optimalizálni lehet a gyártósorokat a selejtek csökkentésének érdekében, illetve egyéb, bináris osztályozó feladatok elvégzését lehet támogatni adat oldalról.

Az egyes lépcsők összegezve láthatóak a 17. ábrán, illetve ezek fontossága egyenként a következő felsorolásban.



17. ábra. Lépcsők összegzése.

1. Duplikációk számának vizsgálata. Általánosságban a duplikációk azonnal törlésre kerülnek, itt azonban kiderül, van-e ennek információtartalma, jelenti-e a bizonyos számú duplikált azonosítójú sor az egyik kimeneti osztályt.

2. Hiányzó értékek számának vizsgálata. Hiányzó értékeket nagyon kis kivétellel minden algoritmus használata előtt valamilyen módon kezelni kell. Azonban, ha az egyik osztály esetén más eloszlású a hiányzó értékek száma (például sokkal több van jelen), akkor ezek feltöltése adott értékkel (átlag stb.) torzítani fogja az adathalmazt, hiszen olyan információt fog tartalmazni, mely nem kiegyensúlyozott a két különböző osztály között.

3. Hiányzó értékek helyének vizsgálata az oszlopok között. Annak elemzése, hogy van-e hatása, mely attribútumokhoz tartozóan szerepelnek ezen értékek. Hasonló a klasszikus gépi tanulási feladathoz, azonban az adathalmaz csak 1-es és 0-ás értékeket tartalmaz. Ha ennek

az eredményessége is alacsony az előzővel együtt, akkor bármilyen módon lehet kezelni a hiányzó értékeket (törléstől eltekintve, hiszen az nagyon redukálja a halmazt).

4. Kiugró értékek számának vizsgálata. Általánosságban a statisztikai jellemzőket nagyon meg tudja változtatni az outlierek jelenléte, emiatt szükséges ezek megváltoztatása. A vizsgálat jelentősége ugyanaz, mint a 2. pontnak.

5. Kiugró értékek helyének vizsgálata az oszlopok között. Fontossága ugyanaz, mint a 3. pontnak.

6. Döntéshozatal a további lépésekkel kapcsolatban. Jelentősége, hogy meghatározzunk szabályokat, mi történik az egyes eredmények hatására, mikor lehet már az előző lépcsőket használni a klasszifikációra és mikor szükséges a további lépések alkalmazása.

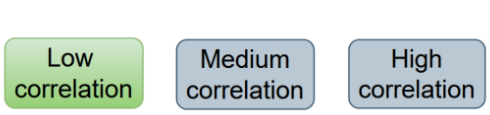
7. Adatelőkészítés, azonban már oly módon, hogy felhasználhatjuk az eddigi eredményeket, ezáltal egyszerűsödik a kiválasztandó módszerek kijelölése, alkalmazása, ezzel gyorsabbá téve ezt a lépést. Ahogy a 3.1. fejezetben már említésre került, a projektek idejének nagy része ezen lépésre fordítódik, így fontos csökkenést lehet elérni a rendszerem alkalmazásával. Fontossága, hogy még jobb adathalmaz jöjjön létre a végső fázishoz.

8. Modellezés. A kialakult végső adathalmazból kinyerhető információk kiaknázása valamilyen osztályozó algoritmus használatával. A végső eredmény kialakítása abban az esetben, ha az 1-5. lépés után ez még nem valósult volna meg.

4.4. Döntéstámogató rendszer és az eredmények közlési formái a felhasználó számára

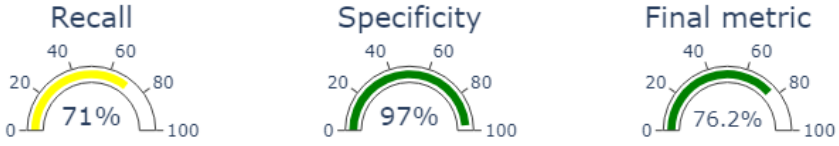
Az összes eddig említett elem kerül bele a döntéstámogató rendszerbe, melynek bemenete a 4.1. fejezetben megadott módon előkészített adat, kimenete pedig a klasszifikációs modell, mellyel új adatok esetén könnyebben eldönthető, hogy melyik csoportba tartoznak. Ez a rendszer a CRISP-DM metódusban az adatelőkészítés-modellezés fázisba tartozik, illetve tartalmaz kiértékeléshez vonatkozó kimeneti elemeket is.

A döntéstámogató rendszerbe beépítésre került még két elem. Az első egy automatizált beszámoló készítő, mely a program futásával egyidejűleg hozza létre a pdf formátumú riportot azon információkról, melyek az első alrendszerben, az *Elvesztett információk kiaknázása* részben kerültek megállapításra. A riport struktúrája úgy néz ki, hogy mindegyik elemhez (duplikációk, hiányzó értékek, kiugró értékek) bemutatja az összesített értékek eloszlását hisztogramon mindkét kimeneti csoportra vonatkozóan, ezután pedig az látható, hogy a Cramer-mutató szerint milyen nagyságú az összefüggés a két változó között, illetve, hogy alacsony, közepes vagy magas értékű. Ez úgy valósul meg, hogy három doboz közül az lesz zöld színű, amely a kimeneti érték.



18. ábra. Alacsony korrelációt mutató kimeneti elem.

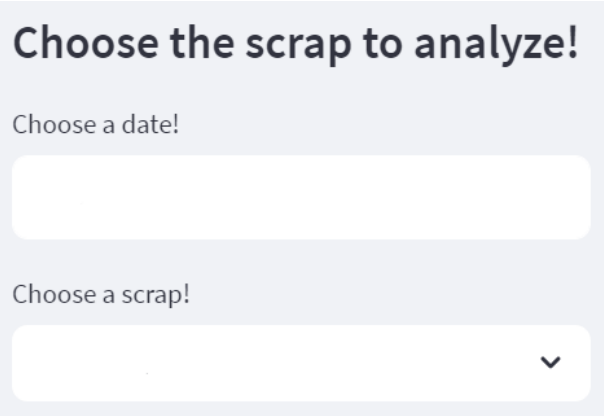
A hiányzó és kiugró értékek esetén pedig az is látható, hogy a 3 alkalmazott algoritmus közül melyik érte el a legjobb eredményt és az ahhoz tartozó kimeneti metrikák (szenzitivitás, specificitás, számolt érték) mik lettek.



19. ábra. Kimeneti metrikák ábrázolása.

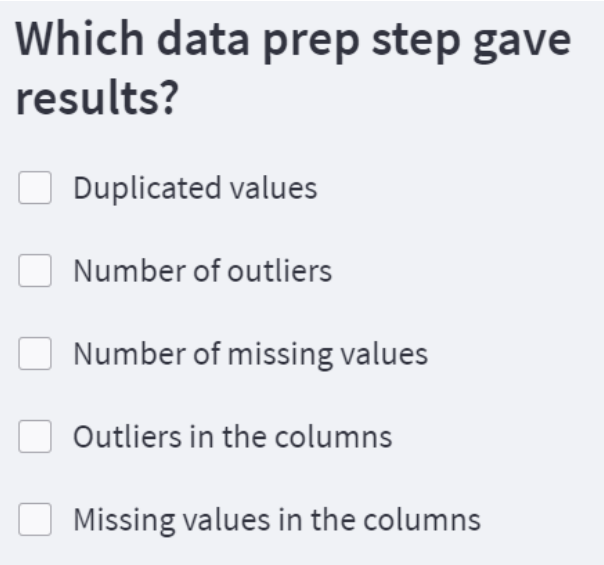
Ennek a beszámolónak a segítségével kapunk vizuálisan is képet arról, hogy mely lépcsők esetében sikerült elérni olyan eredményt, amellyel már lehetséges az osztályozás. Ennek a riportnak az eredménye bemenetként szolgál a következő kimeneti elem használatához.

Ez az elem egy olyan interaktív felület, mely vizuális eszköztárként szolgál arra, hogy jobban érthetővé váljék, mely attribútumok járulnak hozzá legjobban a selejtek csökkentéséhez. Elsőként ki lehet választani egy adott napot, illetve egy adott selejtet az azonosítója alapján.



20. ábra. Dátum és selejt azonosító kiválasztása.

Ezután ki lehet választani egy felsorolásból, hogy az eddigi feldolgozás során mely lépcsők adtak eredményt. Ezen elemeket a beszámoló alapján könnyen meg lehet határozni 1-1 adathalmaz esetén.

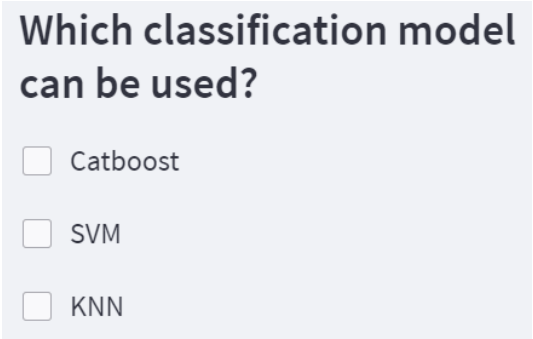


21. ábra. Választási lehetőségek.

Az egyes választási lehetőségek jelölése alapján változik dinamikusan a weboldal többi része. Az egyes lehetőségek által kialakult plusz információkat gyűjtöttem össze az itt látható felsorolásban. A teljes weboldal kinézete megtalálható a következő fejezetben, ahol valós gyártási adatokra mutatom be a rendszerem működését.

- duplikált sorok száma: ennek hatására az adott selejt esetén, amennyiben abból több sor került az adatbázisba, kiíratásra kerül a duplikált sorok száma és az összes sor, illetve pirossal színezve jelennek meg a sorok közötti különbségek, amennyiben vannak, így látható, hogy teljes duplikációról van szó vagy esetleg van köztük eltérés, kimenete tehát egy érték és egy aladathalmaz
- kiugró értékek száma: kiíratásra kerül, hogy az adott sorban hány kiugró érték szerepel, kimenete egy érték
- hiányzó értékek száma: kiíratásra kerül, hogy az adott sorban hány hiányzó érték szerepel, kimenete egy érték
- kiugró értékek elhelyezkedése az oszlopok között: amennyiben ennek van információtartalma, tehát a lefuttatott gépi tanulási modell esetén ez adott ki megfelelő eredményt, akkor kiíratásra kerül az adathalmaz adott selejthez tartozó sora és piros színnel jelölve jelenik meg benne az összes kiugró érték, kimenete egy aladathalmaz
- hiányzó értékek elhelyezkedése az oszlopok között: az előzőhöz hasonló esetben itt is a megfelelő aladathalmaz íródik ki, pirossal jelölve benne a hiányzó értékek helyét, kimenete ismét egy aladathalmaz

A továbbiakban következnek az oldalon megtalálható egyéb vizualizációk. Annak megfelelően, hogy mely elem kiválasztása történt a felületen, lefut egy modell. Szükséges még azonban megadni, hogy melyik modell adta a legjobb eredményt, ez is megtalálható volt a beszámolóban.

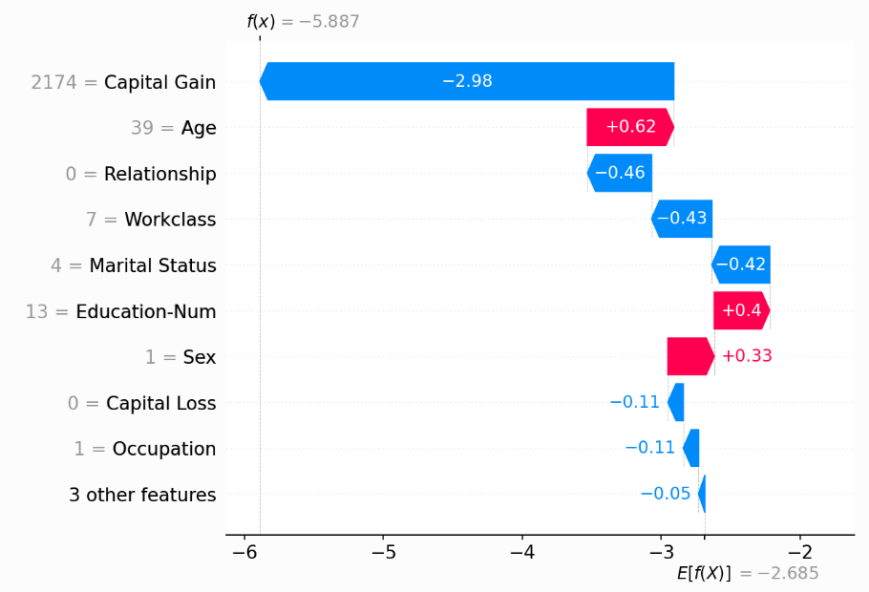


22. ábra. Algoritmus kiválasztása a beszámoló alapján.

Amennyiben az adatelőkészítési lépésekhez tartozó elemek közül egyik sem került kiválasztásra, úgy a rendszer szerint az optimalizált adatelőkészítés és modellezés lépcső következik. Ebben az esetben is lefut a modell és megkapjuk az eredményeket a vizualizációhoz, ilyenkor is három algoritmus közül a legjobb eredményt adó fut le.

Az első ábra a felületen a modell eredményeiből számolható Shapley értéken alapul. Ennek elméleti alapjai a játékelméletből származnak, ahol azt szerették volna meghatározni, hogy egy játék esetén az egyes játékosok mennyire járultak hozzá a végső eredményhez. Ezen metódus lett általánosítva más területekre, így jelent meg az adattudományok terén is. Az itteni értelmezésben azt lehet vizsgálni, hogy egy adott kimenethez (pl. selejt) az egyes attribútumok mennyire járultak hozzá. Ez azért is nagyon fontos, mivel ezzel egyfajta betekintést tudunk nyerni abba, hogy a változók milyen mértékben járulnak hozzá a kialakult eredményhez. Ez egyfajfa magyarázhatóságot ad, betekintést nyújt a döntéshozás okaiba. Első ránézésre ez nagyon hasonlít az adatelőkészítés során említett ábrázoláshoz, melyen az attribútumok fontosságát lehet megfigyelni. Azon feature importance grafikon a globális magyarázhatóság csoportba tartozik, ugyanis az összes elem alapján kinyert információkra vonatkozóan mutatja be az attribútumok fontosságát. [21] [22]

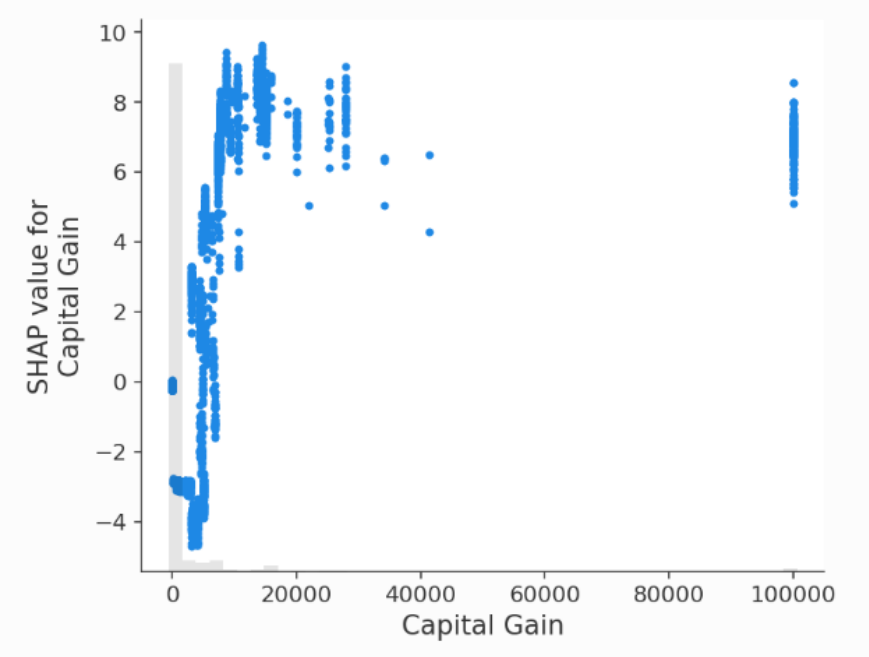
Az alábbiakban illusztrált ábra a lokális magyarázhatóságot segíti elő, ugyanis 1-1 kiválasztott elemre mutatja be az attribútumok fontosságát. Egy ilyen illusztráció látható a 23. ábrán.



23. ábra. Shapley értékek alapján egy elem ábrázolása. [23]

Értelmezése a következő: alul az ábrázolás egy olyan $E(f(X))$ értékről indul, mely a modell kimenetének várható értéke. Ezután minden attribútum esetén megmutatja, hogy az melyik irányban és mennyire járul hozzá a végső eredményhez. Felfelé haladva minden elem az előző nyíl végső értékétől indul, egészen addig, míg el nem éri a végső eredményt. Az első 10 legjobban hozzájáruló elem kerül kirajzolásra általános esetben, de látszik is az ábrán, hogy a többi elem már csak nagyon kismértékben játszik szerepet a döntéshozatalban.

Ezután a legjobban a végeredményhez hozzájáruló jellemző további vizsgálata következik. Amennyiben ez egy numerikus érték, kirajzolásra kerül egy scatter ábra, hogy jobban láthassuk az adott jellemző eloszlását a többi elem esetében, ezzel érthetőbbé válhat, hogy csak az adott elem esetében ért el ilyen eredményt vagy más elemek esetén is.



24. ábra. Scatter ábra egy változóról. [23]

Minden esetben emellett kirajzolódik egy hisztogram, mely ugyancsak az adott attribútum teljes adathalmazra történő hatásának megértését szolgálja.

5. A rendszer tesztelése valós gyártási adatokra

5.1. Bevezetés

A rendszeremet valódi gyártási adatokon is teszteltem. Ehhez a Robert Bosch Kft. hatvani gyárában lévő szelektív forrasztást végző állomás adatait használhattam fel. Ezen a soron a gépjárműelektronika csoporthoz tartozó motorvezérlők áramköri elemeit forrasztják be egy Ersa Versaflow géppel. Az adatbázisban lévő adathalmaz tartalmazza a forrasztás során alkalmazott paramétereket, a beszállítói adatokat, gyártás során keletkező egyéb adatokat (szállító eszköz azonosító, tárolási idő stb.) Az eredmények bemutatása során az adatokat már olyan formában mutatom meg, mely a rendszer bemeneti eleme, tehát mindegyik oszlopról csak annyi fog látszani, hogy kategorikus vagy numerikus a benne lévő adat. Ezen a gyártósoron a bináris klasszifikációs problémakör a selejt-nem selejt kategóriába esik, tehát selejteket próbálunk megjósolni. Forrasztás során többféle selejtfajta is felmerülhet, például, ha nem kerül megfelelő mennyiségű forraszanyag egy adott helyre, vagy akár, ha túl sok kerül és ezáltal olyan elemek is összekötésre kerülnek, melyeknek nem szabadna.

25. ábra. ERSA Versaflow3 szelektív forrasztógép. [24]

Az eredmények közlése során a titoktartási kötelezettségemből kifolyólag bizonyos paraméterrészeket ki fogok takarni, azonban ez nem befolyásolja a végső eredmény alakulását.

5.2. Az elkészült beszámoló az első alrendszer eredményei által

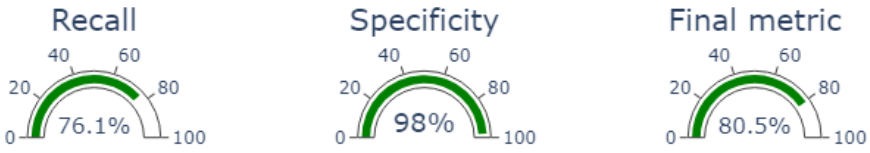
A következő két oldalon található az elkészült beszámoló. Látható, hogy az egyes esetekben hogyan alakultak az összegzések a két csoport esetén, illetve, hogy melyikre mennyi lett a Cramer-mutató. Mint azt a riport mutatja, mindhárom esetben alacsony volt ez az érték, így ennek eredményét nem tudjuk felhasználni, nem volt nagy eltérés a két osztály között.

Az oszlopok között történő vizsgálat során az első esetben, a hiányzó értékek esetén, habár a specificitás magas volt, azaz a hibátlan termékek besorolása megfelelő volt, de a szenzitivitás nagyon alacsony volt, a selejtes termékeket nem lehet prediktálni ezzel a módszerrel. Tehát az esetlegesen szükséges további adatelőkészítés során bármilyen metódussal lehet ezeket kezelni.

Az utolsó esetben a Catboost modell a kiugró értékek esetén azonban nagyon jól teljesített (84,9%), a nem selejtes termékeket szinte tökéletesen sikerült besorolnia, a selejtes termékek esetén pedig ugyancsak nagyon jó eredményt ért el, így ezt lehet használni klasszifikációs modellként.

A modell futási ideje erre az adathalmazra, mely 427054 sorból és 70 oszlopból állt, egy átlagos, nem erős erőforrásokkal rendelkező számítógépen nagyjából 20-30 perc között van, ebben benne van az adatok beolvasása, minden lépcső elvégzése, illetve a végső pdf beszámoló legenerálása, a hozzá tartozó képek mentése is.

Összehasonlításképpen az adathalmaz esetén elvégeztem a feladatot úgy is, hogy az általános lépéseken haladtam végig a következőket használva: duplikációk törlése, hiányzó adatok átlaggal történő feltöltése, kiugró értékek minimummal-maximummal történő helyettesítése, numerikus adatok skálázása, korrelációk alapján attribútum csökkentés. Ezekkel az adatelőkészítési lépésekkel az SVM modell bizonyult a legjobbnak, eredménye a következő:



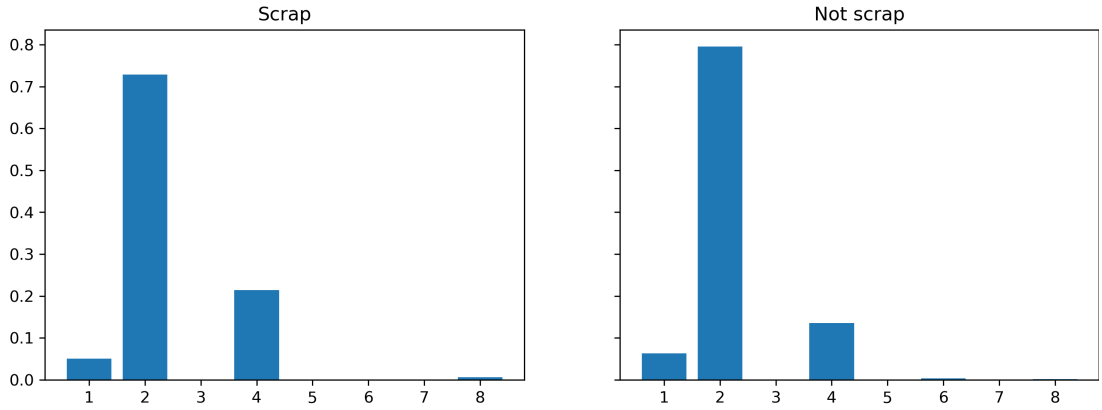
26. ábra. Hagyományos adatelőkészítéssel kapott eredmény.

Összegezve tehát erre az adatsorra, kis erőforrás befektetéssel, előre elkészített keretrendszer segítségével úgy sikerült jobb eredményt elérni, hogy nem kellett időt tölteni a megfelelő adatelőkészítési lépések kiválasztásával, illetve számos olyan lépést nem kellett elvégezni (skálázás, adatimputáció numerikus adatokra, adatredukció), melyet egy hagyományos adatelőkészítési lépés esetén szükséges lett volna.

Data preparation report

31/10/2023

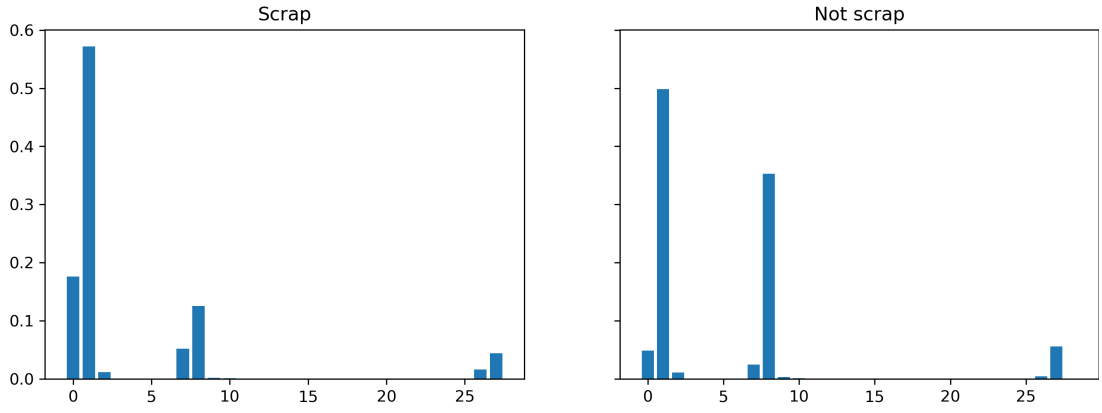
1. Duplications



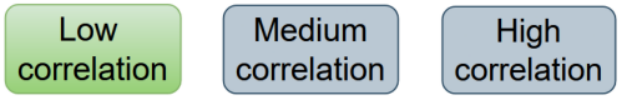
The dependency between the end result and the number of duplications: 4.5253%

Low correlation | Medium correlation | High correlation

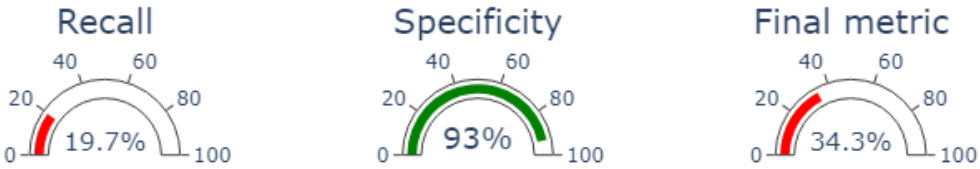
2. Missing values



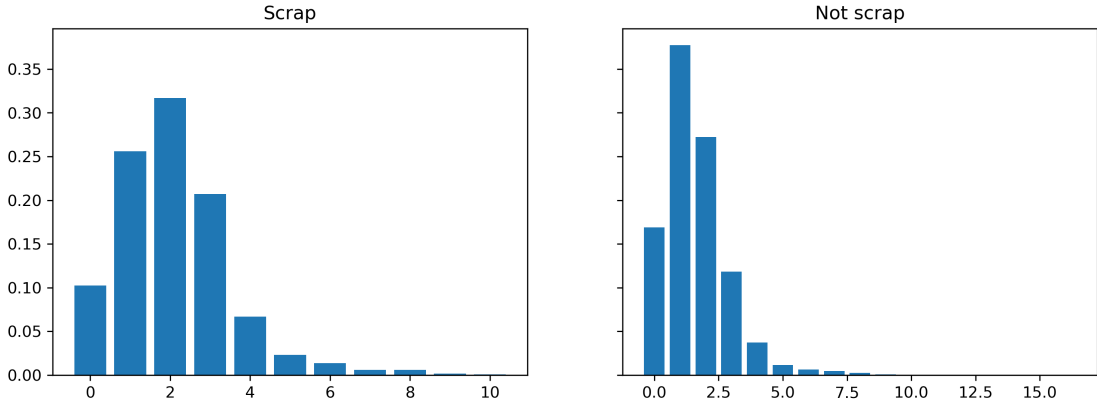
The dependency between the end result and the number of duplications: 9.52%



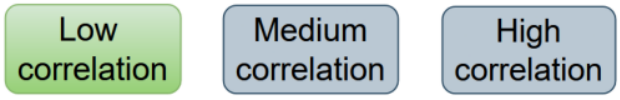
The selected algorithm: KNN



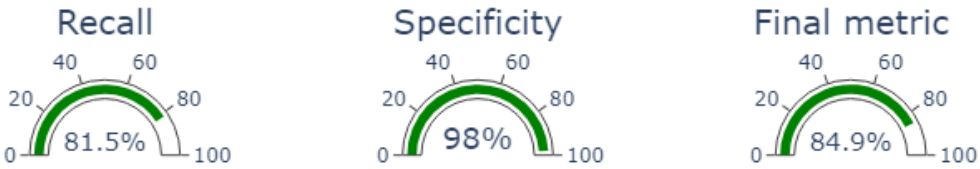
3. Outliers



The dependency between the end result and the number of duplications: 5.6446%



The selected algorithm: Catboost



5.3. Interaktív felület a legyártott selejtes termékek vizsgálatához

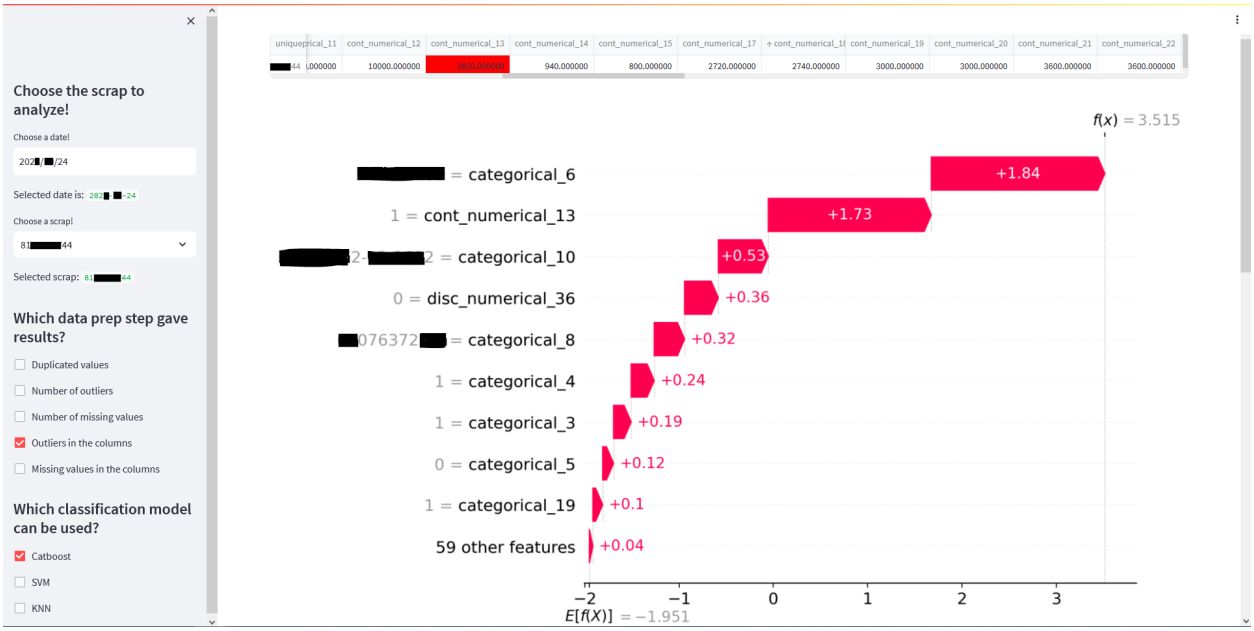
Az oldalon a beszámoló alapján két bejelölnivaló mező van, ez pedig az outlierek elhelyezkedése az oszlopok között, illetve a Catboost algoritmus. A kérdéses nap és selejt kiválasztása után le fog futni a modell és kirajzolásra kerülnek az adott elemhez tartozó vizualizációk, illetve az adathalmaz, jelölve a kiugró értékeket. Erre mutatnak példát a következő ábrák. Ezeken látható, hogy egy kiugró érték (ez fért rá egy képernyőre) pirossal van jelölve, maga a Shapley érték alapján rajzolt attribútum lista, illetve az első változó eloszlása hisztogramon. Ennek alapján tehát láthatjuk, hogy ezen elem esetén egy kategorikus és egy kiugró érték (jelölt) járul hozzá legjobban ahhoz, hogy selejt legyen.

A harmadik ábrán azt is bemutatom, hogy ha a többi esetben lett volna eredménye a rendszernek, akkor pontosan hogyan is nézne ki a felület. Itt látható lesz a duplikációk száma, a hiányzó és kiugró értékek száma, illetve az aladathalmazok a megjelölt piros cellákkal.

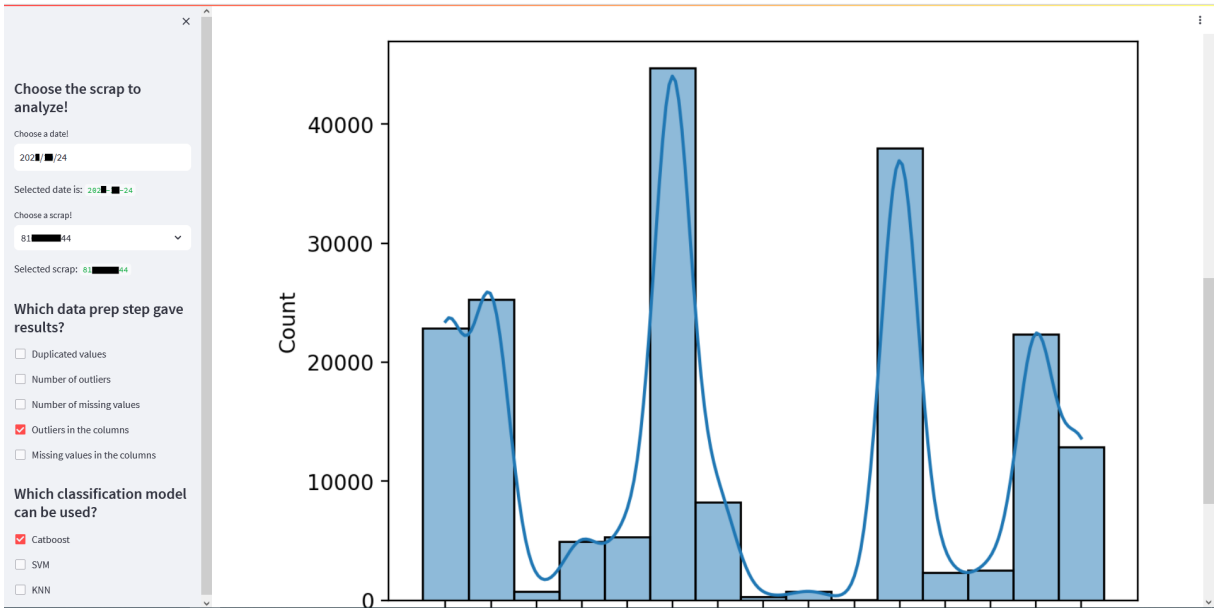
Alapjáraton előfordulhat, hogy a Shapley érték alakulását bemutató ábrán bizonyos attribútumok éppen az ellenkező irányba is mutathatnak, de ezen elem esetén ez nem fordult elő, azonban egy másik terméknél igen, erre látható példa a 30. ábrán.

Összefoglalva tehát ezen vizuális eszközt, a szakértő nem csak az adathalmaz rejtette információkat tudja könnyen, programozás nélkül megtekinteni, hanem hamar képet kaphat arról, hogy pontosan mi járulhatott hozzá 1-1 selejt képződéséhez. Ilyen módon visszacsatolást kapva, hogy milyen problémák jelentkeztek.

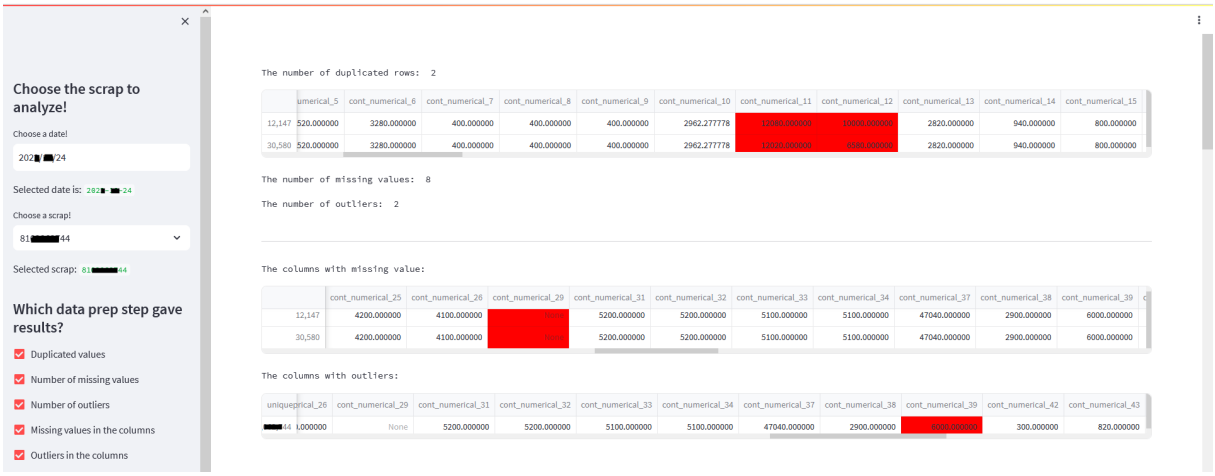
A program úgy épül fel, hogy minden gombnyomásra, változtatásra újra fut elölről, azonban van lehetőség bizonyos adatok cache-be történő mentésére, így a futási idő sem okoz gondot, nagyon rövid várakozás után frissül az oldal.



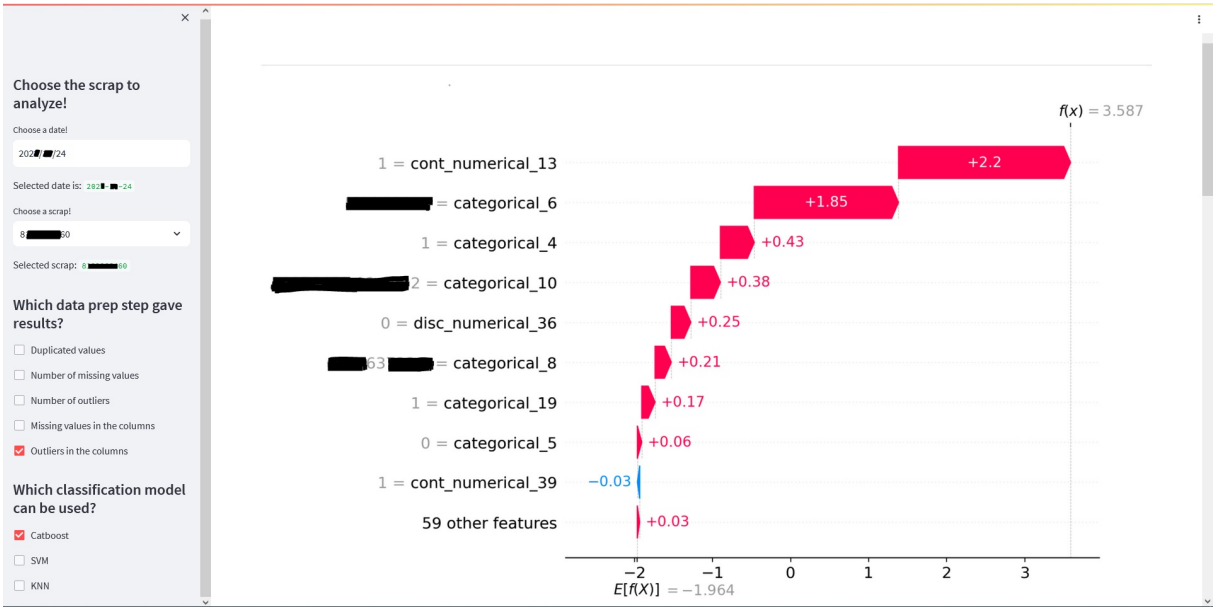
27. ábra. Shapley értékre alapuló diagnosztika egy adott elemre.



28. ábra. Hisztogram az első helyezett változó esetében.



29. ábra. Minden kimutatható aladathalmaz.



30. ábra. Másik irányba mutató attribútum érték.

6. Összefoglalás

Jelen munkámban bevezettem egy többlépcsős selejtfelismerő rendszert, amely a következő új elemeket tartalmazza: kiaknázza az adattisztítás fázisában rejtett többletinformációkat, több vizuális eszközt és eljárást építettem be, amelyek a felhasználó számára könnyen értelmezhetők, kiépítettem egy interaktív környezetet, amely által a szakértő kapcsolatba léphet a modellel, továbbá egy diagnosztikai eljárást vezettem be a Shapley értéket beépítve, amely segítségével a felhasználó felismerheti, mik lehetnek a hibát okozó tényezők.

Összefoglalóan a munkám során először bemutattam, hogy az elkészített rendszerem hogyan kapcsolódik a modern gyártási folyamatokba, melyeknek fontos részét képezik az adatok feldolgozása. Mivel a gyártásban az adattudományi feladatokat a CRISP-DM lépései által szokták definiálni, így elhelyeztem a megoldásom ebben a rendszerben. A célom, melyet sikerült elérni, tehát az volt, hogy csökkenteni lehessen az erőforrásokat az adatelőkészítés során, illetve, hogy ezt úgy végezzük el, hogy minden rendelkezésre álló adatot megfelelő módon és minőségben használjunk fel, ne veszítsünk fontos információt a selejtek csökkentésének érdekében az adattisztítás során.

Ezután részletesen kifejtettem, hogyan is működik a keretrendszerem, mik a részei, milyen szempontból fontosak az egyes lépcsők. Ezután ismertettem azon kimeneti vizualizációs elemeket, melyek a végső modellen kívül eredményt szolgáltatnak. A valós gyártási adatokra tesztelve szemléltettem a rendszer erősségét, közöltem az információkat, melyek segítségével úgy tudtam megjósolni a selejtek túlnyomó többségét, hogy csupán a keretrendszeremet használtam. Illusztráltam emellett magát az automatizáltan elkészült beszámolót, illetve interaktív felületet is.

Többféle továbbfejlesztési irány is lehetséges még a rendszeremmel kapcsolatban. Az első, hogy általánosabbá lehetne tenni úgy, hogy ne csak bináris klasszifikációt tudjon végrehajtani, hanem többosztályosat is, ezzel elérve, hogy többféle hibafajta is jósolhatóvá váljon. A kiugró értékek vizsgálata során az IQR metódust használtam, azonban rengeteg egyéb algoritmus van még, például az egyik legújabb, az ECOD, mely nagyon jól szerepel sok folytonos adat esetén, ennek beépítése még javíthat az algoritmuson. Az interaktív felületet lehetne még informatívvá tenni azáltal, hogyha két egymás után keletkező selejt esetén ugyanaz az első számú attribútum az ábrán, akkor azt jelezze a rendszer (lehetséges, hogy például egy szenzorral kapcsolatos általános hiba lépett fel). Utolsóként pedig megemlíteném, hogy ki lehetne egészíteni egy olyan kóddal, mely az adatot automatikusan hozza a bemeneti formára, tehát eldönti, mely oszlopok tartalmaznak kategorikus és melyek numerikus értékeket, így a felhasználónak ezzel sem lenne feladata.

Hivatkozások

[1] Saurabh Vaidyaa, Prashant Ambad, Santosh Bhosle: Industry 4.0 – A Glimpse (2018)

[2] Bzhwen A Kadir: Designing new ways of working in industry 4.0 (2020)

[3] Pai Zheng, Honghui Wang, Zhiqian Sang, Ray Y. Zhong, Yongkui Liu, Chao Liu, Khamdi Mubarok, Shiqiang Yu, Xun Xu: Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives (2018)

[4] Bernar Taşcı, Ammar Omar, Serkan Ayvaz: Remaining useful lifetime prediction for predictive maintenance in manufacturing (2023)

[5] Ivo Perez Colo, Carolina Saavedra Sueldo, Mariano De Paula, Gerardo G. Acosta: Intelligent approach for the industrialization of deep learning solutions applied to fault detection (2023)

[6] Julius Pfrommer, Clemens Zimmerling, Jinzhao Liu, Luise Kärger, Frank Henning, Jürgen Beyerer: Optimisation of manufacturing process parameters using deep neural networks as surrogate models (2018)

[7] https://www.precedenceresearch.com/generative-ai-in-manufacturing-market

[8] Dorothea Schneider, Steffen Ihlenfeldt, Steffen Huber, Hajo Wiemer: Dmme: Data mining methodology for engineering applications – a holistic extension to the crispdm model. (2019)

[9] https://www.ibm.com/docs/en/spss-modeler/saas?topic=preparation-data-overview

[10] Anaconda: State of data science 2020 Moving from hype toward maturity (2020)

[11] Anaconda: State of data science 2022 Paying the way for innovation (2022)

[12] Jong Hae Kim, Sang Kyu Kwak: Statistical data preparation: management of missing values and outliers (2017)

[13] Jason Brownlee: Data preparation for machine learning

[14] https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/

[15] https://en.wikipedia.org/wiki/Cram%C3%A9r%27s_V

[16] https://catboost.ai/

[17] Kiber-fizikai rendszerek tantárgy: Gépi tanulás órai diasor

[18] Bodon Ferenc, Buza Krisztián: Adatbányászat

[19] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Jonathan Taylor: An introduction to statistical learning

[20] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html

[21] Lloyd S. Shapley: A value for n-person games (1953)

[22] https://dmlab.hu/blog/xai-avagy-a-magyarazhato-mesterseges-intelligencia/

[23] https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/waterfall.html

[24] https://www.kurtzersa.de/produkte/elektronikfertigung/loetmaschinen/selektivloetanlagen/versaflow-345