Eötvös Lóránd Tudományegyetem
Budapesti Corvinus Egyetem



# Machine learning algoritmusok aktuáriusi alkalmazása

Készítette: Szentkereszti Gábor
Biztosítási és pénzügyi matematika MSc
Aktuárius szakirány
2020

Szakszemináriumvezető: Dr. Vékás Péter

# NYILATKOZAT

**Név:** Szentkereszti Gábor

**ELTE Természettudományi Kar, szak:** Biztosítási és pénzügyi matematika

**NEPTUN azonosító:** R94FL0

**Szakdolgozat címe:**
Machine learning algoritmusok aktuáriusi alkalmazása

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2020.05.10.

*a hallgató aláírása*

# Tartalomjegyzék

# Táblázatok jegyzéke



# Ábrák jegyzéke

# 1. Mesterséges intelligencia és gépi tanulás

A mesterséges intelligencia (Artificial intelligence, későbbiekben: AI) manapság nagyon felkapott és hangzatos tudományterület, de hogy tulajdonképpen mit is jelent, mire képes, azt már nagyobb homály fedi és az emberek nehezen tudják besorolni valamely szakterülethez. Az AI viszonylag új tudományágnak tekinthető, születését gyakran Alan Turing 1950-es a *Computing Machinery and Intelligence* tanulmányára teszik. Itt fogalmazza meg Turing a később Turing tesztként elhíresült metódust, ami az intelligencia egyfajta definíciója. Turing úgy fogalmaz, hogy egy gép intelligens, ha a vele kapcsolatba lépő személy nem tudja eldönteni a reakciók alapján, hogy egy géppel vagy egy emberrel van interakcióban.

A mesterséges intelligencia nem határozható meg egy tudományterületen belül, sok ág metszeteként alakult ki az összetettsége miatt. Russel és Norvig a következőket emeli ki: számítástudomány, közgazdaságtan, matematika, pszichológia, filozófia, kibernetika és nyelvészet. (Stuart J. Russel, Peter Norvig, 2009, 5 - 16. o.)

Ahhoz hogy a gépek teljesítsék a korábban említett Turing tesztet, vagyis hogy emberként viselkedjenek a következő kritériumoknak kell megfelelniük (Stuart J. Russel, Peter Norvig, 2009, 2. o.):

- Natural language processing (NLP)[1]: Sikeres kommunikáció egy adott nyelven.
- Tudás reprezentáció (Knowledge reprezentation): Elraktározni azt, amit tud vagy hall.
- Automatizált gondolkozás: Felhasználni az elraktározott információt, kérdéseket megválaszolni és új összefüggéseket levonni.
- Gépi tanulás (Machine learning, rövidítés: ML): Új körülményekhez való igazodás és mintázatok kikövetkeztetése.

Ezek csak a gondolkozást leíró részek, bővíthetjük a Turing tesztet, így egy még inkább emberhez hasonlító gépet kaphatunk. A teljes Turing teszt:

[1]Gyakran bizonyos módszerek angol megfelelőjét fogom alkalmazni. Erre véleményem szerint azért van szükség, mivel nagyon sok módszer elég újkeletű és nem alakult ki megfelelő magyar terminológia rá, illetve ha ki is alakult, lehet hogy nehezebben azonosítja be az olvasó az által, mintha az elterjedt angol megnevezéseket használnám.

- Gépi látás (Computer vision): Objektumok észlelése
- Robotika: Tárgyak mozgatása

Célom itt az AI egy rövid összefoglalása volt, hogy teljesebb képet kapjunk arról, hogyan is tudunk eljutni a mi témakörünkhöz, tehát a Machine Learning-hez. Így már látható, hogy a ML milyen helyet foglal el az AI-on belül.

Ez a technikai fejlődés a biztosítók számára is sok előrelépési lehetőséget kínál, amit ha jól adaptálnak, nagy előnyre tudnak szert tenni a piacon.

# 2. AI a biztosításban

A biztosításban az adat mindig is fontos szerepet játszott. A technológia és a számítástudomány fejlődésével, ezért nem meglépő, hogy az új technológiák nagyban implementálva vannak a biztosítási szektorban. A mesterséges intelligencia egy új utat nyitott meg a biztosítók részére, hogy még hatékonyabban tudjanak működni. A leggyakoribb problémák, amelyre egy újfajta megoldást kínál az AI a biztosítási szaktorban a csalásdetektálás, kárrendezés, online szerződéskötés, szofisztikáltabb kockázat besorolás és díj kalkuláció, valamint a személyre szabott marketing. Ezekből is látható, hogy majdnem teljes mértékben lefedik a biztosító működését. A következőkben felvázolnék néhány innovációt, amik a tradicionális biztosítási működésben nem jelentek meg, de az AI fejlődésével teret nyertek, avagy olyan módszereket, amik korábban is léteztek, de az AI ezt talán még tökéletesebben tudná kivitelezni. Igyekeztem olyan fejlesztéseket keresni, amik gépjármű biztosítás szempontjából relevánsak lehetnek, mivel későbbiekben egy ilyen adatbázist fogok elemezni. Így összességében talán egy még teljesebb képet kaphatunk e biztosítási ágazat jövőjéről.

A UBI (Used Based Insurance) egy újfajta megközelítésmódot kínál a járműbiztosítások kivitelezéséhez. Többfajta verziója ismert, de hasonló logika mentén működnek. A működésének alapja, hogy valósidejű adatokkal dolgozik, ezek az adatok különböző műszerek segítségével mérhetőek, amiket továbbít egy eszköz a biztosítónak. Ezek az adatok lehetnek a megtett út (PAYD - Pay As You Drive), a sebesség (Pay As You Speed) vagy az egyén közlekedési időpontja. Ezekkel az adatokkal még akkurátusabb kockázat besorolásra lehet képes a biztosító. Például kimutatható

(Yuanjing Yao, 2018) (de logikusnak is tűnik) hogy sokkal több közlekedési baleset történik hétköznaponként a reggeli órákban és délután 16-17 óra körül, mint más napszakokban. Ez nagyobb kockázatot is jelent természetesen, így akik a munkába tömegközlekedéssel járnak és autójukat csak a hétvégén használják, azok kevésbé lesznek kockázatosak, így a díjuk is alacsonyabban lesz meghatározva. [2]

Ehhez járult hozzá erősen a telematics, ami egybe foglalja az összes olyan eszközt és technológiát, ami integrálja a telekommunikációt, az információt és a kommunikációs technológiákat. Biztosítási szempontból a telematics méri a szükséges adatokat, amelyek az UBI módszerhez szükségesek. Négyfajta eszközt alkalmaznak jelenleg e jelek mérésére (Yuanjing Yao, 2018): dongle, black box, beágyazott telematics eszközök és az okos telefon. Telematics hátrányai között említik, hogy ezen eszközöknek vannak adminisztrációs költségeik, a jövőben a technológia még szélesebb körűbb elterjedésével egy cél lehet, hogy miképpen tudják minél olcsóbban megoldani a működtetést. Jelenleg a legelterjedtebb eszközök a dongle és a black box, azonban költségmegtakarítás szempontjából az okos telefonokban látják a jövőt, hogy begyűjtse és továbbítsa a szükséges adatokat. Persze itt már megbízhatósági problémák merülnek fel az adatokkal kapcsolatban. Látható a sok felmerülő kérdés az UBI kapcsán, de a tendenciák azt mutatják, hogy bővülő piacról van szó és a biztosítók előszeretettel használják ezt a technológiát. Ezt igazolja, hogy 2013-ban 5,5 millió, azonban 2018-ra már 107 millió UBI használó volt jegyezve.[3] Észak-Amerikában tapasztalható leginkább az elterjedtség.

A Tractable egy stratup cég, amit Alex Daylac és Razvan Ranca alapított 2014-ben.[4] Machine learning szempontból munkásságuk kép klasszifikációnak tekinthető, amihez mesterséges neurális hálókat használnak. Szolgáltatásuk gyorsítja a kárbejelentést és a kárfelmérést. A technológia a gépjármű biztosítás és egyéb vagyonbiz-

[2]Ez a jelenség a magyar adatokon is egyértelműen látszik. A KSH 2015-ös kiadványa alapján a legtöbb közúti baleset reggel 7-8 és 17-18 óra között történik. Hét napjait tekintve hétfő és péntek a legveszélyesebb, feltehetően ekkor még/ már a hétvége kényelmét élvezik az emberek és figyelmetlenebbek. Hétvégén 11-12 óra között történik a legtöbb baleset, illetve 16-17 óra között van egy másik tetőzés. Forrás: `https://www.ksh.hu/docs/hun/xftp/idoszaki/baleset/baleset15.pdf` (Utolsó letöltés: 2020.04.27.)

[3]`https://www.statista.com/statistics/737540/ubi-users-worldwide/` (Utolsó letöltés: 2019.08.01.)

[4]Honlapjuk itt megtekinthető: `https://tractable.ai/` (Utolsó letöltés: 2019.09.06.)

tosítások tekintetében használható (pl.: lakásbiztosítás). Működési metódusa, hogy a káresemény bekövetkeztekor az egyén fényképeket készít a károsult vagyontárgyról, majd ezeket elküldi a biztosítónak. A Tractable segítségével a biztosító a képek alapján meg tudja határozni, hogy nagyságrendileg mekkora lesz a kár az adott vagyontárgyon. Így gyorsul a kárbejelentési procedúra és a kárfelmérés, ami által a biztosító jelentős költségeket tud megtakarítani. A Tractable oldala szerint a képklasszifikációs deep learning algoritmus hiba rátája 2015-ben az átlagos emberi hiba ráta alá csökkent.

ML technikákat gyakran használnak csalás detektálására, mind a banki mind a biztosítási szektorban. Sokfajta megközelítés létezhet, az egyik legegyszerűbb egy logisztikus regresszió vagy egyéb hagyományos klasszifikációs módszer, ha rendelkezésre áll egy megfelelő adathalmazunk. Biztosítóknál a sok bejelentés közül nehéz lehet a nem szabályszerű esetek kiszűrése, sok munkát és figyelmet igényel a dolgozók részéről, ezeket az erőforrásokat spórolják meg a csalás detektáló algoritmusok. A csalás kiszűrése sok milliókat takaríthat meg évente a biztosítók számára, ezért erőteljesen foglalkoznak vele. Előbbi okok miatt a piacon is elkezdtek ilyeneket fejleszteni, az egyik legnépszerűbb a Daisy Intelligence, amit 2003-ban alapított Gary Saarenvirta.[5] Elmondásuk alapján reinforcemnet learning [6] mechanizmust használnak arra, hogy a csalást kiszűrjék, ez által növeljék a cégek teljesítményét.

A Zendrive az UBI egyik megvalósulása, kockázat management szempontjából a hagyományos díjkalkulációnál figyelembevett adatokon kívül, egyéb vezetés közbeni mért kockázatokra is támaszkodik.[7] A Zendrive egy applikáció, ami az okos telefon segítségével különböző mutatókat rögzít (GPS, gyorsulás stb.). Ezekből az adatokból a következőket tudja meghatározni: agresszív gyorsulás, heves fékezés, adott területen sebességtúllépés, vezetés közbeni mobil telefon használat stb. Ezeket az adatokat az okos telefon szenzorival begyűjtve és különböző algoritmusokon

[5]Honlapjuk itt megtekinthető: `https://www.daisyintelligence.com/` (Utolsó letöltés: 2019.09.06.)

[6]Az ML algoritmusok alapvetően három csoportba sorohlatóak: Supervised, Unsupervised és Reinforcement learning. A legutóbbival mi nem fogunk foglalkozni, azonban egy nagyon érdekes és előszeretettel kutatott ágról van szó. A legtöbb hozzá tartozó algoritmus alapja egy büntetés/jutalom faktor és egy cél függvény, ami a tesztelés során megtanítja a gépnek, hogy milyen műveleteket éri meg végrehajtani és milyeneket nem a cél elérése érdekében.

[7]Honlapjuk itt megtekinthető: `https://zendrive.com/` (Utolsó letöltés: 2019.09.06.)

futtatva egy nagyon precíz kockázati profilt képesek előállítani az egyénről, mely segítségével már kalkulálható a díja. Honlapjuk szerint 160 milliárd mérföldnyi adatok elemeztek már, ezek segítségével alakulnak ki az algoritmusaik. Adataik alapján elemzéseket is publikálnak, amelyek szerint három kategóriába sorolták a vezetőket: excellent, fair és risky. Az excellent kategóriába átlagosan a vezetők 15%-a, a fair-be 60% és a risky-be 25% kerül. Kimutatták a cikkük szerint, hogy a legrosszabb vezetők (tehát a 25%) okozzák a balesetek 53%-át.[8] Ezen kívül felhívják a figyelmet, hogy napjainkban a telefonfüggő vezetők 1,5x nagyobb veszélyt jelentenek a közlekedés során, mint az ittas sofőrök.

Ezen technológiák előnye a biztosítási kockázat besoroláson túl, hogy az egyén tudatában van a viselkedése költségvonzatának. Így rábírhatja a vezetőt az óvatosabb közlekedésre és a szabályok betartására. Ezen kívül az eszköz hamar észlelheti, hogy kár történt, így a biztosító rögtön értesül róla. Akár elkezdheti rögtön az assistance szolgáltatást vagy egészségügyi segítség szervezését esetektől függően. Természetesen itt is akadnak negatívumok. Például egy eszköz nehezen észleli, ha az egyén a saját hibáján kívül vagy mások hibájából fékez hirtelen vagy a vezető dönthet úgy, hogy a sárga lámpán még áthajt a hirtelen fékezés helyett, amit az eszköz negatívan észlelne.

A bemutatott AI alkalmazások után szűkítek egyet a kereten és most specifikusan ML megoldásokat ismertet, amelyek kapcsolódnak aktuáriusi tevékenységekhez.

Henckaerts et al. (2019) megemlít három követelményt az ML algoritmusok árazásra használt hatásával kapcsolatban, amire érdemes figyelmet fordítani. Először említik a GDPR-t, amiből kiemelik, hogy a személyeknek joguk van megkövetelni valamilyen logikus magyarázatot, ami a különböző döntésekhez vezetett, ahogy fogalmaznak, ezért transzparenciára és könnyen kommunikálhatóságra van szükség azon folyamatok kapcsán, ami alapján meghatározzuk az árat. Ez kiterjed a szerződőkre, menedzserekre és a szabályozókra egyaránt. Másodszor említik, hogy a szerződőknek fair árazást kell alkalmazni, ami tükrözi a kockázati profiljukat. Harmadszor kiemelik, hogy a biztosítónak egy szociális szerepe van, ami szolidaritást generál a szerződők között. Ezeket összefoglalva arra a következtetésre jutnak, hogy

[8] `https://zendrive.com/wp-content/uploads/2019/04/Zendrive_White_Paper_Loss_Control.pdf` (Utolsó letöltés: 2019.09.07.)

az algoritmikus árazásnak meg kell felelnie a transzparenciának, a korrektségnek és a szolidaritásnak. A legutolsó követelmény véleményem szerint vitatható. El kell különítenünk az állami társadalombiztosítást és a piaci biztosítókat, ezt az elválasztást a cikk nem teszi meg. Modellezés tekintetében az említett szerzők fa alapú modelleket alkalmaznak. Kiemelik, hogy egy döntési fa végén csoportok alakulnak, amik megfelelően homogenizálják az állományunkat. Rájuk szabható egy konstans díj csoporton belül. A döntési fa, véletlen erdő, Gradient Boosting Machine [9] és a GLM közül végül a Gradient Boosting Machine modelljük adta a legjobb eredményt.

Panlilio et al. (2018) célja az volt, hogy bemutassák a tradicionális aktuáriusi feladatokban alkalmazható ML technikákat. Először egy kamatláb modellezést írnak le, mivel ez az aktuáriusi gyakorlatban fontos szerepet tölt be. Kiemelik, hogy a megfelelő kamatláb előrejelzés hatással van az ALM-re, a megfelelő élet és nyugdíjbiztosítási termékekre vagy akár különböző tőke modellezésekre bizonyos szcenáriók mellett. Két modelltípust (Véletlen erdő és Bayesian Structural Time Series) tesztelést futtattak le, egyrészt olyan adatokkal, ami tartalmaz szöveg inputot, másrészt szöveg input nélküli adatokkal. A szöveg inputos modellek jobban teljesítettek. A szöveg, mint magyarázó változó alkalmazása egy új utat nyithat meg bizonyos modellek kapcsán. Ehhez a szöveg adathalmazt elő kell készíteni, ez alatt a töltelékszavak eltávolítását értik (pl.: a, az, ... stb.), ez után a szavakat a szótőre bontják vissza. Innentől a gyakoriságuk alapján elemezhető. Rövid példával szemléltetve, ha a „növekedés" és a „profit" gyakran fordul elő, akkor pozitív gazdasági szcenárióra számíthatunk. Panlilio és szerzőtársai egy mortalitási modellt is építettek ML eszközökkel. Az eredményváltozó a halálozási kor volt, magyarázóváltozónak különböző személyes és egészségügyi adatokat használtak. Három modelltípust alkalmaztak: Ridge regresszió, Döntési fa és a Gradient Boosting Machine. Ezek közül náluk a Ridge regresszió bizonyult a legjobbnak.

Az alkalmazások sokszínűségét mutatja, hogy tartalékszámításnál is előszeretettel vizsgálták a ML módszereket. Egyik jó példa erre az AUSTIN 2018-as kiadványa. A munka csoport vezetője Salma Jamal volt. Összehasonlítottak tradicionális tartalékszámítási módszereket, mint például a Lánc-létra vagy a Bornhuetter-Ferguson módszer, ML alapú eszközökkel. A bekövetkezett és a kifizetett károkat modellez-

[9] Később bemutatásra kerül.

ték. A kifizetetteknél a neurális hálók, a bekövetkezetteknél a Gradient Boosting Machine teljesített a legjobban.

Richman (2018) a következő Schreiber idézettel indít: *„The next insurance leaders will use bots, not brokers, and AI, not actuaries"* [10]. Ez majd idővel eldől, hogy mennyire valósul meg, azonban ahogy folytatja a cikket és amilyen alkalmazásokat hoz a biztosítási szektorban, amire gépi tanulási eszközöket használhatnak/használnak, az már minden bizonnyal megvalósult valahol. A korábban látott példákkal kezdi, jelesül az árazással (kárgyakoriság és kárnagyság becslése), ezen kívül megemlíti a tartalékolást, a mortalitási modelleket (mortalitási ráta és halálozás éve) és különböző életbiztosítási termékek értékelését. Összefoglalva az ML hatáskörét a következőt emeli ki: *„If an actuarial problem can be expressed as a regression, then machine and deep learning techniques can be applied"*[11]. A szerző az említett aktuáriusi problémakörökön a neurális hálókat alkalmazta előszeretettel és a hagyományos GLM alapú megközelítéssel hasonlította össze. Elemez úgymond „high-frequency" adatokat is, ilyenek például, amik a korábban bemutatott UBI-tól származnak gépjármű esetében, de lehet ez egy okos eszköz, ami az egészségügyi funkciókat méri. Ezek az eszközök szinte másodpercenként szolgáltatnak adatot a megfigyelt alanyról, így hagyományos eszközökkel már nem elemezhető problémát kapunk. Itt már a deep learning mezején találjuk magunkat.

# 3. A modellezés során használt algoritmusok

## 3.1. Döntési fa

A döntési fa egy előrejelzésre használatos eszköz, megalkotása az 1980-as években történt, két legnagyobb alakja Ross Quinlan és Leo Breiman. Mechanizmusában az eredményváltozó legjobban elkülönítése a lényeg. Erre többfajta algoritmus létezik. Döntési fa alkalmazható mind klasszifikációra, mind regresszióra. A döntési fák kedveltek, mivel működése az emberi gondolkozásra hasonlít, könnyen interpretálható

[10]Magyarul: "A következő időszakban a vezető biztosítók robotokat fognak használni nem brókereket, mesterséges intelligenciát és nem aktuáriusokat."

[11]Magyarul: "Ha egy aktuáriusi probléma felírható regresszióként, akkor gépi tanulási eszközök is alkalmazhatók."

és vizualizálható. Működése átlátható, ezért „*white boxnak*" is nevezik, ellenben a később tárgyalt neurális hálókkal. Hátránya hogy nem eléggé robosztus, tehát az adatok kis megváltozása nagyban módosíthatja az eredményt, illetve hajlamos a túlilleszkedésre.

Először nézzük a klasszifikációs fa esetét. Az egyik alapvető algoritmus az ID3, valamint ennek továbbfejlesztett változata C4.5. Ezek az algoritmusok az információs entrópiát, valamint az információ nyereséget [12] használják a szeparációra. E mellett egy gyakran használt algoritmus a CART, Gini-index (GI) szeparálóval.

### 3.1.1. ID3

Az ID3 (Iterative Dichotomiser 3) algoritmust Ross Quinlan 1986-ban publikálta. Ez még csak egy kezdetleges verzió volt, itt az algoritmus csak kategorikus változókat tudott kezelni. Heterogenitás mértékeként az információs entrópiát használja. Az algoritmus először meghatározza az információs entrópiáját az eredményváltozónak, utána minden magyarázóváltozóra kiszámítja az információs entrópiát, majd az információ nyereséget. Ahol a legnagyobb a nyereség, vagyis a legkisebb az entrópia ott fogjuk elágaztatni a fát. Ezt Quinlan (1986) a cikkében egy időjárásokról készült adatbázissal szemléltette, egy kis példán keresztül én is így teszek az átláthatóság kedvéért.

Quinlan által használt adatok:

Az információs entrópia H(S) a bizonytalanságot méri az S adathalmazban, a következő módon határozható meg:

$$H(S) = \sum_{c \in C} -p(c) log_2 p(c)$$

, ahol

- S az adathalmaz, amire az entrópia számolva van
- C az S-ben lévő cél változók halmaza. Itt C = {P, N}
- p(c) a relatív gyakorisága a C-ben lévő változóknak.

[12] Angol terminológia: Information Gain

| Outlook | Temp. | Humidity | Windy | Class |
|---|---|---|---|---|
| sunny | hot | high | false | N |
| sunny | hot | high | true | N |
| overcast | hot | high | false | P |
| rainy | mild | high | false | P |
| rainy | cool | normal | false | P |
| rainy | cool | normal | true | N |
| overcast | cool | normal | ture | P |
| sunny | mild | high | false | N |
| sunny | cool | normal | false | P |
| rainy | mild | normal | false | P |
| sunny | mild | normal | true | P |
| overcast | mild | high | true | P |
| overcast | hot | normal | false | P |
| rainy | mild | high | true | N |

1. táblázat. Forrás: Quinlan (1986)

Az információ nyereség IG(A, S) méri a különbséget az entrópiában mielőtt és miután szétbontottuk S-t „A” attribútum (magyarázó változó) szerint.

$$IG(A, S) = H(S) - \sum_{t \in T} p(t)H(t)$$

, ahol

- H(S) az információs entrópiája az S halmaznak
- T részhalmazai a szétválasztott S halmaznak az „A” attribútum által

$$S = \bigcup_{t \in T} t$$

- p(t) a relatív gyakorisága a t részhalmaznak, tehát a magyarázóváltozó egy fajta elemének

- H(t) az információs entrópiája a t részhalmaznak

Az adathalmaz eredményváltozója jelen esetben a Class, először ennek számoljuk ki az információs entrópiáját a képlet alapján:

$$H(S) = -(5/14)log_2(5/14) - (9/14)log_2(9/14) = 0,94$$

Ezek után minden egyes magyarázóváltozóra számítsuk ki az információs entrópiát, majd az információ nyereséget. Nézzük a windy esetet, 8 false (ezen belül 2 no és 6 yes) és 6 true (ezen belül 3 no és 3 yes) szerint történik meg a szétbontás, tehát:

$$H(false) = -(2/8)log_2(2/8) - (6/8)log_2(6/8) = 0,811$$

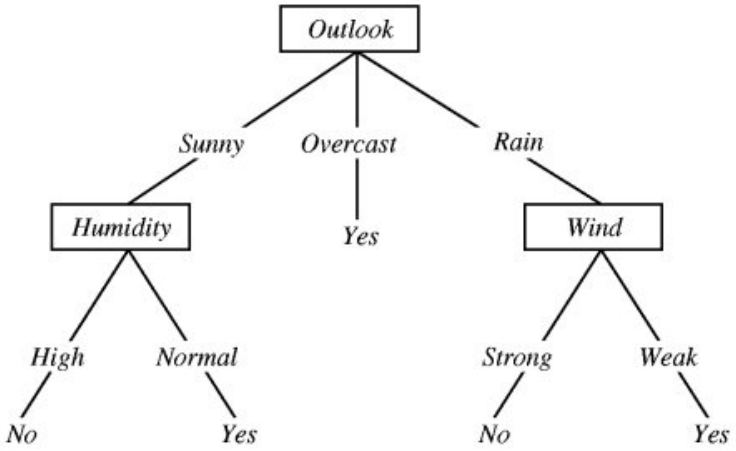
$$H(true) = -(3/6)log_2(3/6) - (3/6)log_2(3/6) = 1$$

$$p(false) = 8/14 \quad és \quad p(true) = 6/14$$

$$IG(Windy, S) = 0,94 - (8/14 \cdot 0,811 + 6/14 \cdot 1) = 0,048$$

Ez alapján a többi magyarázó változóra is ki kell számítani az IG-t, ahol a legnagyobb a nyereség aszerint ágaztatjuk el a fát. Az első elágaztatás neve gyökér, a további elágaztatásokat ágaknak, illetve a végső részt leveleknek nevezzük. Ezt a metódust utána addig folytatjuk, amíg a kívánt fát megkapjuk. Továbbiakba olyan adatokkal kell számolni, amelyek az ágaztatásnak megfelelőek, tehát ha itt elágaztatjuk true és false szerint, akkor a true oldalán úgy számolunk tovább, hogy a Windy/True-ra szűrt adatbázis marad meg. Végül egy ilyen fát kapunk: [13]

1. ábra. Quinlan példa fája




[13]Forrása: `http://science.slc.edu/~jmarshall/courses/2005/fall/cs151/lectures/decision-trees/figure3.1.jpg`, (Utolsó letöltés: 209.11.09.)

Ennek az ID3-nak a továbbfejlesztése a C4.5, ami szintén Ross Quinlan által lett kifejlesztve. Legnagyobb előnyei elődjéhez képest, hogy képes folytonos és kategorikus változókat is kezelni és lefut hiányzó adatok esetén is.

ID3 regressziós esetében az információ nyereséget lecseréljük a Standard Deviation Reductionra (SDR) és az entrópiát a szórásra. Első lépésként számoljuk ki a szórását az eredményváltozónak, majd a magyarázó változók szerinti bontásban a szórásokat.

$$SDR(S, A) = S(S) - S(S, A)$$

$$S(S, A) = \sum_{t \in T} p(t)S(t)$$

, ahol S(S) az eredményváltozó szórása és S(S, A) az A attribútum szerinti bontásból adódó szórások, ami az attribútum lehetséges értékei szerinti megbontásból fakadó szórások relatív gyakoriságokkal súlyozott összegeként áll elő. A levelek értéke végül az ágaztatások mentén lévő szűrések elvégezése utáni megmaradó adatbázis eredményváltozójának az átlaga lesz.

Ha folytonos magyarázóváltozóval van dolgunk, akkor bizonyos érték mentén ketté választjuk őket. Ezt Quinlan (1996) alapján a sorba rendezett értékek felezőpontjában nézzük meg, tehát $(v_i + v_{i-1})/2$ helyeken, így n elemű minta esetén n-1 helyen kell vizsgálnunk és e szerint számolni a megfelelő értéket, így már alkalmazható a korábbi technikákra.

### 3.1.2. CART

A másik népszerű módszer klasszifikációs fák képzésére a CART (Classification And Regression Tree) algoritmus, ez 1984-ből származik Breiman és szerzőtársaitól, Ők a Gini-indexet használják szeparálás céljára. A Gini-indexet a következő módon határozzuk meg:

$$G = 1 - \sum_{t=0}^{k} P_t^2$$

, ahol k a kategóriák száma és $P_t$ a kategóriák relatív gyakorisága. Kiszámoljuk az eredmény változóra a Gini-indexet, majd a magyarázó változók katagóriák szerinti bontásokra is számítjuk a GI-t, a bontásból számított GI-ket utána még a bontás relatív gyakoriságával súlyozva összeadjuk. A lehetséges bontások közül amelyik a legkisebb, vagyis legmesszebb van a bontás nélküli GI-től, ott válasszuk el a fát.

Továbbiakban a kalkuláció és a folyamat mechanizmusa ugyan az, mint az előző algoritmusnál.

### 3.1.3. Ensemble metódusok

Döntési fáknál nagyon gyakori, hogy a jobb teljesítmény érdekében nem csak egy fát kreálunk, hanem többet. Így növelhetjük a modellezésünk hatékonyságát. Egyik elterjedt módszere a Bagging (Bootstrap Aggregating) (Breiman, 1996) mely esetben visszatevéses mintavétellel veszünk egy akkora mintát, amekkora az adott mintánk, erre építjük rá a modellt majd a modelljeink előrejelzett értékeiből átlagolással vagy valamilyen más módszerrel meghatározzuk a kimenetelt. Döntési fákál használt ensemle metódust Véletlen erdőnek nevezzük. Itt alkalmazunk Feature Bagging módszert is, ami annyiban különbözik az előbb tárgyalt Baggingtől, hogy nem csak a megfigyeléseinkből veszünk véletlen mintát, hanem az attribútumokból is. Ez gyakran $\sqrt{p}$, amennyiben a magyarázóváltozóink száma p.

### 3.1.4. Gradient Boosting Machine

Egy közkedvelt és gyakorlatban sokszor igen erős predikciós erővel rendelkező algoritmus a Gradient Boosting Machine, 1999-ből származik J. H. Friedman tollából. Klasszifikációra és regresszióra is alkalmas módszer. Szintén ensemble kategóriába tartozik, annyi különbséggel, hogy itt a korábbi predikciók hibája van modellezve. Mindig a korábbi hibára építünk egy újabb fát, amíg elérjük a kezdéskor megadott iterációszámot. Hamar felmerülhet a túlillesztés problémája ezeknél a metódusoknál, ennek kivédésére alkalmaznak itt egy learning rate-et, ez egy 0 és 1 közötti szám, amekkora súlyban beszámítjuk a modellezett hibák értékét a korábbi becslésekhez. Az itt megjelenő problémát „bias-variance trade off" -nak nevezzük, vagyis, hogy a modell komplexitása függvényében az előrejelzéseknek mennyire nagy a varianciájuk vagy a pontosságuk. Ha növeljük a modell komplexitását, akkor csökken az előrejelzés torzítása, de a túlillesztés miatt nő az előrejelzések varianciája. A formalizálás során (Firedman, 2002)-et követem végig. [14]

[14]Mindenképpen érdemes még megemlíteni Joshua Starmer-t, aki a StatQuest oktatási célzattal létrejött youtube csatorna alapítója. A GBM-hez (és sok más egyébhez) kiváló értelmező anyagokat készít, amelyeket én is előszeretettel felhasználtam a munkám során. Elérhető itt:

Legyen egy n elemű mintánk $(\mathbf{x_i}, y_i)_{i=1}^n$ és egy differenciálható veszteségfüggvény $L(y_i, F(x))$. Ezt a számítások megkönnyítése érdekében rögzítsük $L(y_i, F(x)) = \frac{1}{2}(y_i - \hat{y_i})^2$, learning rate legyen $0 < \alpha < 1$. Első lépésként határozzuk meg a kiinduló értékeket.

$$F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$$

, ahol gamma a becsült értékünk, $F_0(x)$-et Friedman "initial guess"-nek nevezi, tehát ez lesz az első becslésünk az értékekre. A megadott veszteségfüggvény mellett ez az eredményváltozók átlaga lesz. Jelölje M a modellezésre használt fák számát, legyen m a fák indexe. Ezek után kalkuláljuk ki a reziduálisokat. A következő jelölje az i-edik minta elem, m-edik fánál lévő hibatagját:

$$r_{i,m} = -\Big[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\Big]_{F(x)=F_{m-1}(x)}$$

A belső tagunk a gradiens, innen kapta a módszer a nevét. Az egész a reziduális lesz a kényelmesen megválasztott veszteségfüggvényünk miatt. Ezt gyakran pseudo-reziduálisnak nevezik, mivel értéke a veszteségfüggvénytől függ. Ezek után modellezzük a reziduálisokat egy döntési fával. Jelölje $R_{j,m}$ az m-edik fa j-edik levelének az elemeinek a halmazát. A fa előrejelzéseknek az értékeit megkapjuk a következő módon.

$$\gamma_{j,m} = argmin_\gamma \sum_{x_i \in R_{i,j}} L(y_i, F_{m-1}(x_i) + \gamma)$$

Ez a levelek előrejelzett értékeinek az átlagát adja meg. Az új értékünk az előző becslés és az új reziduálisokra épített modell becslése szorozva a learning rate-el:

$$F_m(x_i) = F_{m-1}(x_i) + \alpha\gamma_{j,m} I(x_i \in R_{j,m})$$

Ez lesz az új becslésünk és kezdjük az új reziduálisok kalkulálásától újra, ezt a ciklust M-szer ismételjük meg.

Ezzel a fa alapú modellek végére értem. Összességében érdemes megtekinteni a folytonosságát a fa alapú modellek fejlődésének. A kiindulópont a döntési fa volt. Ha visszatevéses mintavételezéshez nyúlunk, akkor kapjuk a Bagging módszert. A Bagging tovább fejleszthető véletlen erdővé, amennyiben az attribútumokat is véletlen választjuk meg. A Boosting módszer (ezt önmagában nem részleteztem) annyiban más a baggingtől, hogy itt a fák összefüggnek, szekvenciálisan épülnek. Végül a GBM

www.youtube.com/watch?v=3CC4N4z3GJc (Utolsó letöltés: 2020.03.02.)

egy boosting módszer, amely gradiens csökkenés módszerét alkalmaz az optimalizáció során.

## 3.2. Mesterséges neurális hálózatok

Kezdetleges Neurális hálók már a 1950-es és az 1960-as években elkezdtek fejlődni Frank Rosenblatt munkássága nyomán, Warren McCulloch és Walter Pitts korábbi eredményeit felhasználva.

Az elnevezés a biológiai neuronok mintáján alapul, miszerint egy neuronnak számos dendritje és egy axonja van. A dendritek, mint a bemeneti adatok, a neuron a számító egység és az axon a kimenet, a biológiai jeltovábbító rendszer a szinapszis feleltethető meg, annak, ahogy az input változók végig futnak a neurális hálón és a bemeneti ingerületek valamilyen kimeneti hatásként transzformálódnak át.

A működésük megismerése előtt célszerű jelezni, hogy a neurális hálók akkor hatékonyak, ha sok tanuló adatunk áll rendelkezésre. Gyakran nevezik „black box"-nak, mivel értelmezésük nehéz, ezért inkább előrejelzésre használatosak, mint az adatok mögötti kapcsolat feltérképezésére. Ebből látható, hogy elég körülményesek, azonban olyan problémákat is képesek megoldani és nagyszerű eredményeket elérni, amiket másfajta tanuló algoritmusok nem. Illetve olyan területeken is használhatók eredményesen, amit a többi algoritmus kezelni sem tud, ezeket Convolutional Neural Network-nek (CNN) nevezik, itt az input lehet akár kép vagy hang is. [15]

### 3.2.1. Matematikai modellje

A modell bemutatásához az Uppsalai Egyetem Statistical Machine Learning anyagát használtam fel, a neurális hálókról szóló előadás Niklas Wahlströmtől hangzott el.[16]

**1. Definíció.** *Egy neurális háló alatt a következő nem lineáris függvényt értjük* $y = f(x, \theta) + \epsilon$*, ahol az* $x$ *az input vektor* $y$ *az output és* $\theta$ *a paraméterek.*

[15] Ezekről részletesebben itt nem ejtünk szót, de érdemes gondolni a korábbiakra, amikor a biztosító mesterséges intelligencia alapú kárfelmérést alkalmaz, ilyen esetekben egy CNN remekül használható.

[16] Előadás anyagok megtalálhatóak a kurzus honlapján, itt: `http://www.it.uu.se/edu/course/homepage/sml`, Utolsó letöltés: 2019.09.28

**2. Definíció.** *Lineáris regressziós modell x input vektor és y output között*

$$y = \beta_0 + \sum_{j=1}^{p} x_j \beta_j + \epsilon = \beta^T \boldsymbol{x}$$

*,ahol* $\beta = (\beta_0, \beta_1, ..., \beta_p)^T$ *súlyvektor és* $\boldsymbol{x} = (x_0, x_1, ..., x_p)^T$ *inputok,* $\epsilon$ *hibatag.*

**3. Definíció.** *Általánosított lineáris regressziós modell, ha a lineáris regressziónak vesszük egy nem lináris transzformáltját. Tehát az előbbi jelöléseket megtartva:*

$$y = \sigma(\beta^T \boldsymbol{x}) + \epsilon$$

*. A* $\sigma$ *függvényt aktiváló függvénynek hívjuk.* [17]

Legyen $z = \sigma(\beta^T \mathbf{x})$, tehát $y = z + \epsilon$.

Egy neurális hálót végül sok általánosított lineáris regresszió kombinálásával kapunk meg. Az inputokat bemeneti rétegnek, a köztes részeket, ahol a súlyozások és a transzformáció történik rejtett rétegnek (rétegeknek) és az ouptut részt kimeneti rétegnek nevezzük.[18]

Jelölje

$$h_1^{(1)} = \sigma(\beta_{01}^1 + \sum_{j=1}^{p} \beta_{j1}^{(1)} x_j)$$

$$h_2^{(1)} = \sigma(\beta_{02}^1 + \sum_{j=1}^{p} \beta_{j2}^{(1)} x_j)$$

$$...$$

$$h_M^{(1)} = \sigma(\beta_{0M}^1 + \sum_{j=1}^{p} \beta_{jM}^{(1)} x_j)$$

,ahol $\mathbf{h} = [h_1, h_2, ..., h_M]^T$ és $b^{(1)} = [\beta_{01}^{(1)} \beta_{02}^{(1)} ... \beta_{0M}^{(1)}]$

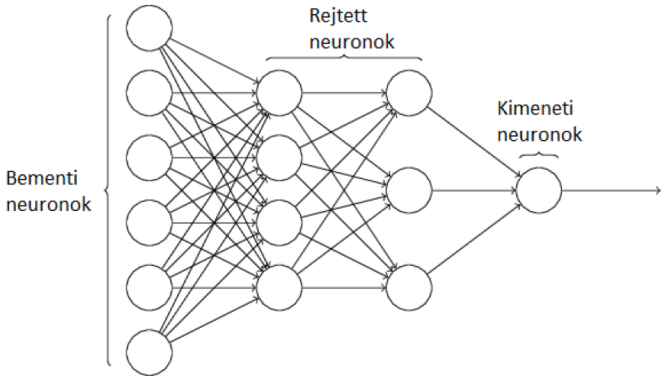
$$W^{(1)} = \begin{bmatrix} \beta_{11}^{(1)} & ... & \beta_{1M}^{(1)} \\ . & ... & . \\ \beta_{p1}^{(1)} & ... & \beta_{pM}^{(1)} \end{bmatrix}$$

$W^{(1)}$ az első réteg előtti súlyok, $h_l^{(1)}$ az első réteg kimeneti értékei, ahol l = 1, 2 ..., M. Szemléltetve:

[17] Angol terminológia: Activation function

[18] Angol terminológia: Input layer, Hidden layers, Output layer

2. ábra. Neurális háló modellje




Itt a 2.ábrán egy két rejtett rétegű neurális háló látható, természetesen lehet több rejtett rétege is, akkor azokat jelöljük $W^{(i)}$ és $h_l^{(i)}$, ahol l = 1, 2,..., M és i = 1, 2, ..., N. Ha ez az N nagy, akkor mély neurális hálónak nevezzük. [19] Az N-edik réteg futtása és kiértékelése után kapjuk meg a kimenetet vagyis z értékét.

A neurális hálók alkalmazhatóak mind klasszifikációra mind regresszióra, csupán megfelelően kell megválasztani az aktiváló függvényeket.

### 3.2.2. Aktiváló függvény

A neurális hálózatokkal való modellezést sokan művészetnek tartják, ez abból is adódik, hogy a modellező itt több szabad teret kap, mint más algoritmusoknál. Ez egyfelől nehezíti a munkát és tapasztalatot igényel, másfelől aki jól használja a lehetséges eszközöket, az kiváló eredményeket tud elérni. Egyik ilyen egyéni döntés, amit a modellező saját maga választhat meg az aktiváló függvény. Itt bemutatom a legnépszerűbbeket, amiket erre a célra szoktak használni.

- Identitás. $\sigma(x) = x$
- Sigmoid. $\sigma(x) = 1/(1 + e^{-x})$
- ReLU. $\sigma(x) = max(0, x)$
- Bináris.

$$\sigma(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

[19] Angol terminológia: Deep neural network

- Hyperbolikus tangens (tanh). $\sigma(x) = (e^x - e^{-x})/(e^x + e^{-x})$

Fontos, hogy az aktiváló függvénynek differenciálhatónak kell lennie, mivel az optimalizálásra gradiens csökkenés (gradient descent) módszerét és backpropagationt fogok használni. Mély neurális hálóknál a sigmoid és a tanh nem kedvelt, mivel a deriváltjaik nagyon kicsi értéket adnak, így a grandiens csökkenés módszerével elvégzett optimalizáció igen lassú és nehézkes, ezt "vanishing grandient" problem-nek nevezik. Ebből adódóan manapság a ReLU egy igen közkedvelt aktiváló függvény, amit a rejtett rétegeknél használnak. Kimeneti rétegnél az aktiváló függvény választása, attól függ, hogy az eredményváltozónk milyen értékeket vehet fel.

### 3.2.3. Gradiens csökkenés módszere

A gradiens csökkenés módszere a legnépszerűbb optimalizációs technikája a mesterséges neurális hálózatoknak. Kiindulásképpen szükségünk van egy veszteség függvényre [20], ami alapján meghatározható a különbözően paraméterezett modellek jósága és egymáshoz való viszonya. A leggyakoribb veszteség függvények a

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

vagy a

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

.[21] Természetesen sok egyéb más veszteség függvény is választható. Jelölje a veszteség függvényünket $L(\theta_0, \theta_1, ..., \theta_n)$, értéke a paraméterek és az adataink függvénye. Az algoritmus indításánál a paraméterek önkényesen vagy véletlenszerűen vannak beállítva. Ezek után számítsuk ki a veszteség függvény grandiensét. Kapjuk a gradiens vektort: $\langle \frac{\partial L}{\partial \theta_0}, \frac{\partial L}{\partial \theta_1}, ..., \frac{\partial L}{\partial \theta_n} \rangle$. Learning ratenek vagyis tanuló rátának nevezzük azt a hiperparamétert, ami meghatározza, hogy mekkora lépéseket teszünk az optimalizáció során. Értékét mi választjuk meg, jelöle: $\eta$. Végezetül a következő formula adja meg a frissített paraméterünket:

$$\theta_i^{\text{Új}} = \theta_i^{Régi} - \eta\frac{\partial L}{\partial \theta_i}$$

A $\eta\frac{\partial L}{\partial \theta_i}$ kifejezést lépésköznek [22] nevezik.

[20] Angol terminológia: Loss function

[21] MSE - Mean Squared Error és MAE - Mean Absolute Error

[22] Angol terminológia: step size

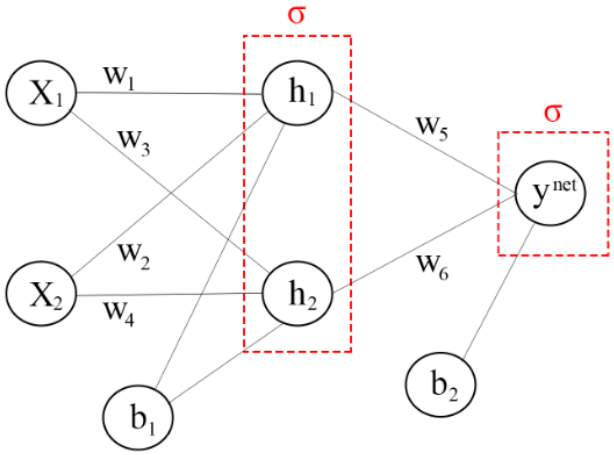
Ezt addig folytatjuk, amíg a lépésköz egy nagyon kicsi érték (pl.: 0.0001) alá csökken vagy beállítunk egy maximális iterációs számot.

### 3.2.4. Backpropagation

A gradiens csökkenés módszere után rátérhetek a backpropagation-re, amely az egyik legközkedveltebb optimalizációs eszköz a neurális hálók esetében. A logikája a gradiens csökkenést követi, vagyis kellenek nekünk a veszteség függvény különböző paraméterek szerinti deriváltjai és ebből konstruáljuk a frissített paramétereket, amíg nem találunk optimális megoldást. Az előbb bemutatott gradiens csökkenés egy univerzális optimalizáció, a backpropagation-nel kiegészítve ez már alkalmazható neurális hálókon is. Neurális hálóknál a paramétereket visszafelé frissítjük, tehát a leghátsó rejtett réteg súlyait majd így tovább és tovább az első réteg súylaihoz.

Forward propagationnek nevezzük azt, amikor a meglévő paramétereinkkel ellátott neurális hálón végig futatjuk a tesztelő adatainkat, ez által megkapjuk a becsléseket, valamint ebből a veszteség függvényünk értékét. Ez után következik a backward rész, amit a következő egyszerű példával szemléltetünk, tekintsük a következő neurális hálót: [23]

3. ábra. Példa modell




Ebből célunk megkapni a $\frac{\partial L}{\partial w_i}$, hogy a gradiens csökkenés módszerét alkalmazhassuk. A következő egyenleteink állnak fenn:

$$w_1 x_1 + w_2 x_2 + b_1 = h_1$$

[23] Hasonló példa található meg a következő oldalon: `https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/`, Utolsó letöltés: 2019.09.29

$$w_3 x_1 + w_4 x_2 + b_1 = h_2$$

, az inputok befutnak az első rétegbe.

$$h_1^{out} = \sigma(h_1)$$

$$h_2^{out} = \sigma(h_2)$$

, az aktivizáló függvény transzformálja a bemeneti értékeket.

$$w_5 h_1^{out} + w_6 h_2^{out} + b_2 = y^{net}$$

$$\sigma(y^{net}) = y^{out}$$

, a rejtett rétegből fut az output rétegbe.

Ezekből adódóan a változás, alkalmazva kétszer a lánc szabályt, a rejtett réteg súlyait módosítjuk:

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial y_{out}} \frac{\partial y_{out}}{\partial y^{net}} \frac{\partial y^{net}}{\partial w_5}$$

Utána az első réteg súlyai következnek:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{out}} \frac{\partial y_{out}}{\partial y^{net}} \frac{\partial y^{net}}{\partial h_1^{out}} \frac{\partial h_1^{out}}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

Majd az előző alfejezetben látott módon a súlyok változása:

$$\eta \frac{\partial L}{\partial w_5} \text{ és } \eta \frac{\partial L}{\partial w_1}.$$

## 3.3. SVM

A support vector machine vagyis a támaszvektor-gépek algoritmust az 1960-as években alkotta meg Vladimir Vapnik és a későbbi változatokban Corinna Cortes munkásságát kell még kiemelni. Az SVM egy gépi módszer, valószínűségszámítási elemeket nem tartalmaz. Közkedvelt, mivel klasszifikációra, regresszióra és klaszterezésre is használható. Predikciós ereje gyakran felveszi a versenyt egy többrétegű neurális hálóval vagy egy véletlen erdővel.

### 3.3.1. Lineáris SVM

Az alapötlet, hogy egy optimálisan megválasztott hipersíkkal elválasszuk az adatainkat. Támaszvektoroknak azokat a pontokat nevezzük, amik a legközelebb esnek a hipersíkhoz, a hipersíkot úgy akarjuk megválasztani, hogy maximális legyen a távolság a támaszvektoroktól, ezáltal a legjobb szeparációt biztosítsa.

Legyen n elemű mintánk $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)$, $\mathbf{x_i} \in \mathbb{R}^p$, $y_i \in \{-1, 1\}$, $\mathbf{w} \in \mathbb{R}^p$ keresett paraméter vektor, $b \in \mathbb{R}$ torzítás. A hipersíkunk egyenlete:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

Legyen d a távolság a támaszvektorok között. Ekkor $\frac{d}{2}$ a támaszvektorok és a hipersík távolsága. Értelemszerűen a legoptimálisabb, ha a hipersík a két class közé esik, mivel ha az egyikhez közelebb lenne, és pont a hipersík másik oldalára esne egy előrejelzendő pontunk, akkor az abba az osztályba kerülne, amelyiktől távolabb helyezkedik el. A támaszvektorokat úgy határozzuk meg, hogy

$$\mathbf{w}^T\mathbf{x} + b \geq 1$$

$$\mathbf{w}^T\mathbf{x} + b < -1$$

Az optimális hipersík, ahol d maximális. Egy pont egyenestől való távolsága: $dist = |ax_0 + by_0 + c|/\sqrt{a^2 + b^2}$ , tehát $\frac{d}{2} = |\mathbf{w}^T\mathbf{x} + b|/||\mathbf{w}|| = 1/||\mathbf{w}||$ Ebből a támaszvektorok távolsága $d = 2/||\mathbf{w}||$ Célunk a támaszvektorok közötti távolság maximalizálása, tehát ezzel ekvivalens, ha minimalizáljuk $||\mathbf{w}||$ -t, ezt módosítsuk a $\frac{1}{2}||\mathbf{w}||^2$ problémára, mivel így egy kvadratikus felületünk lesz és ez esetben az optimális megoldás globális minimum is lesz.

Problémánk összesítve:

$$min_{\mathbf{w},b}\frac{1}{2}||\mathbf{w}||^2$$

$$s.t.$$

$$\begin{cases} \mathbf{w}^T\mathbf{x} + b \geq 1, ha\ y_i = +1 \\ \mathbf{w}^T\mathbf{x} + b < -1, ha\ y_i = -1 \end{cases}$$

Az utóbbit rövidíthetjük:

$$y_i(\mathbf{w_i}^T\mathbf{x}) \geq 1$$

Ezt Lagrange multiplikátor módszerrel megoldva, kapjuk a megfelelő paramétereket. A Lagrange függvény:

$$min_{\mathbf{w},b}\mathcal{L} = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1} \lambda_i y_i(\mathbf{x_i}\mathbf{w} + b) + \sum_{i=1} \lambda_i$$

$$s.t.$$

$$\forall i \ \ \lambda_i \geq 0$$

Kapjuk:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1} \lambda_i y_i x_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1} \lambda_i y_i = 0$$

Tehát eredményink:

$$\mathbf{w} = \sum_{i=1} \lambda_i y_i x_i$$

$$\sum_{i=1}^{n} \lambda_i y_i = 0$$

Az előbbi megfogalmazás sok esetben nagyon szigorú, nem engedi a pontokat a határok közé. Írjuk fel a problémát rugalmas határokkal, ahol definiálunk változókat, amik „büntetik a határsértést". (T. Hastie, R.Tibshirani, J. Friedman, 2009, 418 – 420 . o.) [24] Korábbi jelöléseinket megtartva az alábbi módon fogalmazhatjuk át a problémát, itt már a módosított, számítások számára egyszerűbb kvadratikus optimalizációs problémát fogalmaztuk meg:

$$min_{\mathbf{w},b}\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i$$

$$s.t.$$

$$y_i(\mathbf{w_i}^T\mathbf{x}) \geq 1 - \xi_i, \forall i$$

$$\xi_i \geq 0 \ , \sum_{i=1}^{n} \xi_i \leq C, \ \ C \in \mathbb{R}$$

### 3.3.2. Nem-lineáris SVM

Az előbbi csak abban az esetben működik, ha lineárisan szeparálhatóak az adataink, ha ez nem áll fenn, akkor transzformáljuk az adatainkat egy magasabb dimenzióba valamilyen függvénnyel, hogy szeparálhatóvá tegyük őket. Azonban némely

[24] A kétfajta metódust gyakran „soft margin" és „hard margin"-nak emlegetik a szakirodalomban.

esetben nehéz megtalálni azt a függvényt, amire szükségünk van, erre a problémára alkalmazzák a kernel trükköt.[25] Megjegyzendő, hogy az előbb látott optimalizálási problémának felírható a duálisa, ami alapján szintén levezethető a megoldás. Most tekintsük a duálisát, hogy lássuk a magfüggvény megjelenését. Transzformáló függvény legyen $\phi(x)$ és a magfüggvényünk K(x,y). (T. Hastie, R.Tibshirani, J. Friedman, 2009, 132 – 133 . o.)

$$\mathcal{L} = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \phi(x_i) \phi(x_j)$$

Kernelt használva: $K(x_i, x_j) = \phi(x_i)\phi(x_j)$ Ez csökkenti a számolás komplexitását. Gyakorlatban gyakran használt magfüggvények:

- Polinomiális. $K(x_i, x_j) = (x_i x_j + 1)^p$
- Gaussi. $K(x_i, x_j) = exp{-}(x_i x_j)^2/2\sigma^2$
- Sigmoid. $K(x_i, x_j) = tanh(\eta x_i x_j + \upsilon)$
- RBF kernel. $K(x_i, x_j) = exp\{-\gamma(x_i - x_j)^2\}$

### 3.3.3. SVR

Eddig klasszifikációs esetben láttuk az SVM működését, regresszióra is meg van konstruálva a módszer. Ezt support vector regression-nek nevezzük. Korábbi jelöléseinket megtartjuk. Felveszünk egy $\epsilon$ értéket, amin belül megengedhető, másképp fogalmazva 0 büntetéssel vesszük figyelembe az értéket. A hipersíktól $\epsilon$ távolságnál messzebb lévő értékeket jelöljük $\xi_i$-vel. ($\xi_i = hipersíktól\ való\ távolság - \epsilon$) A konstrukció erősen hasonlít egy lineáris regresszióhoz, a $\mathbf{w}$-ket $\beta$-nak megfeleltetve. A probléma ez esetben így formalizálható:

$$min_{\mathbf{w},b}\, \mathcal{L} = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i + \xi_i^*$$

$$s.t.$$

$$\mathbf{w}^T\mathbf{x_i} + b - y_i \leq \epsilon + \xi_i$$

$$y_i - \mathbf{w}^T\mathbf{x_i} - b \leq \epsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0, \forall i$$

[25] Angol terminológia: Kernel trick

## 3.4. Haladó regressziós módszerek

Végül két egyszerűbb, ámde közkedvelt eljárást mutatnék be röviden, amik a szakirodalomban és a gyakorlatban is igen gyakran használatos eszközök, ha megtekintjük a gépi tanulás aktuáriusi alkalmazásait.

### 3.4.1. Multivariate adaptive regression spline - MARS

A MARS egy nem-lineáris modellezési eljárás, amit Jerome H. Friedman publikált először 1991-ben. Jól kezeli a sok dimenziós problémákat. Alapjában véve egy általánosított stepwise lineáris regresszióként értelmezhető a szerzők szerint. (Hastie, Tibishirani és Friedman, 2009) A MARS a következő páronkénti lineáris bázis függvényeket használja a regresszióban: $(x-t)_+$ és $(t-x)_+$, ahol a + a pozitív egészrészt jelöli.

Meghatározzuk a t pontokat, ezeket csomóknak nevezzük [26], majd ezek a csomók mentén bontjuk szét a regressziót különböző függvényekre. Az egy csomóhoz eső függvények halmaza:

$$\mathcal{C} = \{(X_j - t)_+, (t - X_j)_+\}_{\{t \in x_{1j}, x_{2j}, \ldots, x_{Nj} \text{ és } j=1,2,\ldots,p\}}$$

,ahol $X_j$ a j-edik prediktor és N elemű mintával rendelkezünk.

A modell építési stratégia olyan, mint egy forward stepwise lineáris regresszió, azonban itt a $\mathcal{C}$ elemeit használjuk vagy a szorzatukat. Tehát,

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X)$$

, ahol minden $h_m(X)$ egy függvény $\mathcal{C}$-ből, a szorzatuk vagy egy konstans.

### 3.4.2. Elastic Net

Az elastic net egy predikcióra és változó szelekcióra is alkalmas eljárás, ami lényegében egy lineáris vagy egy logisztikus regresszió, azzal a bővítéssel, hogy a paraméterek L1 és L2-beli büntetéssel épülnek be a modellbe.[27] Így a kevésbé releváns változókat kiszűri és az optimális "büntető" hiperparaméter megválasztásával a torzítás és a variancia megfelelően csökkenthető a becslésünkben.

[26] Angol terminológia: knot

[27] Ha csak szimplán L1-et használunk, akkor a LASSO-t, ha L2-t akkor a Ridge regressziót kapjuk vissza.

Legyen egy n elemű mintánk és p darab magyarázóváltozónk, magyarázóváltozók és a megfigyelések mátrixa legyen szokásos módon $\mathbf{X}$. Az eredményváltozó vektora y. Legyen $\lambda_1$ és $\lambda_2 > 0$, ekkor az elastic net: (Zou és Hastie, 2005)

$$\mathbf{L}(\lambda_1, \lambda_1, \beta) = |y - \mathbf{X}\beta|^2 + \lambda_2|\beta|^2 + \lambda_1|\beta|$$

, ahol

$$|\beta|^2 = \sum_{j=1}^{p} \beta_j^2,$$

$$|\beta| = \sum_{j=1}^{p} |\beta_j|.$$

A megoldásunkat a $\mathbf{L}(\lambda_1, \lambda_1, \beta)$ minimalizálása adja $\beta$ szerint.

# 4. Modellek összehasonlítása

Most hogy már tudjuk hogyan működnek, nézzük meg mik az érvek bizonyos modellek mellett és ellen a használat során, illetve hogy mikre kell figyelnünk, amikor alkalmazzuk valamelyiket. A döntési fa, a neurális háló és a SVM esetében részint támaszkodhatunk Bhavsar és Ganatra (2012) elemzésére ebben a témakörben.

- Lineáris regresszió: *Előnyök*: Nincs hiperparaméter, könnyű értelmezni és elmagyarázni, valamint a Lasso és Ridge módszerekkel (meg persze az Elastic nettel) leszűkíthetjük a modelleket csak a releváns változókra és elkerülhetjük a túlillesztést. A predikciós ereje sok esetben gyengébb mint más módszereknek, de a változók hatását nagyon jól leírja. *Hátrányok*: Nem-lineáris esetben kevésbé működik jól, nehéz lehet a megfelelő kereszt-szorzatok és magasabb rendű magyarázóváltozók optimalizálása.

- MARS: *Előnyök*: Rugalmasabb, mint egy lineáris regresszió, interpretálása könnyű. Automatikus változó szelekciót hajt végre, túlillesztés ritkán jelentkezik. *Hátrányok*: Teljesítménye általában elmarad a GBM és a Véletlen erdőtől.

- Döntési fa: *Előnyök*: Könnyű interpretálhatóság. Egy szépen vizualizált fa a menedzsment előtt is megállja a helyét. Működése emberi gondolkozásra hasonlít, így elmagyarázni is könnyű. Nem-lineáris esetekben is jól működik, nagy mennyiségű adatot is tud kezelni. Ezen kívül futási ideje gyors. *Hátrányok*: A

túlillesztést megfelelő hiperparaméterekkel kordában kell tudni tartani. Nem túl robosztus, vagyis kis változás a tanuló adathalmazban, a kimenetelt nagyban befolyásolhatja. Általában rosszabb a teljesítménye, mint az ensemble metódusoknak.

- Véletlen erdő: *Előnyök*: Jó teljesítményt szoktak nyújtani, robosztusak a túlillesztés ellen, nem követlenek meg lineáris kapcsolatot és a változók fontossága is kalkulálható velük. *Hátrányok*: Egy SVM-nél gyorsabb a futtatása, de összességében elég lassú a kiszámítása, valamint az interpretálása nehézkes lehet.

- SVM: *Előnyök*: Lineáris és magfüggvények segítségével nem-lineáris eset is kezelhető. Általában kiváló a teljesítményük, legkevésbé hajlamos a túlillesztésre és robosztus. *Hátrányok*: A kalkulációja nagyon nehézkes és lassú, illetve nehezen interpretálható a működése.

- GBM: *Előnyök*: Kiváló teljesítmény, rugalmasság a sok hiperparaméter miatt. *Hátrányok*: Nehéz megérteni a működését és ez által interpretálni is, nem robosztus, könnyen túlilleszkedik. Hiperparaméterek optimalizációja nehéz lehet, gyakorlatot igényel.

- ANN: *Előnyök*: Jól tanul a nem-lineáris adatokra, nagy mennyiségű adatot kiválóan kezel. Kép, szöveg és hang inputokat is tud kezelni. (Deep learning témakör) *Hátrányok*: Nagy mennyiségű adatra van szükség (Ott jön ki az ereje a többi algoritmushoz viszonyítva), sok hiperparaméter, nehéz az értelmezése.

# 5. Modellezés

Az adatok személyessége miatt elég nehéz interneten megfelelő minőségű biztosítási adatot találni. A kaggle oldalán [28] fellelhető egy adatbázis, ami biztosítási károkat reprezentál. A leírás szerint minden egyes sor egy biztosítási káreseményt jelent, az adatok az Allstate biztosítótól származnak. [29] Előnye a hatalmas adatmennyiség, 188 318 megfigyelést és 131 változót tartalmaz. Hátránya, hogy ekkora

[28] `https://www.kaggle.com/c/allstate-claims-severity/data` (Utolsó letöltés: 2019.10.29.)

[29] Az Allstate egy Egyesült Államokbeli biztosító, központja Northfield Township-ben található, 1931-ben lett alapítva. Jelenlegi CEO: Thomas J. Wilson

adatbázist már bizonyos számítógépek nem is tudnak kezelni, illetve a modellek futási ideje igencsak hosszú lehet. A sok magyarázó változó igen jó perdikciót biztosíthat, illetve elég jól reprezentál az adathalmaz egy komplexebb adatelemzési problémát, talán olyan módszerek is kiválóan alkalmazhatóak, mint a neurális hálók, amelyek igen sok adatot követelnek meg. A nagy adathalmaz jól kivetíthető egy komolyabb biztosító állományának, tehát elég valóság közeli példáról van szó. A személyes adatok védelme miatt az adathalmaz anonim, a változók neveit nem közlik, csak a mérési skálájukat, illetve a kategorikus változók fajtáit. Numerikus változók esetében standardizált értékek szerepelnek. A kárnagyság szerepel a végén, ezt „loss"-nak nevezték.

Az anonim változók miatt a perdikciós erő nem fog gyengülni, csak az értelmezés lesz hiányos, azonban a gyakorlatban egy valódi biztosítónak az adatbázisán ezzel nem lesz probléma. Tételezzük fel, hogy egy gépjármű biztosítási adathalmazról van szó. Ennél a biztosítás típusnál igencsak sok magyarázó változó lehetséges. Bemutatnék néhány lehetséges példát, ami meghatározhatja a díjat:[30]

- Kategorikus nem-gépjármű jellegű változó: Szerződő lakhelye, Szerződő foglalkozása, Szerződő családi állapota, Szerződő iskolai végzettsége, Szerződő neme [31], Természetes vagy nem természetes személy...etc.

- Kategorikus gépjármű jellegű változó: Hajtóanyag, Gépjármű típusa, Üzemeltetés módja, Gépjármű színe. . . etc.

- Numerikus nem-gépjármű jellegű változó: Szerződő kora, Szerződő gyerekeinek a száma, Jogosítvány megszerzése óta eltelt napok. . . etc.

- Numerikus gépjármű jellegű változó: Gépjármű kora, Műszaki vizsga óta eltelt napok, Teljesítmény, Hengerűrtartalom, Gépjármű használat ideje, Büntető pontok száma, Korábbi károk száma. . . etc.

A modellezéshez és elemzéshez az R programcsomagot használtam. A modellezés során a következő alfejezetben bemutatott lépéseket hajtom majd végre, ez általánosságban minden hasonló modellezési problémánál egy követhető „recept".

[30] Az alábbi változók többsége magyar biztosítók gépjármű biztosítási dokumentációiból származik.

[31] Díjkalkulációnál nem vehető figyelembe.

## 5.1. A modellezés lépései

1. *Adatok bemutatása és előkészítése*: Az adathalmaz származása, megbízhatóságának vizsgálata, nagyságának az ismertetése. Célváltozónk meghatározása, értelmezése. Magyarázóváltozók megtekintése, mértékegységek megállapítása, outlierek és a hibás értékek eltávolítása, vizualizálás után az első intuíció megszerzése. Modellezés fő kérdésének meghatározása és összevetése a rendelkezésre álló adathalmazzal, értelmezhető-e a kutatási kérdés az adott adathalmazon. A már első ránézésre is irreleváns oszlopok elhagyása. (pl.: id) Ezen kívül a hiányzó adatok beazonosítása, valamilyen módszerrel való kezelése. Ha szükséges, akkor az adatokon transzformáció végrehajtása. Az adathalmaz mellé egy magyarázó jegyzet készítése, ami értelmezi az oszlopneveket, ezt meta adatoknak nevezzük. [32]

2. *Adatok szétválasztása*: Tesztelő és modellező adatok elkülönítése. Megfelelő split-ratio kiválasztása. A modellező adatokon fogjuk futtatni a modelljeinket, a tesztelőn kiértékeljük őket. Ez egy alapvető és könnyen értelmezhető megközelítés. Ennek vannak szofisztikáltabb módszerei is, amiket a későbbiekben használni fogok: Kereszt-validáció és Ismételt kereszt-validáció.

3. *Modellépítés*: Megfelelő technikák kiválasztása. Hiperparaméterek optimalizációja, modell futtatás, megfelelő modellszelekciós kritériumok meghatározása és a mutatók kiszámolása. A legjobb modell kiválasztása és értelmezése.

4. *Interpretálás és vizualizálás*: Az eredmények értelmezése, bemutatása és magyarázata.

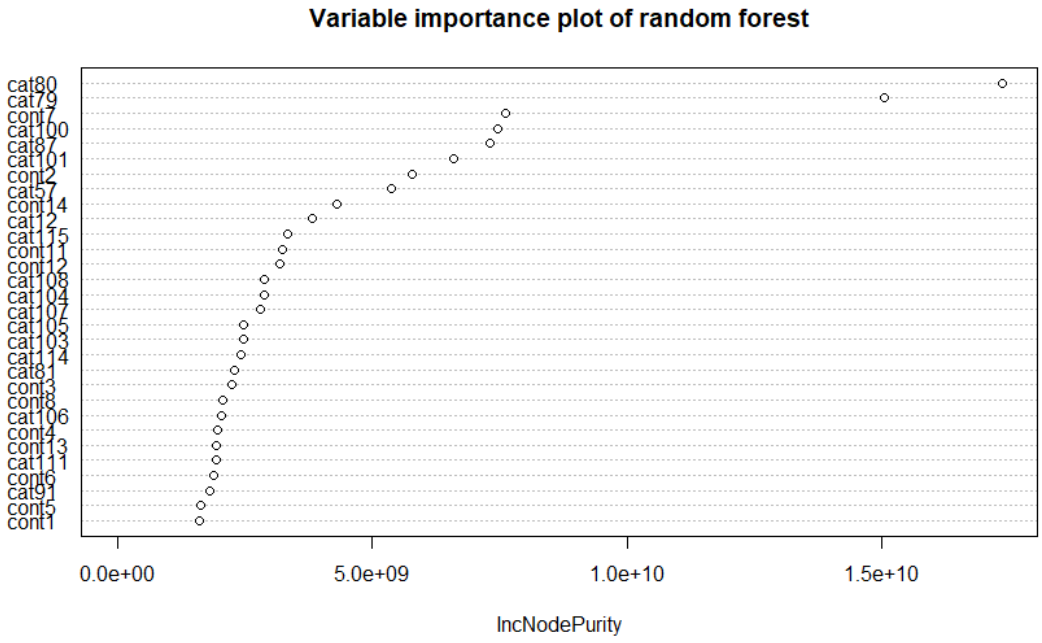
## 5.2. Az adatok előkészítése

Az előző alfejezetben már megemlíettem néhány információt az adathalmazról. Ekkora adat fájlhoz megfelelő számítógépes kapacitás nem állt rendelkezésemre, ezért módosítani kell a méretén. Először is a kiinduló adatokból vettem egy véletlen mintát, ez az alap adatok 5%-a lesz, így maradt 9273 megfigyelésünk és 131

[32]Sok adat tudós vallja, ami a leírás terjedelméből is látszik, hogy a legnehezebb és legtöbb időt igénylő feladat az első lépés, a megfelelő adatok beszerzése és modellezhető állapotra hozása.

változónk. A változók számát is csökkentenünk kell, mivel a 131 változóból 116 kategorikus, némelyik igen sok kategóriával rendelkezik, ezt azt jelenti, hogy dummy változók készítése után 1000 feletti változó számot kapunk a modellekben, ami nagyon sok. [33] Ezeket autómatikusan eltávolítottam, mivel az R némelyik algoritmusa, csak bizonyos számú kategória számot tud kezelni. [34] Ezt követően futtattam egy véletlen erdőt a teljes állományra, majd tekintsük meg a változók fontossága nevű diagrammot:

4. ábra. Változók fontossága




Ez egy jó kiinduló pontot szolgáltat arra, hogy melyik változókat tartsuk meg és melyeket hagyhatunk el. Ez még tartalmazza a 100+-os kategória számú változókat, azok kivételével használjuk ezeket a változókat a továbbiakban. Így kaptuk meg a végső adatállományt, amivel dolgozni fogunk, ennek mérete 9273 megfigyelés és 22 változó, amik a "legfontosabbnak" bizonyultak, ebből egy a "loss" célváltozónk. Magyarázóváltozók közül 6 numerikus és 15 kategorikus. [35]

Ebben az adatázisban most szerencsés esettel találkoztunk, nincsenek hiányzó

[33] Akad nem egy köztük, amelynél akár 100 feletti kategória szám is előfordul.

[34] Pl.: randomForest kód maximálisan 53 kategóriát tud szétbontani.

[35] A végső adatbázis méret próbálgatások során alakult ki, hogy a rendelkezésre álló számítógépeknek mennyi idő szükséges az algoritmusok futtatására úgy, hogy még hatékonyan lehessen dolgozni.

adatok, azonban ez inkább a kivétel, mint a szabály. Ha hiányzó adatokba futunk, akkor háromféle megoldás lehetséges a gyakorlatban:

1. Hiányzó adatot tartalmazó sor kitörlése. Ez nagyon drasztikus, nem ajánlott megoldás. Abban az esetben használható, ha túl sok adat hiányzik egy megfigyelésből, így az nagyon bizonytalanná válik és nincs értelme becsülni, sok információ többletet nem hordoz magában. Eltávolítható az oszlop is, ha túl sok megfigyelési egység hiányzik belőle és nem tekintjük túlságosan releváns-nak, így az adott megfigyeléseink többi értéke megmarad.

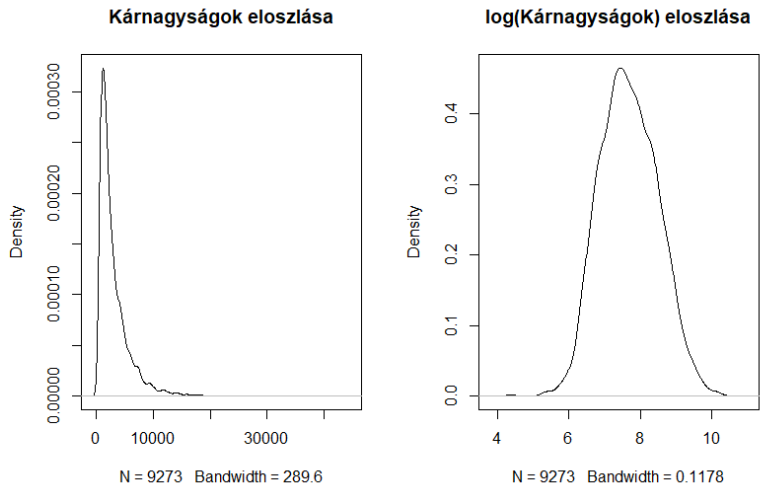
2. Valamilyen egyszerű mutatóval való becslés. (pl.: átlag, medián) Egyszerű módszer, mégse veszítjük el azt az információ többletet, amivel a megfigyelésünk hozzájárul az adatbázishoz.

3. Valamilyen prediktív módszerrel a többi magyarázó változót felhasználva megbecsüljük a hiányzó értékeket. Ez a legkomplexebb, ámde a leghatásosabb eljárás.

## 5.3. Az adatok bemutatása

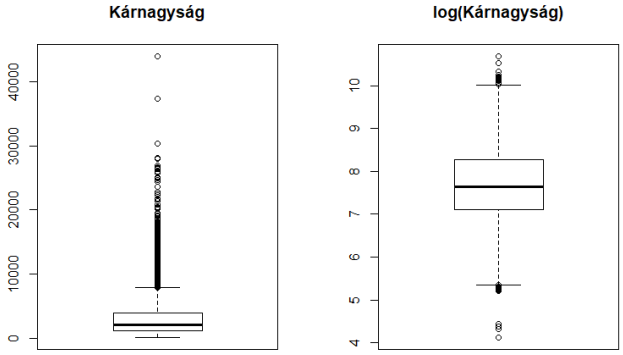
A következőkben bemutatok néhány leíró statisztikai diagrammot, mutatót az adatokról. Ezek segítségével a hibás adatbevitelek (nem racionális értékek bizonyos változók helyén) könnyen kiszűrhetők, illetve a modellezés során hasznos, ha van egy rálátásunk az adatainkra. A legfontosabb változó számunkra a kárnagyság.

5. ábra. Célváltozók szemléltetése



Első ránézésre elégedettek lehetünk ezekkel az adatokkal. Tapasztalataink alapján származhat ténylegesen egy biztosító állományából. Kárnagyság tekintetében gyakran tapasztalunk egy jobbra elnyúló csúcsos eloszlást, kivehető néhány nagyobb káresemény is az eloszlás farok részén. A jobbra elnyúló eloszlás miatt vegyük és használjuk a logaritmikus transzformációját a modellezés során, így egy szimmetrikusabb eloszlást kapunk, amivel könnyebb dolgozni. [36]

6. ábra. Célváltozók szemléltetése - Box plot



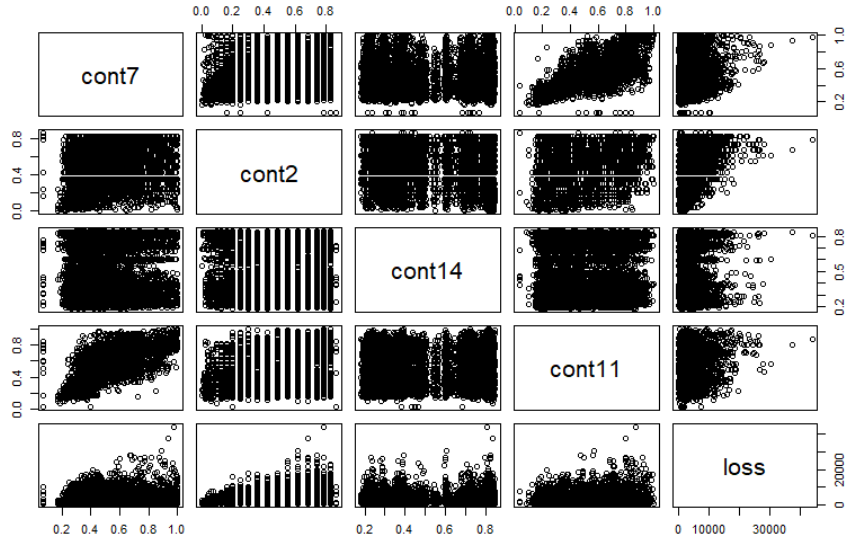

A box ploton jobban kivehetőek az extrém értékek, a logaritmizálás után is tapasztalhatunk kilógásokat, de egyenletesebbnek néz ki így az eloszlás. Transzformáció nélkül a nagy kiugró értékek jóval nagyobb számban fordultak elő.

A célváltozó megismerése után nézzük meg a magyarázóváltozókat, egy pont diagram jól szemlélteti a köztük és a célváltozó közötti kapcsolatot. [37]

[36] Metaadat hiányában és az ábrákból kiindulva levonható a következtetés, hogy az adatok egy biztosító egyéves állományából származnak.

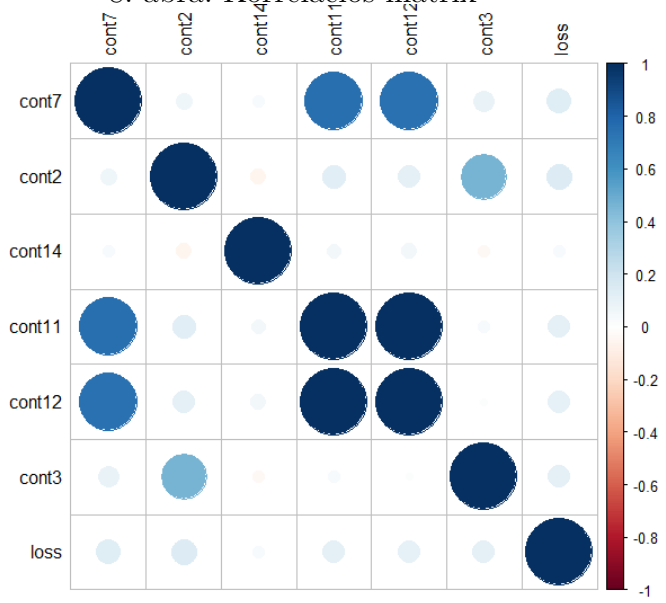
[37] Az átláthatóság miatt csak a 4 legjelentősebb változót ábrázoljuk és a célváltozót.

7. ábra. Változók pont diagrammja




Ezekből már következtethetünk a közöttük lévő lineáris kapcsolatra, de sokkal könnyebben átlátható, ha ábrázoljuk a korrelációs mátrixát a numerikus változóinknak.

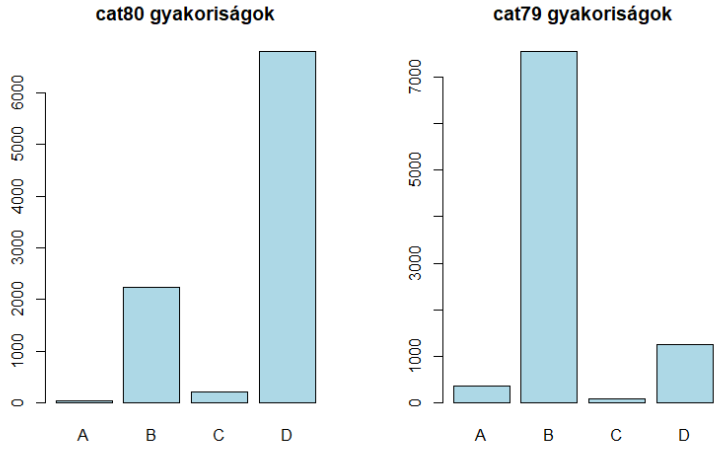
8. ábra. Korrelációs mátrix




Cont11 és cont12 változók között találhatunk nagyon erős pozitív lineáris kapcsolatot, ezen kívül a cont7 változó még pozitívan korrelál az előbb említettekkel, ez szépen kirajzolódik a pontdiagrammon is. Közepes erősségű pozitív korreláció a képen látható esetekben néhány változónál szerepel, negatív korreláció nem igen fedezhető fel.

Végül nézzük meg a két legjelentősebb kategorikus változó gyakorisági jellemzőit.

9. ábra. Oszlopdiagram - cat80 és cat79




## 5.4. A modellezés eredményei

A korábban bemutatott algoritmusok segítségével feldolgozom az adatokat, célom, hogy az előbb bemutatott adatbázisból a "loss" változó minél jobb perdikciója lehetséges legyen, a változó logaritmusát vettem a jobb eloszlás miatt. A modellezés során kereszt-validációt fogunk használni, ez egy jobb képet ad a modellek teljesítményéről, mint egy sima tanuló és tesztelő adathalmaz szétválasztás. A kereszt-validáció során 5 felé bontjuk majd az adatainkat, az 5-ös kereszt-validáció egy gyakran használt szám, lehetne nagyobb is, de az adathalmaz mérete miatt ezzel most megelégedünk a számítási kapacitások problémája miatt. Gyakorlatban gyakran használnak még 10 felé bontást, vagy ami mégjobb meghatározást adhat egy-egy modell teljesítményéről az ismételt kereszt-validáció. A modellek összehasonlítására a korábban bemutatott MSE gyökét, vagyis az RMSE-t fogjuk használni.

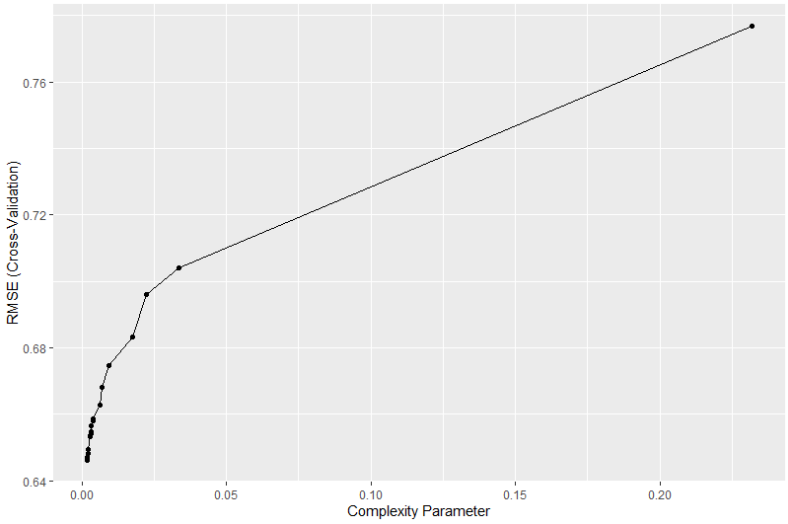
### 5.4.1. Döntési fák

Döntési fából kétfajta algoritmust tekintettünk meg az ID3-t és a CART-ot, egyik az Információs entrópiát, a másik a Gini-indexet használja az algoritmusában. Futtathatjuk mindkét fát az R-ben, nézzük először az ID3 alapút. Az egyetlen általunk meghatározható paraméter itt a cp[38], egy ábra segítségével könnyen szemléltethető,

[38]Komplexitás paraméter

hogy bizonyos cp mellett, milyen RMSE-t produkál a modell.

10. ábra. ID3 Komplexitás paraméter

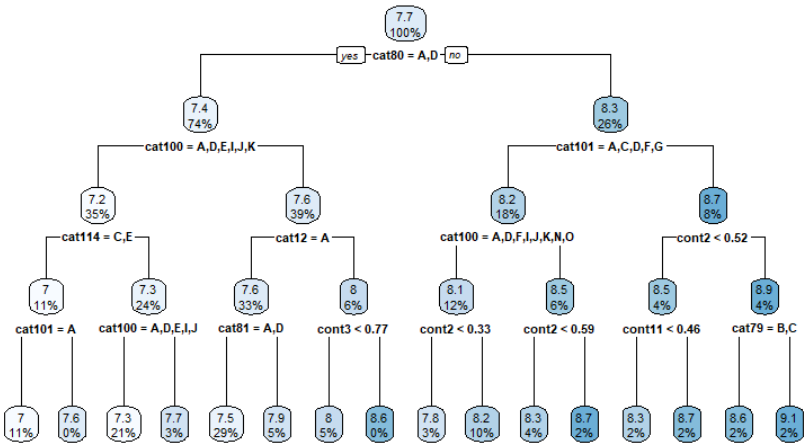



25 kipróbált cp után a 0.001877547 érték további csökkentésével már nem javult a RMSE értékünk, sőt növekszik újra, így ezt tekintjük optimálisnak a modell esetében. A kereszt-validált modellek RMSE-inek az átlaga 0,6461807, ezt nevezzük a modellünk teljesítményének. Az RMSE értékek variabilitását majd a modellek összehasonlítása során bemutatom a box plotokon.

A CART algoritmust használva eredményeim megegyeznek a ID3-val, különbség nem mutatkozik ezen az adatállományon. [39] Végül nézzük meg az ábrázolt fát, azonban a legoptimálisabb túl nagy a vizualizációhoz a szakdolgozat keretei között, ezért egy visszanyesettett mutatok be.

[39]Valószínűleg ez azért van, mert a prediktorok elkülönülnek és a kalkuláció változtatása is megtartja a hasonló bontást a fákban.

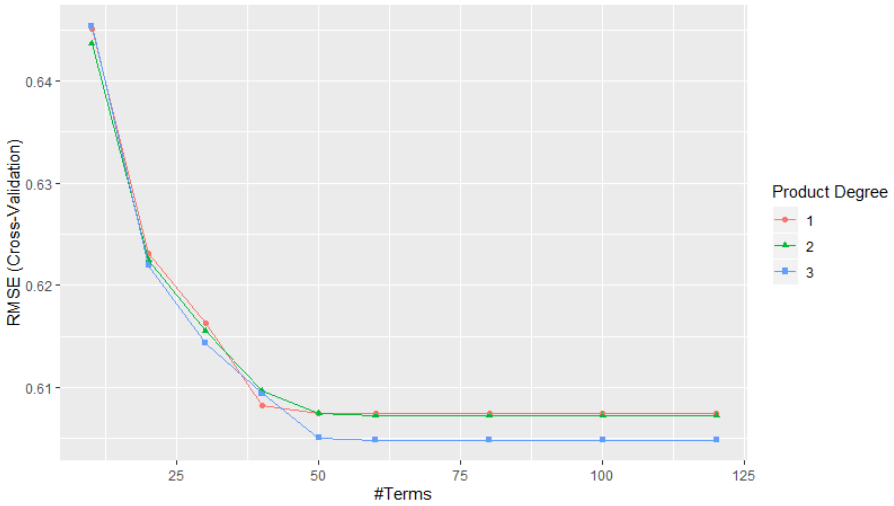
11. ábra. Döntési fa




### 5.4.2. MARS

A MARS regressziónál az R-ben kétfajta hiperparamétert optimalizálhatunk, először is a degree-vel meghatározhatjuk a maximális fokszámát az interakcióknak, másodszor a csomópontok számát állíthatjuk be. Ezek tesztelése után a következőket láthatjuk:

12. ábra. MARS

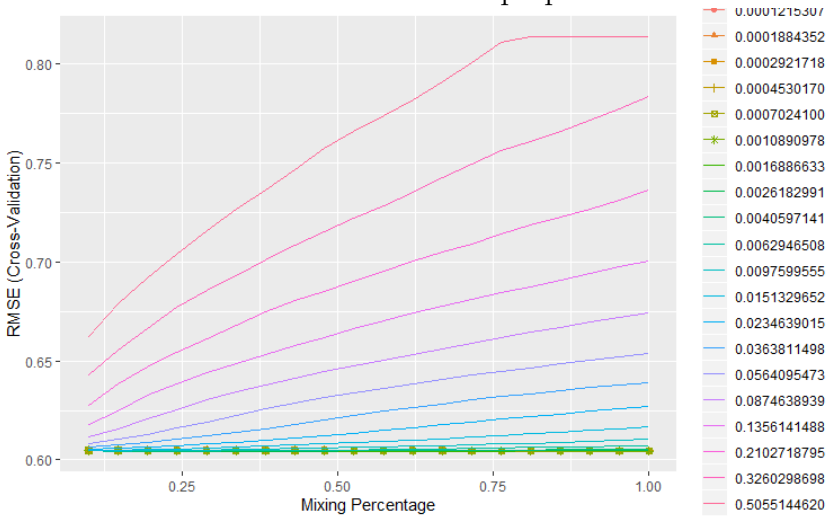



60 körüli csomók után már nem jelentős a teljesítmény növekedés, illetve a 3-as maximális fokszám adja a legjobb eredményeket, 3-nál nagyobb fokszámmal a modell komplexitása miatt nem próbálkozunk . A modell teljesítménye 0,6048705.

### 5.4.3. Elastic Net

Az elastic net futtatásánál két hiperparamétert tudunk optimalizálni, $\alpha$ [40] adja meg, hogy az L1 és L2 milyen arányban keveredik egymással, illetve $\lambda$ [41] határozza meg a büntetés mértékét a hozzáadott változók után. Futtattam az R paraméter generálójával 400 különböző kombinációt, a következő ábra mutatkozik:

13. ábra. Elastic net hiperparaméterek



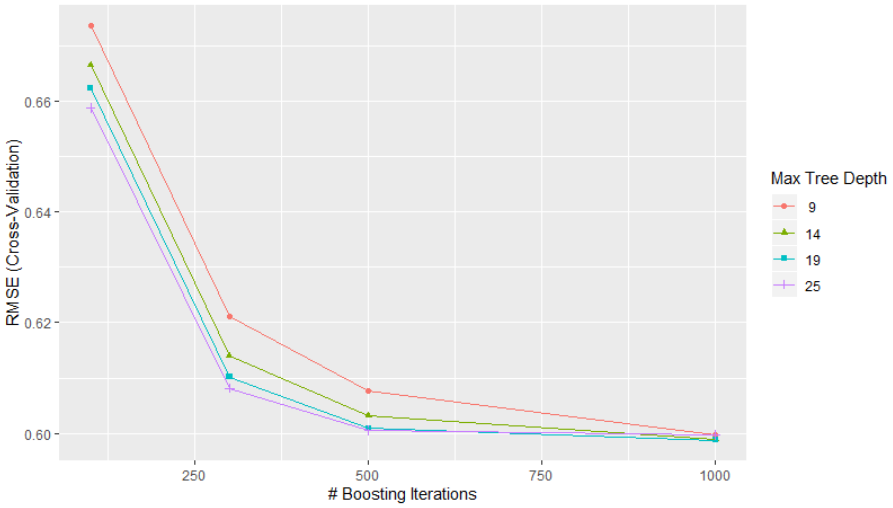

Az optimális paraméter kombináció $\alpha = 0,3368421$ és $\lambda = 0,002618299$. E mellett a minimális az RMSE, aminek értéke 0,6044618.

### 5.4.4. Gradient Boosting Machine

Hiperparaméter optimalizáció szempontból a GBM-nél a fák számát kell meghatároznunk, a learning rate-et [42], valamint a maximális levélszámot megadhatjuk a fáknak [43], így kontrollálni tudjuk a mélységüket. Ezen kívül még meghatározható, hogy egy levél minimálisan hány mintaelemet tartalmazzon, tehát ha kevesebb lenne, akkor már nem bontja tovább. [44] Ezt a legutóbbi hiperparamétert 20-nak állítottuk be, úgy vélem az még megfelelő szám egy levélen. Ezen kívül futtattam a modellt 100, 300, 500 és 1000 döntési fával, valamint 0,01-es learning rate-el. Maximális levélszámot 9, 14, 19 és 25-el teszteltem.

[40] Mixing percentage
[41] Regularization parameter
[42] Ezt az R shrinkage-nek nevezi
[43] R kódban: interaction.depth
[44] R kódja: n.minobsinnode

14. ábra. GBM hiperparaméterek




Az eredményekből látszik, hogy a 19-es levélszám és az 1000-es fa szám bizonyult a legjobbnak. A learning rate meghatározása nehezebb feladat volt, itt próbálkoztam 0,001 és 0,1-es értékekkel is a végsőn kívül. Első esetben rosszabb eredményeket kaptunk, mivel így túl kis mértékben veszi figyelembe a hiba modellezést az algoritmus. A másik, 0,1-es esetben pedig már nagyon túlilleszti a modellt és a növekvő faszám rontja a teszt adathalmaz értékelését. Az optimális modell teljesítménye 0,5988005.

### 5.4.5. Véletlen erdő

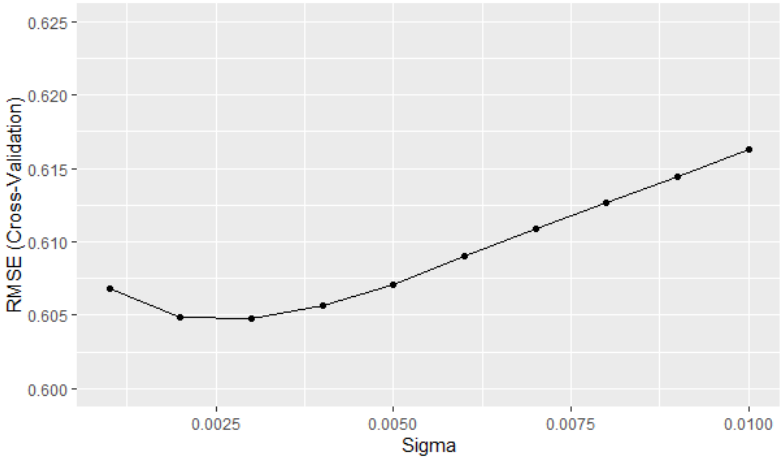
Gyakran használatos hüvelykujj szabály a véletlen erdőnél, hogy a benne szereplő fák számát 1000-nek válasszuk meg, én is így tettem, azonban már olyan 500 fától a végső eredmények nem nagyon változnak. Az R kódban még megadhatjuk a változók számát, amenyit véletlen generál minden egyes fához. Ezt az értéket végül 18-ra állítottam be, nagy különbségek a modell erejében nem mutatkoztak hasonló paraméterezés esetében, de ez bizonyult a legjobbnak. Így a modell teljesítménye 0,6124371.

### 5.4.6. SVM

SVM-nél egy lineáris és egy nem-lineáris modellt próbáltam ki. Nem-lineáris esetben RBF kernelt használtam. Lineáris SVM-nél a költség paramétert tudjuk beállítani az R kódban, ezt a korábbi esetekhez hasonlóan teszteléssel határozzuk meg, végső soron a C = 1 beállítás adta a legjobb eredményeket. A nem-lineáris kernel

használatakor, mint korábban láthattuk az RBF kernel szerkezetét, a költségen kívül még a szigma hiperparamétert kell optimalizálnunk. Az optimális költség érték 0,5 és a szigma 0,003, a hiba változását a szigma függvényeként az alábbi ábrán szemléltethetjük:

15. ábra. RBF SVM szigma értékek



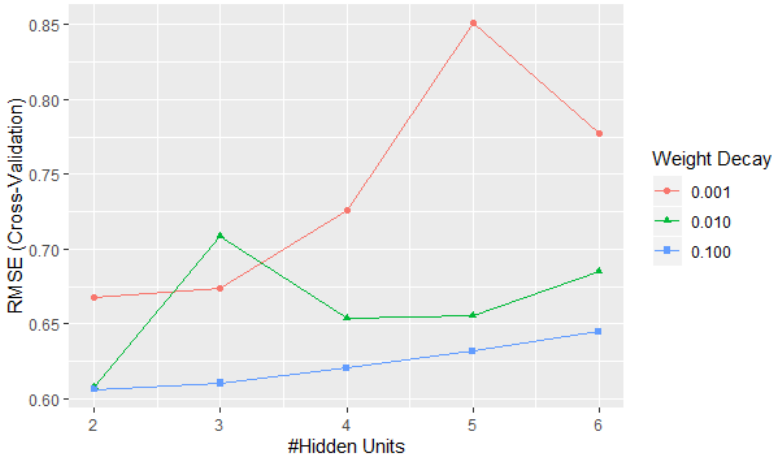

A végső eredmények a lineáris SVM-nél 0.6067556 és a nem-lineárisnál 0,6047688.

### 5.4.7. Mesterséges neurális hálózatok

Neurális hálóknál az R kétfajta hiperparamétert kínál fel. Egyrészt beállíthatjuk a rejtett rétegben lévő neuronok számát, másrészt a learning rate-et. A maximális iteráció számot 1000-nek vettem. Az optimális paraméterek 2 rejtett réteg és 0,1-es learning rate. [45] Ezen kívül érdemes megjegyezni, hogy az ANN-nek sok tanuló adatra van szüksége, ott mutatkozik meg inkább az ereje a többi modellhez képest, itt ugyan csak a számítási kapacitás problematikája merül fel. Ezek figyelembe vételével nézzük a hiperparamétereinket:

[45] Sajnos az R paraméterezése ANN tekintetében elég korlátolt. Sok dolgot láthattunk az elméleti összefoglalóban, ami a programcsomaggal nem adható meg. Ezek az állítások az nnet package-re vonatkoznak. Az R neruralnet egy jobb választás, ezzel a kóddal már operálhatunk a többrétegűség és az aktiváló függvények körben, azonban ez nem kompatibilis a caret csomaggal, ami biztosítja számunkra a kereszt-validációt. A többi modellel konzisztens akartam lenni, ezért ezt a funkciót elvetettem és most az nnet-re hagyatkoztam korlátai ellenére.

16. ábra. ANN hiperparaméterek




A megfelelő beállítások után a legjobb modell teljesítménye 0,6060737.

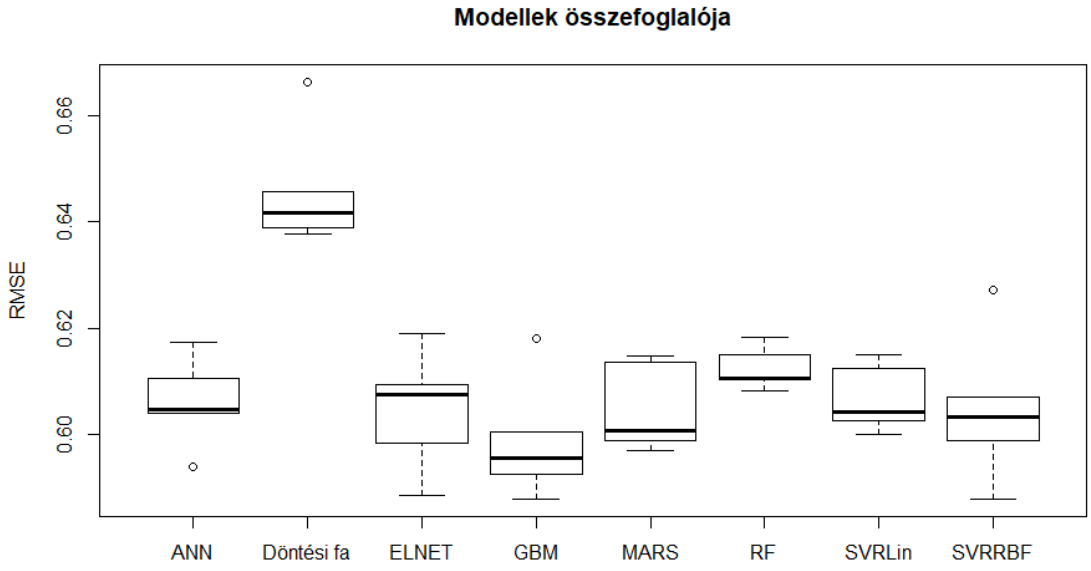
*Megjegyzés*: Neurális hálók tekintetében szerintem a Python jobb megoldást kínál a modellezés, programozás és egyszerűség szempontjából. Az R mellett elkészítettem Pythonba is ugyanazon az adathalmazon a kereszt-validált modellt. Ehhez a sklearn MPLRegressor funkcióját használtam. Ennek segítségével tudunk operálni az akiváló függvények, a rejtett rétegek illetve a neuronok számának tekintetében. A kipróbált aktiváló függvények az identitás, tanh és a ReLU volt. Minden tekintetben a ReLU bizonyult a legjobbnak. Az optimális learning rate, úgy mint az előző modell esetében is, 0,1 lett. Végül 3 rejtett réteget használtam, amik sorjában 6-9-6 neuront tartalmaznak. Ennek a modellnek az átlagos teljesítménye: 0.6131016. Meglepő eredménynek tűnhet, hogy a szofisztikáltabb megoldás rosszabb teljesítméynt adott. Itt az eltérés adódhat a kereszt-validált adahalmaz véletlen ingadozásából. Mindenesetre én ezt a módszert a nagyobb mozgástér miatt stabilabbnak tartom. A továbbiakban az R által nyújtott ANN-ra fogok hivatkozni, amikor a nerurális háló modelljéről ejtek szót, mivel megszeretném tartani a modellek konzekvenciáját. A Pythonba épített modellt csupán azért tartottam fontosnak demonstrálni, hogy látható legyen az összes olyan hiperparaméter kezelés, ami megjelent az elméleti összefoglalóban is.

## 5.5. A modellek kiértékelése

Modelleztük az adatbázist a korábban bemutatott módszerekkel, optimalizáltuk a hiperparamétereket és egy-egy modellcsaládból kiválasztottuk azt, ami a legjobb

teljesítményt nyújtotta az adathalmazon, láttuk külön-külön az eredményeket, most nézzük meg egybe:

17. ábra. Modellek teljesítménye




Első ránézésre a döntési fa gyenge teljesítménye szúrhat szemet, nagyon elszakad a többi modelltől. Az előnyökre/hátrányokra visszagondolva, azonban ez nem egy meglepő eredmény. Tudjuk, hogy például az ensemble technikák jóval hatékonyabbnak bizonyulnak sok esetben. A legjobb modellunk a GBM lett, nehéz nála megtalálni az optimális hiperparaméter beállításokat, de most láthatjuk, hogy megéri az időt és az energiát. Ő volt az egyetlen modell, aminek az átlagos teljesítménye 0,6 alá betudott menni. A legjobb és a legrosszabb modellt leszámítva a többi eredménye elég hasonló. ANN-nél feltehetően finomabb beállításokra lenne szükség és nagyobb adatbázisra, de már ilyen feltételekkel is látszik az ereje. A két SVM/SVR modellünk között sok különbség nem tapasztalható, az RBF kernel kicsit jobban teljesített, de az eredményinek a szórása nagyobb. A lineáris SVM tejesíménye robosztusabbnak tűnik. A RF teljesítménye elég gyenge volt az adathalmazon, míg az Elastic Net az egyszerűsége ellenére elég megbízható becsléssel szolgál számunkra.

# 6. Monte Carlo Szimuláció

A Monte carlo szimuláció egy közkedvelt eszköz a pénzügyi matematikában, aminek a segítségével sztochasztikus modellezések kimeneteleit szimulálhatjuk. Kiválóan szemlélteti a módszert Wilmott (2000, 581 . o.), aki egy opció értékének a szimulálásán keresztül világítja meg a módszer lényegét. Ahogy Wilmott is kiemeli, a módszer lényege, hogy van egy kifejezésünk, aminek vannak sztochasztikus elemei és ezek viselkedését próbáljuk megérteni sok futtatással.

A következőkben a modellezési eredményeimet felhasználva egy biztosítási portfóliót fogok szimulálni. A portfólió kialakítása a korábban használt „nagy" adathalmazból történik. Véletlenszerűen visszatevés nélkül kiválasztottam 1000 káreseményt, ezen káreseményekhez tartozó ügyfelek fogják reprezentálni a portfóliónkat. Megengedtem, hogy olyan megfigyelés is része lehessen a portfóliónak, amit a modellezés során használtunk, mivel életszerűen a biztosítónál is maradhatnak sokáig olyan ügyfelek, akik már régóta ott kötnek biztosítást, így a különböző fejlesztések során felhasználhatták az adataikat. Ezek után természetesen csak azokat a változókat tartottam meg, amik a végső modellben is szerepeltek. Az ügyfélparamétereink így megfelelőek, azonban éves károkra van szükség az árazás során. Így az interneten fellelhető statisztikákat felhasználva feltételezéssel éltem a kár darabszámra vonatkozóan. A statista.com közzétett egy statisztikát az évenkénti "motor insurance" kárgyakoriságára 2016-ig bezáróan, a legfrissebb, vagyis a 2016-os adatot fogjuk használni, mi szerint az egy főre eső várható kár darabszám 0,13. [46]

Ezt követően meghatározzuk ügyfeleink nettó díjait. A legjobb modellnek titulált GBM-et felhasználva megbecsüljük a várható kárnagyságokat az egyes szerződésekre vonatkozóan, majd ezeket a becsült értékeket megszorozzuk az előbb meghatározott várható kár darabszámmal. Tehát a nettó díja az $i$-edik szerződésnek:

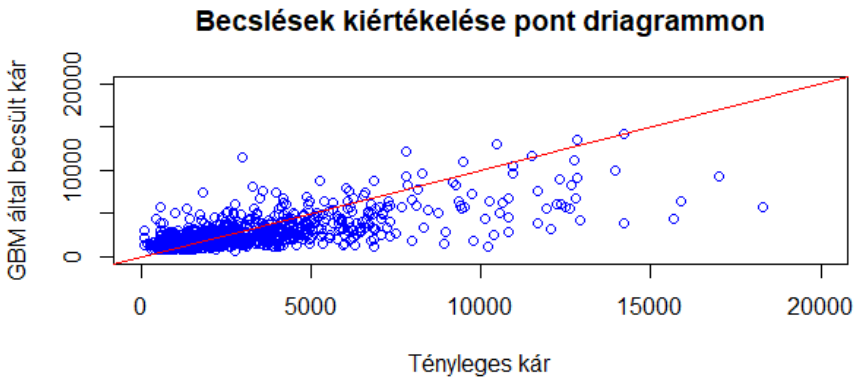
$$Nettó\,díj_i = 0,13\hat{y}_i$$

A szimuláció menetének a további ismertetése előtt kicsit elkalandoznék, és bemutatnám, hogy az előbb említett GBM általi becslések mennyire precízek a tényle-

[46] `https://www.statista.com/statistics/283306/total-motor-insurance-claims-frequency-in-the-uk?fbclid=IwAR3xhiKMo7KFN_9Mqt6rWj2S23wI16URdIt_yYohXXIv2Wd7P-xj36ivrHA` Utolsó letöltés: 2020.01.23. Ez az Egyesült Királyságra vonatkozó adat.

ges károk viszonylatában. Arra a részre koncentrálok, amit $\hat{y}_i$-nak neveztem. Megfelelőek a statisztikai eszközök, amelyek segítségével kiértékelhetjük a ML algoritmusok teljesítményét, azonban nehezen látható rajta a modellünk tényleges teljesítménye, itt most nem a korábban teljesítmény alatt értett fogalomra gondolok, hanem a gyakorlati vetületére. Azt tudjuk, hogy összehasonlítva melyik modell a legjobb, ahogy be is mutattuk. De összességében megéri őket használni? Felfedezik azokat a hangzatos adatok mögötti „mintázatokat”? Tekintsük meg a tényleges károk és a becsült károk pont diagramját a portfóliónkra nézve.

18. ábra. Becslések pont driagrammja




Ezek ismeretében mit mondhatunk? A pont diagrammon is látszik, hogy a nagy kárú, kockázatos ügyfeleket becsüli alul a modell. Erre egy megoldás lehet, hogy a kockázatos ügyfeleket szeparáljuk, és külön modellt építünk rájuk. Ehhez a magyarázóváltozók hatásainak mélyebb elemzésére lenne szükség, hogy mik azok a tulajdonságok, amik kockázatossá teszik az ügyfelet. Alkalmazhatunk rosszabb perdikciós erővel rendelkező, de sokatmondóbb modelleket. Ilyenek lehetnek például a GLM vagy egy döntési fa. Amikor kielemeztük már, hogy melyik változók teszik kockázatosság az ügyfelet, akkor őket már rögtön a „kockázatos” GBM-el lehetne árazni, ami talán egy pontosabb kockázati profil kialakításához vezetne.

Most hogy mélyebben láthattuk a GBM adta becslések viselkedését, amik a nettó díj alapját képezik, tovább haladok a szimuláció elemeivel. A nettó díjra meghatározok egy loadingot, amivel a ténylegesen befizetett díjak nagyságát kapjuk meg, tehát a bruttó díjat. Ezt a loading mértéket most még nem specifikálom, megtekinthető majd különböző loading nagyságokra, hogy a profitabilitásunk hogyan alakul és aszerint árazni. Természetesen piaci környezetben és a fogyasztói magatartást is

figyelembe véve ez nem ilyen egyszerű, mivel a nagyobb bruttó díj csökkenti a biztosítási termékünk iránti keresletet. Azonban ha jó terméket kínálunk és a versenytársak árazását is figyelembe vesszük, akkor lehet egy kis mozgásterünk az árazásban.

$$Bruttó\,díj_i = (1 + loading)Nettó\,díj_i$$

A bruttó díjra kétfajta költségtípust számolok fel a modellben, az egyik a jutalék a másikat csak simán költségeknek nevezem, ez a vállalat működési és egyéb költségeire vonatkozik. A termékre egy 15%-os jutalékot határozok meg és 10%-ot az egyéb költségekre. Ezeket a nettó díjra vetítjük. Tehát:

$$Jutalék_i = 15\%Nettó\,díj_i$$

$$Költség_i = 10\%Nettó\,díj_i$$

A kár darabszámra feltételezzük, hogy Poisson eloszlású valószínűségi változó 0,13 paraméterrel. A kárnagságokat is sztochasztikussá szeretném tenni, ezért bagging módszerrel k db GBM-et fogok felépíteni az eredeti tanuló adatbázison, amiket a portfóliónkon futtatva minden egyes szerződéshez k db becsült kárnagyság fog tartozni. A bagging módszer visszatevéses mintavétellel dolgozik, így különböző adatokon tanítjuk a k db GBM-et, ami biztosítja a véletlen faktort a kárnagyságokon. A k-t 50-nek választottam, mivel így már viszünk bele annyi véletlen hatást, amennyit a különböző mintavételek adhatnak. Kártáblának nevezek egy 1000 soros és k oszlopú mátrixot, minden egyes sora a portfolió egyik szerződését reprezentálja és minden oszlopa a k különböző GBM által becsült értéket. Az egy évben bekövetkező kár mennyiséget egyes szerződésekre annyi darab kárnagyság összege fogja kiadni, amennyi a poisson valószínűségi változó értéke, ezeket a kárnagyságokat minden szerződéshez tartozó károkból véletlenszerűen választjuk ki. Legyen $\eta \sim Pois(0, 13)$ a kárszám valószínűségi változója. Tehát $\eta$ db kárt választunk ki a kártáblából az $i$-edik szerződéshez a szimulációban. Így:

$$Kár_i = Kártábla_{i,\eta_1} + Kártábla_{i,\eta_2} + ... + Kártábla_{i,\eta_n}$$

,ahol n jelölje a $\eta$ aktuálisan felvett értékét az i-edik szerződésnél.

A legoptimálisabb megoldás az lenne, ha minden egyes szimulált évben ezeket a GBM-eket újra építeném és minden j évhez megkapnék egy kártáblát, azonban

ez az adatok nagysága és a programkód futási ideje miatt nehézkes, ezért minden szimulációs évben egy fix kártáblával dolgoztam.

Az egy szerződésre jutó profit értelemszerűen:

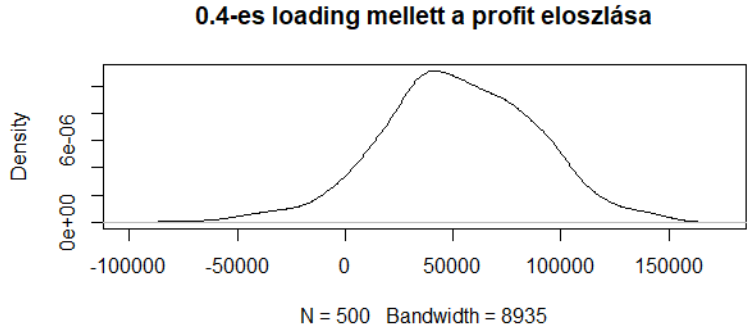
$$Profit_i = Bruttó\,díj_i - Jutalék_i - Költségek_i - Károk_i$$

Ezek után simán összegezve az egyes szerződésekre jutó profitokat, megkapjuk az éves teljesítményt.

A megvalósítás röviden: Egy excel file-ban létrehoztam a szimulációs tábla vázát, tehát az előbb nevesített oszlopokat elneveztem. Minden sor egy szerződést reprezentál, amit jelöltem id1, id2, ... stb. indexekkel. Ezt az excel file-t importáltam az R-be, majd for ciklusokkal feltöltöttem a megfelelő adatokkal a korábbi képleteknek megfelelően. Így keletkezik egy feltöltött állomány, amit a károkba vitt sztochasztikusság miatt tudunk többször szimulálni.

A szimuláció számát 500-nak választottam, ez már kellően nagy, hogy a különböző eseteket pontosan megfigyeljük. Első esetben tételezzünk fel egy 40%-os loadingot a nettó díjakra, így a nettó díj beárazásán felül egy 10%-ot hagyunk a káringadozásokra. Ezt a nálam most megjelenő 10%-ot gyakran nevezik biztonsági pótléknak is, ez pusztán a 40%-os loadingból a költség elvonások és az előírt profit után megmaradó rész. A kockázatokon kívül van a termékben még 10% költségrész, 15% jutalék és így 5% profit maradhat a vállalatnak, ha a károk a számításaink szerint történnek és a 40%-ból levonjuk az előbb említett költségeket. Ezek ismeretében tekintsük meg az éves eredményeinket. Így az összprofit eloszlása:

19. ábra. Egy évre a profitok eloszlása




Látható, hogy még ekkora loadinggal is sok esetben negatívan zárjuk az évet, de a várható értéke az éves profitnak pozitív lesz, 52 082 dollár. A portfóliónkra 468

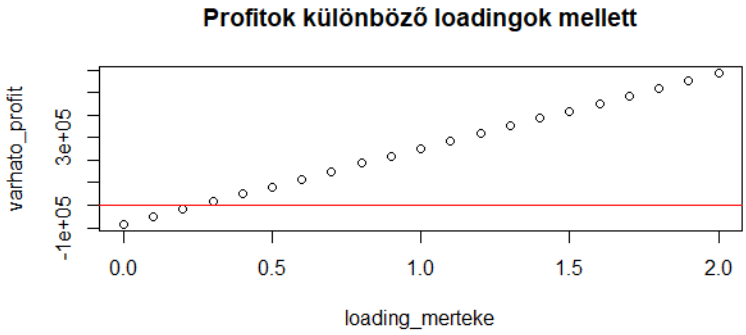
529 dollár bruttó díjbevételünk volt, és átlagosan 384 656 dollár kárkifizetés keletkezik. A működési költségekre 33 466 dollárt vontunk el a díjból és 50 199 dollár jutalékot fizettünk ki az alkuszoknak. Első ránézésre, ha csak a számokat analizáljuk elégedettek is lehetnénk a biztosítónk teljesítményével, de az ábra már óvatosságra inthet minket. A profitunknak pozitív a várható értéke, ami szerencsés. A varianciája azonban igen nagy, illetve ha a szimuláció eredményét nézzük, akkor 0,064 valószínűséggel negatív lesz a profitunk, vagyis veszteséges évet zárunk. Ez úgy gondolom még nem egy teljesen biztos érték és egy biztosító nem feltétlen engedheti megmagának hosszú távon. Természetesen ez így nem jelenthető ki általánosságban, sok függ a piaci helyzettől, illetve hogy a biztosító árul-e más termékeket. Ez a terméke lehet csak a szerződők bevonzása miatt, akik ha ezt a biztosítást nála kötik, akkor a jövedelmezőbb termékekből is az adott biztosító szolgáltatásait választhatja. Másik életszerű példa, hogy új a biztosító az adott piacon és növelni akarja a piaci részesedését, még a kezdeti veszteségek árán is. [47] Ha a piaci környezetet kivesszük a számításból, akkor a biztosítónk viszontbiztosíthatja erre a termékre kötött szerződéseit. A viszontbiztosítás megfelelő mód lehet az itt jelentkező problémára, vagyis a nagy variancia kiküszöbölésére az összkárokon. Másik megoldás lehet az önrész bevezetése. Ez persze a biztosítási díjat csökkenteni fogja, de ha sok kis kárunk van, akkor az összkárt jól kordában tudjuk vele tartani. Gépjármű biztosításoknál gyakran alkalmaznak önrészt, ez lehet fix összeg, a kár egy bizonyos százaléka [48] vagy a kettő maximuma. Én most nem ebben az irányban fogok tovább haladni, mivel a felépített modell keretein belül akarom kezelni a problémát, de a való életben ezek életszerű megoldások lehetnek egy termékfejlesztés kapcsán.

A következőkben bemutatom, hogy bizonyos loading értékek mellett mekkora lesz a várható profit, illetve a negatív profit valószínűsége. Ezekből az információkból meghatározzuk azt a biztonságos loadingot, ami mellett nyugodtan értékesítenénk a terméket. Természetesen figyelve a túlárazásra! A következőket látjuk:

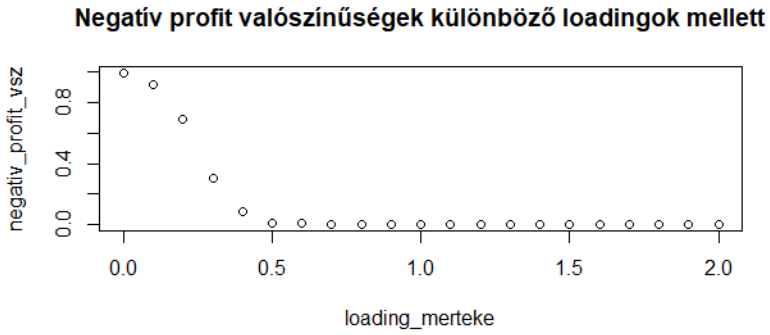
[47] Ezek az állítások akkor igazak, ha a nagyarányú veszteség a termék olcsósága miatt keletkezik. Ha a versenytársakhoz képest felül van árazva, akkor nyilvánvalóan ezek az érvek nem helytállóak.

[48] Ezt a kettőt nevezzük levonásos önrésznek

20. ábra. A profit és a loadingok viszonyulása




21. ábra. Negatív profit valószínűsége




A 20. ábra szemlélteti, hogy 0,3-es loadingtól már várhatóan nem lenne veszteségünk az adott állományon, azonban ha e fölé megyünk, akkor nagyobb magabiztossággal tudunk profitot realizálni. A 21. ábra sokatmondóbb, itt láthatjuk, hogy a negatív profit valószínűsége szinte 1-ről indul, majd ahogy növeljük a loadingokat ez szépen csökken. 0,4-es loadingnál még ott van egy 0,084-es negatív profit valószínűség, 0,5-nél 0,006, majd innentől szépen lecseng és szinte nulla lesz. 0,5-től már igen nagy magabiztosságunk lehet és a versenyképességet figyelembe véve, ez lehet az optimális érték, amennyit meghatároznék. Esetleg a költségek csökkentésével versenyképesebb díjat tudunk meghatározni, úgy hogy a profit és a kockázat az adott szinten marad. Csökkenthetjük a különböző működési költségeinket, ez most az automatizáció segítségével igen népszerű a biztosítók körében. Persze itt szükség van egy nagyobb kezdeti beruházásra, hogy a különböző IT rendszereket létrehozzák, ami segítségével hosszú távon a fix költségek csökkenthetők. Másrészt a jutalék, amiből tudunk csökkenteni. Ez veszélyes lehet, mivel az alkuszok hozzák az új szerződéseket. Emberek saját maguktól ritkán kötnek biztosítást. Illetve ha kevés

jutalékot ígérünk, akkor elvihetik az új szerződéseket valamelyik versenytársunkhoz, aki ráígér.

Természetesen ez egy kezdetleges példa volt egy termék árazására és egy biztosítási portfólió kezelésére. A célom csupán a korábban bemutatott ML eszközök aktuáriusi munkában betöltött egyik szerepét volt hivatott szemléltetni. Továbbiakban csak néhány ötletet szeretnék felvázolni, amilyen irányban ez a modellezés tovább vihető, illetve hogy ezt a kutatást milyen irányba lehetne még elvinni, azonban itt tartalmi és mennyiségi követelmények miatt már nem tudok vele foglalkozni.

Ahogy említettem az előbb is a kockázatosabb ügyfeleket megkell határozni, mert ott alul becsül a modell. Ezt mélyebb változó analitikával elérhető. Illetve a jobb predikció elérése érdekében gyakran használatos eszköz manapság a "stacking". Ez esetben a modellző több modellt épít egy adott adathalmazra, kiértékeli a tesztelő adatbázison, majd ezeket újra megbecsüli egy úgynevezett meta-modellel. Így többrétegű modelleket kapunk, aminek a gyakorlatban igen jó teljesítményei szoktak lenni.

A való életben természetesen nem csak éves díjfizetés van, ahogy feltételeztem, a havi gyakoribb. Azonban ha havi díjfizetéssel számolunk, akkor nem vehetjük biztosra, hogy mindenki fizeti folyamatosan a díjat. 12 hónappal számolva dinamikussá tehetjük a modellezésünket, lesznek be és kilépők. Belépők az új biztosítást kötők, kilépők a különböző okok miatti törölt szerződések. Másik egyszerűsítés volt, hogy mindenki egyszerre, év elején köti a biztosítást. Nyilvánvalóan ez sem igaz, a biztosítási évfordulók különbözőek szerződésenként, illetve különböznek a számviteli évfordulótól. Ennek következtében megfelelően tartalékolnunk kell.

# 7. Összefoglalás

Úgy érzem, hogy egy nagy témát fedtem le, amiről még számtalan oldalt lehetne írni különböző vonatkozásokban. Így a végére érve fontosnak tartom kiemelni, hogy mi is volt a célja a szakdolgozatnak, figyelembe véve a teljes anyag ismeretét.

Célja volt egy átfogó képet adni az AI sokszínűségéről a biztosításban. Szinte az ügyfél bevonzástól és a marketingtől kezdve, egészen a káresemény kiértékelésig mindenhol megjelenhet. Célja volt meghatározni a ML helyét az AI-on belül, hogy

pontosabban elhelyezhessük a fogalmainkat. Célja volt a legfontosabb ML algoritmusok bemutatása, értelmezése. Végső soron célja volt egy valós aktuáriusi példát hozni mindezekre, ahol a megvalósítást is nyomon tudtuk követni. Így talán még pontosabban meghatározhatóvá vált a gépi tanulás helye, szerepe és jelentősége a biztosítás matematikában.

Ezeken kívül fontos leszögezni, hogy nem volt célja a teljes AI áttekintése. Nagyon sok érdekes témára nem tudtam kitérni a tartalmi korlátok és valamelyest a téma meghatározottsága miatt. Én csak egyszerű adatstruktúrájú modelleket vettem számításba és mutattam be. Azonban a szöveg, kép és hangalapú modellek szintén kiválóan hasznosíthatók biztosítási területeken is. Nem volt célja, hogy az aktuáriusi alkalmazások közül mindet részletesen bemutassa, ez szintén az előbb említett korlátok miatt nem valósítható meg. Nem volt célja, hogy egy mély és teljes körű termékszimulációt adjon, a kapcsolatot tartottam lényegesnek a ML és az aktuáriusi alkalmazások között.

Összességében szerintem egy új és izgalmas terület az AI és ML, aminek az erejét csak most kezdjük érezni. Természetesen sok fejlődés áll még előtte, amíg megfelelően fognak működni az AI adta lehetőségek, de a biztosítók nem hagyhatják figyelmen kívül, mivel a hosszú távú versenyképességük jelentősen múlhat rajta. Ha nem is teljesen az AI, de az automatizáció problémája és nyomása már az itthoni biztosítókon is tetten érhető, már akkor is ha csak a nagy biztosítók vezetőinek nyilatkozatait követjük nyomon az elmúlt években. A gépi tanulási eszközök azonban veszélyeket is rejtenek magukban. Nem véletlen hívják sok modelljét „black box”-nak, vagyis nehezen elemezhető a különböző változók jelentősége és hatása az egész modellt tekintve. Ezért a gyakorlatban sokszor félnek tőle és nem preferálják, helyette nyúlhatnak egy jólbevált GLM-hez, ami teljesen megfelelő választás sok esetben. Fontos hangsúlyozni, hogy ha tudunk egyszerűsíteni a modellünkön, akkor tegyük meg, természetesen a teljesítmény kisméretű változásának a hatása mellett. A ML modellek nagyrésze olyan felépítésű, hogy akkor jön ki az ereje, ha nagyadathalmaz áll rendelkezésre és abból képes kiolvasni az adatok mögött rejtőző mintázatokat. Itt válik el a statisztika letisztult, feltételezésekre alapozó módszertana és a ML zűrzavarossága. De ahogy látjuk a biztosítási terület pont olyan, hogy hatalmas mennyiségű adat áll rendelkezésre, tehát megéri foglalkozni a témával.

# Hivatkozások


1. ASTIN Working Party (2018). Elérhető itt:https://www.actuaries.org/IAA/Documents/ASTIN/ASTIN_MLTMS%20Report_SJAMAL.pdf (Utolsó letöltés: 2019.11.18.)

2. Bhavsar, H., Ganatra, A. (2012): *A Comparative Study of Training Algorithms for Supervised Machine Learning*. International Journal of Soft Computing and Engineering, **Vol 2.**, 74 – 81. o.

3. Breiman, L. (1996): *Bagging Predictors*. Machine Learning, **24**, 123-140 . o.

4. Friedman, J.H. (2002): *Stochastic Gradient Boosting*. Computational Statistics & Data Analysis, **38**, 367-378 . o.

5. Hastie, T., Tibshirani, R., Friedman, J. H. (2009): *The elements of statistical learning (Second edition)*. Springer kiadó

6. Henckaerts, R., Cote, M.-P., Antonio, K. és Verbelen, R. (2019): Boosting insights in insurance tariff plans with tree-based machine learning methods. Elérhető itt: https://www.researchgate.net/publication/332631030_Boosting_insights_in_insurance_tariff_plans_with_tree-based_machine_learning (Utolsó letöltés: 2019.11.18.)

7. Panlilio, A., Canagaretna, B., Perkins, S., Preez, V. és Lim, Z. (2018): Practical Application of Machine Learning Within Actuarial Work. https://www.actuaries.org.uk/system/files/field/document/Practical%20Application%20of%20Machine%20Learning%20within%20Actuarial%20Work%20Final%20%282%29_feb_2018.pdf (Utolsó letöltés: 2019.11.18.)

8. Quinlan, J. R. (1986): *Induction of Decision Trees*. Machine Learning, **1**, 81-106. o.

9. Quinlan, J. R. (1996): *Improved Use of Continuous Attributes in C4.5*. Journal of Artificial Intelligence Reasearch **4** 77-90. o.

10. Richman, R. (2018): *AI in Actuarial Science*. Presented at the Actuarial Society of South Africa's 2018 Convention 24-25 October 2018, Cape Town International Convention Centre. Elérhető itt: https://www.actuarialsociety.

org.za/wp-content/uploads/2018/10/2018-Richman-FIN.pdf (Utolsó letöltés: 2019.11.19.)

11. Russel, S. J., Norvig, P. (2009): *Artificial Intelligence: A modern approach (Third edition)*. Pearson kiadó, New Jersey

12. Turing, A. M. (1950): *Computing Machinery and Intelligence*. Mind, **49**, 433-460. o.

13. Wahlström, N. Statistical Machine Learning előadásai. Megtalálható itt: http://www.it.uu.se/edu/course/homepage/sml, (Utolsó letöltés: 2019.09.28)

14. Wilmott, P. (2000): *Paul Wilmott Introduces Quantitative Finance (Second edition)*.John Wiley & Sons Ltd kiadó, Chichester

15. Yao, Y. (2018): *Evolution of insurance A Telematics Based Personal Auto Insurance Study* https://opencommons.uconn.edu/cgi/viewcontent.cgi?article=1563&context=srhonors_theses (Utolsó letöltés: 2019.09.07.)

16. Zou, H., Hastie, T. (2005): *Regularization and Variable Selection via the Elastic Net*. Journal of the Royal Statistical Society **Series B** 301-320.o.

17. https://www.statista.com/statistics/283306/total-motor-insurance-claims-freq?fbclid=IwAR3xhiKMo7KFN_9Mqt6rWj2S23wI16URdIt_yYohXXIv2Wd7P-xj36ivrHA (Utolsó letöltés: 2020.01.23.)