# Hagyományos statisztikai módszerek és neurális hálókon alapuló modellek összehasonlítása biztosítási adatokon


## Szendi Ágoston

Biztosítási és pénzügyi matematika MSc

Aktuárius szakirány


Szakdolgozat


Témavezető:

Prokaj Vilmos Dr.

Habilitált egyetemi docens

Valószínűségelméleti és Statisztika Tanszék




Eötvös Loránd Tudományegyetem
Természettudományi Kar
Budapest, 2024

# Tartalomjegyzék

# 1. fejezet

# Bevezetés

Köztudott, hogy táblázatos adatokra (Tabular Data) a mélyháló-modellek (Deep Neural Networks - DNNs) általában túlillesztenek. Emiatt kevésbé használhatóak, mint a hagyományosabb statisztikai eljárások, amilyen például az általánosított lineáris modell (Generalised Linear Model - GLM) vagy a döntési fa alapú módszerek, amilyen például az eXtreme Gradient Boosting, azaz az XGBoost. Talán egyetlen kivétel akad: a TabNet architektúra. Szakdolgozatom célja hagyományos statisztikai módszerek és neurális hálókon alapuló modellek összehasonlítása biztosítási adatokon. Választásom gépjárműkár adatokra esett, amelyeken a felsorolt módszerekkel próbálom a kárszámot illetve a kárnagyságot előrejelezni.

Az internetkapcsolattal és fedélzeti érzékelő készülékekkel felszerelt gépjárműveknek köszönhetően megbízhatóbb adatok birtokába juthatunk. Ezek egyre gyakoribbá válása valamint a mélyháló-modellek fejlődése előrelépéshez vezetett abban, hogy képesek legyünk a vezetési viselkedést precízen modellezni. Különösképpen a sofőrök kockázatának méréséhez szükséges információk gyűjtése - úgymint agresszív vezetési magatartás vagy a környezettel összefüggésbe hozható kockázat - járhat társadalmi szinten is előnyökkel. Ilyen sofőrkockázati profilok kialakításával és vizsgálatával ugyanis csökkenthető a balesetek száma és a károsanyag-kibocsátás. A sofőrnek küldött visszajelzések és a személyre szabott biztosítási termékek - azaz vezetési kompetencián alapuló, dinamikusan árazott biztosítások - ösztönzőleg hathatnak egy biztonságosabb vezetési magatartás kialakulására. Ezek a személyre szabott biztosítási termékek - mint amilyenek a pay-as-you-drive vagy pay-how-you-drive, azaz fizess, ahogy vezetsz szerződések is - kezdenek elterjedni a gépjármű-biztosítási ágazatban, amint a biztosítók a telematika és gépi tanulási (Machine Learning - ML) módszerek kombinációját használják a kockázatok árazásához. A hagyományos kockázatárazó modellek - mint például az általánosított lineáris modell (Generalised Linear Model - GLM)

- továbbra is uralják a biztosítási piacot, különösen a nem-életági üzletágakon belül. Annak érdekében azonban, hogy ténylegesen megragadhassuk a kapcsolatot a nem hagyományos adatok, például telematikus, műholdas vagy machine vision, azaz gépi látással nyert adatok minduntalanul táguló univerzuma és az aktuáriusi árazás célváltozói között - például kárgyakoriság és kárnagyság -, innovatív árazó mechanizmusokra van szükségünk. A mélytanulási (Deep Learning - DL) módszerek komplex, nemlineáris adathalmazok[1] modellezésére való kapacitása felülmúlja a legtöbb hagyományos árazómodell korlátait. Annak ellenére, hogy a mélyhálómodellek engedte számítási képességek növekednek, valamint nagyobb számban elérhetőek telematikai adatok, a mélytanulás lehetőségei kihasználatlanok a biztosítási kockázatárazás és a kárelőrejelzés terén. Szakdolgozatomban bemutatom egy alternatív mélytanulási architektúra, a TabNet (Tabular data learning neural Network) használatát kárszám és kárnagyság előrejelzésben is.

A mélyháló-modellek voltaképpen olyan gépi tanulási modellarchitektúrák, amelyek egyesítenek vagy összekapcsolnak különböző rétegeket ahhoz, hogy tanuljanak az adatokból. Ez a többrétegű felépítés lehetővé teszi, hogy minden egyes réteg megtanuljon egy egyedi tulajdonságot a vizsgált adatokról[8]. A mélytanulás fejlődése rendkívül pontos modellekhez vezetett, amelyek felülmúlták a gépi tanulási módszereket számos területen, úgymint önvezetés, természetes nyelvfeldolgozás (Natural Language Processing - NLP) és marketing[8]. Ezen mélyhálómodellek ráadásul minimális előtanításba vetett erőfeszítés árán képesek komplex adattípusokat is megtanulni[17]. Mindezek ellenére a biztosítói kockázatelőrejelző modellek között aligha találkozhatunk mélyháló-modellekkel, amelynek egyik alapvető oka ezek feketedoboz elméleti felépítése illetve szerkezete, vagyis ezek a modellek csak bemeneteik és kimeneteik alapján tekinthetőek meg anélkül, hogy a belső működésüket ismernénk. Ennek következtében ugyanis a kapott eredmények értelmezése és interpretálása nehézkessé válik, ami kihívást jelent a szabályozói kérdések megválaszolásakor. Bár a mélyháló-modellek kiemelkedő teljesítménnyel képesek komplex adattípusokat is megtanulni, táblázatos adatokon mégis szegényesen teljesítenek más, klasszikus maximum-likelihood módszeren alapuló modellekkel összevetve, mint például a Support Vector Machine, a logisztikus regresszió vagy a naiv Bayes-osztályozó[1].

**Definíció.** Adatszemcsésség: Az adatok szemcséssége (granularity) az adattudományokban használt szakkifejezés. Arra utal, hogy milyen részletességű illetve pontosságú az adat. Minél szemcsésebb, annál több egyedi információt tartalmaz,

[1] A nem lineáris adathalmaz elemei nem lineárisan, azaz egymást követően vannak elrendezve. Az elemek tehát nem kapcsolódnak az előttük és utánuk következő elemhez, így nem egy szintű az adat, és emiatt nem tudjuk egy futással végiglátogatni az összes elemet.

amely lehet egyedi feljegyzés vagy mérés.

A szemcsés telematikai adatokra alkalmazott korlátozások tovább bonyolítják a mélytanulás alkalmazását a biztosításban, mivel az újonnan bevezetett szabályozás megköveteli a telematikai adatok anonimizálását és feldolgozását, ami azt eredményezi, hogy a táblázatos adatok elterjedtek a biztosítás területén[21]. Másrészt komoly korlátot jelent a mélyháló-modellek biztosítási alkalmazásra való áttérésében az összetett, túlparaméterezett kialakításuk is[33].

A biztosításban kockázatárazáshoz használt modelleknek megmagyarázhatónak kell lenniük azért, hogy a pénzügyi hatóságok elfogadják azokat. Az általánosított lineáris modellek és az együttes módszerek (ensemble methods) hatékonynak bizonyultak némely sofőrkockázati profilok kockázatának meghatározásában és számszerűsítésében, ezért nem meglepő, hogy rendszeresen használják ezeket biztosítási kockázatárazásban[2]. Ezenkívül az általánosított lineáris modellek könnyen értelmezhetőek, ami alapvető tulajdonság a biztosítási díjkalkulációs modellek létrehozásához. Az általánosított lineáris modellek teljesítményét azonban korlátozhatja a rendkívül összetett vagy magasdimenziójú adat, mivel a lineáris prediktor nem tudja pontosan modellezni a magasdimenziójú magyarázóváltozók hatásait. Az olyan együttes módszerek, mint a véletlen erdő (Random Forest) vagy az XGBoost, képesek megtanulni bonyolult adatstruktúrákat, miközben magas szintű pontosságot biztosítanak. Ezeknek az együttes tanulási módszereknek azonban hátrányai is vannak, mivel a bonyolult folyamatok mind a modell finomhangolását, mind a modell értelmezését tekintve kevésbé vonzóvá tehetik őket a biztosítók számára[28]. Emiatt a feketedoboz modellek nem kedvezőek.

Az általánosított lineáris modell, az XGBoost és a mélytanulás korlátai különböző következményekkel járhatnak a biztosító hatékonyságára a tekintetben, hogy pontos gépjármű-biztosítási árazómodelleket tudjon felépíteni. Ráadásul az XGBoost és a mélytanulás abban is különbözik az általánosított lineáris modelltől, hogy nem képes teljes mértékben magyarázható és értelmezhető árazómodelleket nyújtani, amely tovább korlátozza ezek használatát. A TabNet, egy csúcstechnológiás mélyháló-modell architektúra a mélytanulási modellek korlátait igyekszik leküzdeni. Arik és Pfister 2021-ben mutatták be a TabNetet tanulmányukban[1], ahol a neurális háló és az XGBoost hasonló pontosságát említették. A TabNet egyetlen mélyháló-modellt, többlépcsős feldolgozást, szekvenciális figyelmet és gradienssereszkedést alkalmaz, így egy olyan architektúrát hoz létre, amely különbözik a hagyományos mélytanulástól, miközben magas pontosságot biztosít és lehetőséget nyújt a modell értelmezhetőségére. Ezen felépítésbeli adottságok

kombinációja teszi a TabNetet alkalmas mélyháló-modellé biztosítási kockázatárazáshoz, és kezeli más mélytanulási modellek korlátait.

Szakdolgozatom a TabNetnek egy alkalmazását nyújtja kárszám- és kárnagyság-előrejelzés révén. Ezen kívül a TabNetet összehasonlítom az általánosított lineáris modellel, az XGBoosttal valamint általam épített neurális hálókkal, hogy megvizsgáljam, melyik hatékonyabb előrejelzésben és értelmezhetőségben, illetve hogy melyik modell kívánja a legkevesebb erőfeszítést a tanításhoz. Az eredeti cikk szerint a korszerű mélytanulási architektúra kivételesen hatékonyan jelez előre biztosítási kockázatokat gépjárműadatok és ügyféladatok kombinációjával, és ez az új megközelítés lehetővé teszi a biztosítók számára, hogy az egyes vezetőknek megfelelőbb díjat szabjanak a TabNettel vezetési teljesítményük alapján, mint más hagyományos árazási modellekkel.

Szakdolgozatom elsősorban Arik és Pfister 2021-es[1] valamint Kevin McDonnel és munkatársainak 2023-as[23] munkáira épít, hiszen ugyanúgy egy mélyhálómodellt, a TabNetet használom biztosítási kockázatárazáshoz. Szakdolgozatomban összehasonlítom továbbá a TabNetet a gépi tanulási és a hagyományos biztosítási árazási modellekkel. A fent említett tanulmányok alapján a TabNet ebben a feladatban figyelemreméltó pontosságot kínál egyszerű modellértelmezhetőséggel és csökkentett adat-előfeldolgozási igénnyel. Ez a modell képes balesetek előrejelzésére is a telematikai adatok és a biztosítási igények kombinálásával.

Szakdolgozatom további része a következőképpen épül fel. A 2. fejezet bemutatja az adathalmazt. Az ezt követő fejezetben egyenként ismertetem az összehasonlított modellek működési elvét a megfelelő gépi tanulás, mélytanulási és matematikai háttér biztosításával. Ugyanitt kitérek a modellek implementációjára és az ehhez szükséges adatfeldolgozásra, majd bemutatom az így elért eredményeket. A 4. fejezetben megpróbálok magyarázattal szolgálni az eredményeimmel kapcsolatban. Végül pedig levonom a konzekvenciákat, illetve kitűzök lehetséges jövőbeli feladatokat a témában.

# 2. fejezet

# Adatok

A következő fejezetben részletesen bemutatom a modellezéshez használt adathalmazt.

## 2.1. Leírás

Az 'insuranceData' egy R kiegészítő csomag, amely különböző adathalmazokat tartalmaz biztosítási adatok elemzéséhez és modellezéséhez. Az általam választott 'dataCar' nevű is ezek közé tartozik. Ez az adathalmaz egy ausztrál biztosítótársaság 2004-es és 2005-ös, gépjárművekre szóló biztosítási szerződéseit tartalmazza. A 67 856 jegyzett kötvénytulajdonos közül 4 624, azaz 6,8% jelentett be legalább egy kárigényt. A szerződőkről ezen kívül további tizenegy ismérvet vagy jellemzőt rögzítettek. Modelleimet ezen az adathalmazon tanítottam és teszteltem.

## 2.2. Adatstruktúra

Az adathalmaz adatkeret (data frame) formátumú, és a következő tizenegy változót tartalmazza:

- veh_value: gépjármű értéke $10 000-ban
- exposure: az időszak hányad részében állt kockázatban (központi kitettség): 0 és 1 közötti szám
- clm: kár indikátora: 0 = nincs, 1 = van
- numclaims: kárszám
- claimcst0: kárnagyság: 0 ha nincs kár

- veh_body: vázszerkezet, kategorikus változó:
  BUS, CONVT, COUPE, HBACK, HDTOP, MCARA, MIBUS, PANVN, RDSTR, SEDAN, STNWG, TRUCK, UTE
- veh_age: gépjármű kora, ordinális változó: 1 (legfiatalabb), 2, 3, 4
- gender: nem, logikai változó: F, M
- area: környék/térség, kategorikus változó: A, B, C, D, E, F
- agecat: gépjárművezető kora, ordinális változó: 1 (legfiatalabb), 2, 3, 4, 5, 6
- ~~X_OBSTAT_: kategorikus változó: 01101 0 0 0~~

Az utolsó változó nem tartalmaz semmilyen információt sem a szerződőkről, hiszen mindegyik esetében ugyanazt az értéket vesz fel. Ezért, ahogy az áthúzás is jelzi, erre nincs szükség a modellezéshez.

Természetesen sem a kárszám, sem a kár nagysága nem ismert a szerződés megkötésekor, emiatt ezek az információk jövőbe látást feltételeznének. Ennek elkerülése végett az összes modell tanítása előtt ezeket a változókat is kiemeltem az adathalmazból.

## 2.3. Alapstatisztika

```
   veh_value          exposure          clm          numclaims         claimcst0          veh_body      veh_age    gender      area      agecat
 Min.   : 0.000   Min.   :0.002738   0:63232   Min.   :0.00000   Min.   :    0.0   SEDAN  :22233   1:12257   F:38603   A:16312   1: 5742
 1st Qu.: 1.010   1st Qu.:0.219028   1: 4624   1st Qu.:0.00000   1st Qu.:    0.0   HBACK  :18915   2:16587   M:29253   B:13341   2:12875
 Median : 1.500   Median :0.446270             Median :0.00000   Median :    0.0   STNWG  :16261   3:20064             C:20540   3:15767
 Mean   : 1.777   Mean   :0.468651             Mean   :0.07276   Mean   :  137.3   UTE    : 4586   4:18948             D: 8173   4:16189
 3rd Qu.: 2.150   3rd Qu.:0.709103             3rd Qu.:0.00000   3rd Qu.:    0.0   TRUCK  : 1750                       E: 5912   5:10736
 Max.   :34.560   Max.   :0.999316             Max.   :4.00000   Max.   :55922.1   HDTOP  : 1579                       F: 3578   6: 6547
                                                                                    (Other): 2532
```

2.3.1. ábra. Alapstatisztika

A 2.3.1 ábra az R összefoglaló, alap leíró statisztikája, amelyet a 'summary()' függvény meghívása után láthatunk. Ebből első pillantásra kivehető, hogy az adatok között nincs hiányzó érték. A minimumok, maximumok is életszerűek, ezért adattisztításra nincs szükség.

# 3. fejezet

# Modellezés

Az alábbi négy részben egyenként ismertetem az összehasonlított modellek működési elvét a megfelelő matematikai háttér biztosítása mellett. Majd leírom a modellek implementációját, illetve az ehhez szükséges könyvtárakat. Szó lesz arról is, hogy milyen előzetes adatfeldolgozási lépésekre volt szükségem, hiszen minden könyvtár eltérő módon kezelheti a változókat, illetve a különböző modellek is különböző inputot várnak. Végül egyenként beszámolok az elért eredményeimről.

## 3.1. Általánosított lineáris modell

Az általánosított lineáris modell (Generalized Linear Model - GLM) egy statisztikai modell, amely nevét onnan kapta, hogy kiterjeszti a lineáris regresszió képességeit. Eredete két brit statisztikushoz, John Nelderhöz és Robert Wedderburnhöz köthető, akik 1972-ben fejlesztették ki a módszertanát[25].

### 3.1.1. Általánosított lineáris modell a biztosításban

Az általánosított lineáris modellek széles körben elterjedtek a biztosítási kockázatbesorolásban[1] és árazásban, mivel képesek megragadni a változók hatásait, a belőlük számolt eredményt és a közöttük lévő paraméterezett kapcsolatot a vezetői kockázatbesorolásban[22]. A kárigények vagy balesetek előfordulása ritka, más szóval eloszlásuk a nulla értékre pozitív súlyt helyez, ezért az abszolút folytonos eloszlások nem megfelelően modellezik ezeket a ritkán előforduló baleseteket és kárigényeket. Az általánosított lineáris modell bármilyen exponenciális eloszláscsaládbeli eloszlást feltételezhet az eredményváltozók eloszlására vonatkozó-

[1] A biztosítási kockázatbesorolás az a folyamat, amely során a biztosítótársaságok felmérik és kategorizálják a biztosítottak vagy a biztosítandó tárgyak kockázati szintjét. Ennek célja, hogy meghatározzák a biztosítási díjakat és a biztosítási fedezet feltételeit.

an, lehetővé téve olyan modellek becslését, mint például a Poisson-regresszió[18], amelyek pontosabban írják le a kockázati tényezők és a balesetek közötti kapcsolatokat. Ezen felül az általánosított lineáris modellek nagyon jól értelmezhetőek annak köszönhetően, hogy az általánosított formát, azaz az eloszlást és a kapocsfüggvényt magunk választhatjuk meg. A modell súlyai és változói közötti kölcsönhatások egy adathalmazon szintén könnyen értelmezhetők ezeknek az általánosított kifejezéseknek köszönhetően[24].

A klasszikus szerződői adatok és a telematikai adatok kombinálása növelte a biztosítók képességét a biztosítási díjak pontos meghatározására. Verbelen, Antonio és Claeskens 2018-ban bemutatták ennek a megközelítésnek a hatékonyságát, kombinálva ilyen hagyományos tényezőket telematikai adatokkal a kárigények előrejelzéséhez.[32] A szerzők az általánosított lineáris modell egy változatát, az általánosított additív modellt (Generalized Additive Model - GAM) használták a kockázat hatékony modellezésére a telematikai adatokon keresztül. Ez a modell a kitettséget vagy a futásteljesítményt emelte ki mint a kötvénytulajdonos kockázatához elsődlegesen hozzájáruló tényezőt. Az általánosított lineáris modell egy másik változata, a Poisson-regresszió szintén pontos kockázati profil előrejelzéseket biztosított Ma és munkatársai számára 2018-ban[18]. Amikor a hagyományos és a telematikai adatokat kombinálták, a modelljük azt mutatta, hogy a futásteljesítmény, a csúcsidőben történő közlekedés és a vezetési viselkedések, például a hirtelen fékezés erősen összefüggnek a vezetői balesetekkel. Ezenkívül a gyorshajtást és a relatív sebességet azonosították kockázati tényezőkként. Tehát láthatjuk, hogy az általánosított lineáris modellek lehetővé teszik a biztosítók számára, hogy árazómodelleket készítsenek, miközben értelmezhető eredményeket nyújtanak, feltárva a kötvénytulajdonos kockázatához jelentősen hozzájáruló jellemzőket.

### 3.1.2. Az általánosításra való igény

Az $f(x_1; x_2; ...; x_p) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$ lineáris függvény temérdek szempontból előnyös lehet a modellezéshez. Mindenekelőtt $\beta$ paraméterei könnyen értelmezhetőek, ráadásul geometriai tartalma is egyszerűen látható, szemléletes jelentéssel bírhat. Továbbá parciális deriváltjai mind konstansok, és a lineáris függvények más, bonyolult függvények lokális közelítésére is alkalmasak.

Éppen ezért szerethető a Carl Friedrich Gausstól (1777-1855) származó lineáris regresszió is, ha azt feltételezzük, hogy a megfigyelt $y$ értékeink egy lineáris függvény körül véletlenszerűen szóródnak:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \varepsilon$$

Itt az $\varepsilon$ hibatagokról azt feltételezzük, hogy függetlenek és nulla várható értékű normális eloszlásúak, vagyis:

$$\varepsilon \sim \mathcal{N}(0; \sigma^2).$$

Emiatt az $Y$ kimenetelek is feltételesen függetlenek és normális eloszlásúak az $\mathbf{X}$ mintától függő várható értékkel, azaz

$$Y|_{\mathbf{X}} \sim \mathcal{N}(\mu(\mathbf{X}); \sigma^2).$$

Ennek következtében elméletben bármilyen $y \in \mathbb{R}$ kimenetel lehetséges.

Ebben a formában az eredményváltozó várható értékét megkaphatjuk a magyarázóváltozók lineáris függvényeként:

$$\mathbb{E}(Y|\mathbf{X}) = \mu(\mathbf{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p.$$

Ám sok elméleti előnye ellenére a lineáris regresszió gyakran nem használható a gyakorlatban. Például a biztosításmatematika esetében is, ha a kimenet bináris változó, mint a csőd: volt-e vagy sem, csak 0 vagy 1 értéket vehet fel; a kárkifizetésre sem célszerű illeszteni, hiszen ez ferde eloszlású, és csak nemnegatív értékeket vehet fel; de a gyermekek vagy elhunytak számának leírására sem alkalmas, hiszen ez is csak nemnegatív lehet, és egész értékű.

A lineáris regresszió előbbiekben felsorolt korlátai miatt merül fel az általánosításra való igény a biztosításmatematikában, de az általánosított lineáris modell sok más tudományterületen is használatos, mint például az orvostudomány, a biológia és a társadalomtudományok.

### 3.1.3. Az általánosítás

Az $y$ kimenetelekről továbbra is azt feltételezzük, hogy függetlenek, eloszlásuk azonban nem csupán normális, hanem bármely exponenciális eloszláscsaládbeli lehet.

**Definíció.** Exponenciális eloszláscsalád: a $\mathcal{P}$ valószínűségi mértékcsaládot exponenciális eloszláscsaládnak mondjuk, ha $\mathcal{X}$ alaphalmazból származó minták és $\Theta \subset \mathbb{R}^d$, $d \in \mathbb{N}$ paramétertér esetén $\mathcal{P} = \{\mathbb{P}_\vartheta, \vartheta \in \Theta\}$ dominált valamilyen mértékkel, és $f_\vartheta(\mathbf{x}) = \exp\{\langle \vartheta; T(\mathbf{x})\rangle - b(\vartheta)\}$, ahol $b : \Theta \to \mathbb{R}$ és $T : \mathcal{X} \to \mathbb{R}^d$ mérhető függvények.

Az exponenciális eloszláscsalád néhány nevezetes tagja:

- Folytonos:
  - Normális
  - Gamma
  - Lognormális
  - Pareto
  - Inverz Gauss
- Diszkrét:
  - Bernoulli
  - Binomiális
  - Poisson
  - Negatív binomiális

Ahogy eddig, úgy most is feltesszük, hogy az eredményváltozó várható értéke a prediktorok függvénye:

$$\mathbb{E}(Y|\mathbf{X}) = \mu(\mathbf{X}).$$

Csakhogy a lineáris regressziónál általánosabban most nem magát ezt a várható értéket, hanem ennek egy transzformáltját magyarázza a magyarázóváltozók lineáris függvénye:

$$g(\mu(\mathbf{X})) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p,$$

ahol $g$ szigorúan monoton növő és differenciálható. Az ezt a kapcsolatot leíró $g$ függvényt kapocsfüggvénynek (link function) nevezi a szakirodalom.

### 3.1.4. Példák

#### Normális eloszlás

Természetesen az általánosított lineáris modellek közé tartozik maga a modellcsalád ősatyja, a lineáris regresszió is. Ez könnyedén ellenőrizhető.
A kimenet feltételes eloszlása normális, amely az exponenciális családba tartozik:

$$Y|_{\mathbf{X}} \sim \mathcal{N}(\mu(\mathbf{X}); \sigma^2).$$

Várható értéke:

$$\mathbb{E}(Y|\mathbf{X}) = \mu(\mathbf{X}).$$

Ennek transzformáltja a prediktorok lineáris függvénye:

$$\mu(\mathbf{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p.$$

A kapocsfüggvény tehát az identitás:

$$g(u) = u.$$

**Bináris klasszifikáció**

Ahogy már említettem, a bináris klasszifikáció roppant fontos a gyakorlatban, így a biztosításmatematikában is, de a lineáris regressziós modell nem alkalmas erre. Például, ha a csődöt kívánjuk előrejelezni, ahol a kimenet csak 0 vagy 1 lehet attól függően, hogy csődbe megy-e a vállalat, akkor hasznunkra vállhat az általánosított lineáris modell. Erre mutatok néhány lehetőséget.

**Logit modell** Az előbbi példánál maradva vezessük be a csőd valószínűségére a $p(\mathbf{X}) = \mathbb{P}(Y = 1|\mathbf{X})$ jelölést. Ekkor a kimenet feltételes eloszlása Bernoulli, amely az exponenciális családba tartozik:

$$Y|_{\mathbf{X}} \sim Bernoulli(p(\mathbf{X})).$$

Várható értéke:

$$\mathbb{E}(Y|\mathbf{X}) = p(\mathbf{X}).$$

Ennek transzformáltja a prediktorok lineáris függvénye:

$$\ln \frac{p(\mathbf{X})}{1 - p(\mathbf{X})} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p.$$

A kapocsfüggvény tehát a logit függvény:

$$g(u) = \ln \frac{u}{1 - u}.$$

Más kapocsfüggvénnyel más bináris osztályozóhoz juthatunk.

**Probit modell** A kimenet eloszlása itt is Bernoulli. Az előbbi gondolatmenet most is végigvezethető azzal a különbséggel, hogy a kapocsfüggvényt másnak választjuk:

$$g(u) = \Phi^{-1}(u),$$

ahol $\Phi$ a standard normális eloszlás eloszlásfüggvénye. Ennek okán továbbra is $g : (0; 1) \to \mathbb{R}$.

A probit modell az 1930-as, a logit az 1970-es évek terméke. Mára a logit jóval népszerűbbé vált.

**Cloglog** Ez a modell akkor használatos, ha az események valószínűsége szélsőséges: nagyon kicsi vagy nagyon nagy. Az ehhez tartozó kapocsfüggvény:

$$g(u) = \ln(-\ln(1-u)).$$

#### Darabszámok modellezése

A darabszámok modellezése szintén meghaladja a lineáris regresszió határait, hiszen gondoljunk akár az egy háztartásban élő eltartott gyermekek számára vagy akár a biztosításban az egy évben egy vezető által okozott gépjármű-káresemények számára, ezek eloszlása nyilvánvalóan nem normális eloszlású. Azonban az eddigiekben felsorolt példák sem alkalmasak ezeknek a célváltozóknak a leírására, mivel Bernoulli-eloszlás sem feltételezhető. Gyakori választás szokott lenni a negatív binomiális eloszlás vagy a Poisson-eloszlás.

Ez utóbbi neve speciálisan Poisson-regresszió, amennyiben a kapocsfüggvény a természetes alapú logaritmus. Minthogy ez szerves része a szakdolgozatomnak, ezért a következő részben részletesen bemutatom.

### 3.1.5. Kárgyakoriság becslése Poisson-kárszámmodellben

Az általam választott adathalmaz gépjárművekre szóló biztosítási szerződéseket tartalmaz. Ebben az esetben gyakran feltételezzük, hogy a biztosítottak kárainak száma Poisson-eloszlást követ, valamint hogy a károk nagysága független a károk számától. Ennek következtében a biztosított kockázatát az átlagos kárnagyság és a várható kárszám szorzata határozza meg. Ezen kívül ismerjük a biztosítottak életkorát, nemét, lakóhelyét, a jármű értékét stb. Ezek a jellemzők vagy ismérvek mind hatással lehetnek a kárgyakoriságra, és célunk ezen hatás feltárása a megfigyelt kárgyakoriságok alapján.

A továbbiakban az $i$-edik biztosított megfigyelt kárszámát $Y_i$-vel, jellemzőit (kor, nem stb.) pedig $X_{i,j}$-vel jelöljük. Így $Y$ egy vektor és $X$ egy mátrix, ahol

$X_i = \{X_{i,j} : j\}$ az $X$ mátrix $i$-edik sorvektora, amely az $i$-edik biztosított jellemzőit gyűjti össze. $Y$ lesz tehát a válaszváltozónk (response variable), míg $X$ a magyarázó változónk (explanatory variable).

A jellemzők között szerepel a kitettség is. Ezt külön figyelembe kell vennünk a modellben. Ha az $i$-edik szerződő $t_i$ időn keresztül állt kockázatban, és az egyes időszakok (pl.: napok) kárszámait egymástól függetlennek feltételezzük, akkor a következő modellhez jutunk:

$$Y_i \sim Poisson(t_i \lambda_i),$$

ahol $\lambda_i$ az $i$-edik szerződő magyarázó változóinak, vagyis az $X_i$ vektornak rögzített függvénye. Ebben a modellben tehát az $i$-edik szerződő várható kárszámát a kockázatban töltött idő és a magyarázó változók függvényének szorzata adja.

Azt a függvényt, amely $\lambda_i$-t állítja elő, egy paraméteres családon belül keressük, ahol a paramétereket $\beta$-val jelöljük, tehát $\lambda_i$ az $X_i$ és $\beta$ adott függvénye. Célunk $\beta$ paraméter maximum likelihood becslésének (Maximum Likelihood Estimation - MLE) kiszámítása. A számítást kényelmesebbé teszi, ha az $X_i$-től és $\beta$-tól való függés az

$$\eta_i = \beta_0 + \sum_j (X_{i,j}\beta_j)$$

lineáris függvényen, más néven lineáris prediktoron keresztül valósul meg, azaz $\lambda_i = \mu(\eta_i)$ és $\eta_i = \beta_0 + X_i\beta$.

$\eta_i$ tehát a lineáris prediktor, míg a $\mu$ függvény inverze tölti be a kapocsfüggvény szerepét, így $g^{-1} = \mu$ az inverz kapocsfüggvény.

A Poisson eloszlású példánál, amilyen a miénk is, $\mu : \mathbb{R} \to (0, \infty)$. Itt $\mu(x) = e^x$ egy népszerű választás. Ekkor a kapocsfüggvény a $g = log$, azaz a logaritmusfüggvény, ezért ezt szokták röviden log-kapocsnak is hívni. $X_{i,0} = 1$ választással $\lambda_i = \mu(\eta_i) = \mu(X_i\beta)$. Hamar észrevehetjük, hogy a $t_i$ kockázatban töltött idő is beolvasztható a lineáris prediktorba $\log(t_i)$ hozzáadásával. Ez egy olyan tag a lineáris prediktorban, aminek az együtthatója a konstans 1, és ezt nem is szeretnénk becsülni. Az ilyen tagokat offsetnek nevezik.

A magyarázóváltozóink között vannak folytonosak, pl.: a gépjármű értéke; vagy kategorikus változók, mint a nem vagy a környék típusa stb. A kategorikus változókat indikátorokkal kell reprezentálnunk. Mivel például hat térségtípust különböztetünk meg (A, B, C, D, E, F), ezért lesznek $X_{i,A}$, $X_{i,B}$, $X_{i,C}$, $X_{i,D}$, $X_{i,E}$, $X_{i,F}$ magyarázóváltozók, amelyek közül pontosan egy darab egyes a többi nulla. Az ezeknek megfelelő paraméterek $\beta_A$, $\beta_B$, $\beta_C$, $\beta_D$, $\beta_E$, $\beta_F$. Az $X_{i,j}$ lineáris

összefüggése miatt

$$\sum_{j\in\{A,B,C,D,E,F\}} X_{i,j}\beta_j = \beta' + \sum_{j\in\{A,B,C,D,E,F\}} X_{i,j}(\beta_j - \beta'),$$

és így a kategorikus változók szintjei közül az egyik elhagyható anélkül, hogy a modell megváltozna. Az egyik szintet érdemes is elhagyni a túlparaméterezettség elkerülése érdekében; célszerű a leggyakoribbat.

Ezek után nekikezdhetünk a maximum likelihood becslésnek. A likelihood-függvény a következőképpen írható fel:

$$L(\beta) = \prod_i \frac{\lambda_i^{Y_i} \cdot e^{-\lambda_i}}{Y_i!} = \prod_i \frac{\mu(\eta_i)^{Y_i} \cdot e^{-\mu(\eta_i)}}{Y_i!} = \prod_i \frac{e^{\eta_i Y_i} \cdot e^{-e^{\eta_i}}}{Y_i!}.$$

Vegyük mindkét oldal természetes alapú logaritmusát, hogy eljussunk a log-likelhoodfüggvényhez:

$$\ell(\beta) = \sum_i (\eta_i Y_i - e^{\eta_i} - \log(Y_i)).$$

A paramétertől független tagok elhagyása után marad

$$\sum_i (\eta_i Y_i - e^{\eta_i}), \text{ ahol } \eta_i = \log(t_i) + X_i\beta$$

továbbra is. Feladatunk ennek a kifejezésnek a maximalizálása $\beta$-ban, amely, feltéve hogy egyetlen szélsőérték van, a derivált nullhelyének a meghatározásával ekvivalens. Tehát a megoldandó egyenlet a likelihood-egyenlet, amely az alábbi formát ölti

$$\partial_\beta \sum_i (\eta_i Y_i - e^{\eta_i}) = \sum_i (Y_i - e^{\eta_i}) X_i^T = X^T(Y - \mu) = 0, \text{ ahol } \mu_i = \mu(\eta_i) = t_i e^{X_i\beta}.$$

A log-likelihood maximumhelyére legtöbbször nem adható zárt képlet. Numerikus eljárással azonban maximalizálhatunk. Az R például a Newton-módszert használja az optimum megkereséséhez, ez alapján iterál, amíg az eljárás nem konvergál. A Newton-módszer ugyanis használható egy függvény minimumának vagy maximumának meghatározására. A derivált a lokális minimum- vagy maximumhelyen nulla, így a lokális minimumok és maximumok a Newton-módszernek a deriváltra történő alkalmazásával kereshetőek. Az iteráció tehát a következő lesz a mi esetünkben:

$$\beta_{n+1} = \beta_n - \mathbf{H}^{-1}(\beta_n)\nabla\ell(\beta_n),$$

ahol **H** a log-likelikood függvény Hesse-mátrixa.

### 3.1.6. Kárnagyság becslése általánosított lineáris modellel

A kárnagyságok modellezése általánosított lineáris modellekkel különösen kihívást jelenthet az aktuáriusok számára, mivel a káradatok sajátos jellemzőkkel bírnak. Az egyik legnagyobb nehézséget az okozza, hogy a kárnagyságok között gyakran sok a nulla érték, ami azt jelenti, hogy számos esetben egyáltalán nem keletkezik kár. Ez a zero-inflation jelenség jelentősen torzíthatja a hagyományos modelleket, amelyek általában nem képesek hatékonyan kezelni a nagyszámú nullás megfigyelést. Ráadásul, amikor kár történik, a kárnagyságok nemnegatívak és folytonos eloszlásúak, ami további bonyodalmakat okoz a modellalkotás során. A folytonos eloszlások kezelésére első látásra azt feltételezhetnénk, hogy az adatok valamilyen ismert eloszlásból, például normál, gamma vagy inverz-Gauss eloszlásból származnak, amelyek közül egyik sem kezeli jól a 0 értékeket. Ennek eredményeként a modell becslései és predikciói pontatlanok lehetnek, ha nem vesszük figyelembe a zero-inflation hatását. Az aktuáriusok gyakran kényszerülnek alternatív módszerek, például zero-inflated modellek vagy kétlépcsős modellezési technikák alkalmazására, amelyek először a kár bekövetkezésének valószínűségét, majd a kár nagyságát modellezik. Ezek az összetettebb megközelítések jobban figyelembe veszik a káradatok természetes sajátosságait, és pontosabb, megbízhatóbb eredményeket nyújthatnak a kárnagyságok előrejelzésében.

Én azonban egy alternatív megközelítést szeretnék mutatni, amellyel nincs szükség a komplex modellalkotásra. Ez a megoldás egyszerűen implementálható a hagyományos általánosított modell eszköztárával. Egy olyan eloszlás alkalmazása oldja fel ezt a problémát, amely az exponenciális eloszláscsalád tagja, a pozitív félegyenesen folytonos, ám a nullára pozitív súlyt helyez. Éppen olyan, amilyenre szükségünk van.

#### Tweedie-eloszlás

A Tweedie-eloszlás egy olyan eloszláscsalád, amely az általánosított lineáris modellek keretében használható, különösen hasznos azokban az esetekben, amikor az adatok nulla értékeket és folytonos pozitív értékeket is tartalmaznak. Ez az eloszláscsalád a biztosítási matematikában és a kockázatelemzésben különösen fontos szerepet játszik, mivel képes modellezni a zero-inflated folytonos adatokat, amelyek gyakran előfordulnak például káradatok esetén.

**Matematikai háttere** A Tweedie-eloszlások az exponenciális család tagjai, és sűrűségfüggvényük az alábbi formában írható fel[6]:

$$f(y;\mu,\phi,p) = a(y,\phi)\exp\left(\frac{1}{\phi}\left(y\theta - \kappa(\theta)\right)\right),$$

ahol:

- $y$ a megfigyelt érték,
- $\mu$ a várható érték,
- $\phi$ a diszperziós paraméter,
- $\theta$ a természetes paraméter,
- $\kappa(\theta)$ a kumuláns függvény,
- $p$ pedig az eloszlás hatvány-paramétere.

A Tweedie-eloszlás különböző speciális eloszlásokat foglal magában különböző $p$ értékek esetén:

- $p = 0$: Normális eloszlás,
- $p = 1$: Poisson-eloszlás,
- $1 < p < 2$: keverék Poisson-eloszlás - nemnegatív, pozitív súllyal 0-ban,
- $p = 2$: Gamma-eloszlás,
- $p = 3$: Inverz-Gauss eloszlás,
- $p > 2$: Stabil eloszlás a pozitív félegyenesen.

**Definíció.** Stabil eloszlás: A valószínűségelméletben egy eloszlást akkor mondunk stabilnak, ha két független valószínűségi változó lineáris kombinációja ezzel az eloszlással azonos eloszlású a lokációs és a skálaparaméterektől eltekintve.

**Alkalmazása** A Tweedie-eloszlás különösen hasznos a biztosítási matematikában, mert képes egyszerre modellezni a gyakori nullás értékeket és a pozitív, folytonos értékeket. Például, ha egy biztosítási szerződés adatait vizsgáljuk, a legtöbb hónapban nem történik kár (0 érték), de ha kár történik, annak nagysága folytonos és pozitív értéket vesz fel.

**Példa** Tegyük fel, hogy egy biztosító társaság a káradatokat szeretné modellezni, ahol a kárnagyságok gyakran nulla értékűek, de ha kár történik, akkor annak nagysága pozitív. Ilyen esetekben a Tweedie-eloszlás ideális választás, mert az $1 < p < 2$ tartományban képes kezelni mind a nullákat, mind a folytonos pozitív értékeket. Az aktuáriusok ilyen modelleket alkalmazhatnak a kockázati profilok pontosabb meghatározása érdekében, ezáltal jobban megérthetik és kezelhetik a kockázatokat.

**Implementáció** A Tweedie-eloszlást különböző statisztikai szoftverekben, például R-ben vagy Pythonban is implementálhatjuk. Az R 'statmod' csomagja és a Python 'statsmodels' könyvtára támogatja a Tweedie-eloszlás alkalmazását, amely lehetővé teszi a felhasználók számára a paraméterek becslését és a modellezést a rendelkezésre álló adatok alapján.

Összefoglalva tehát a Tweedie-eloszlás egy rugalmas és hatékony eszköz a zero-inflated és folytonos pozitív adatok modellezésére, különösen hasznos a biztosítási adatok és más hasonló típusú adatok elemzésében.

### 3.1.7. Modellimplementáció

A modellezést R-ben végeztem. Az általánosított lineáris modellek építéséhez a 'glmnet' könyvtárra volt szükségem. A Tweedie-eloszlás nincs az eredetileg elérhető eloszlások között, ezért használatához a 'statmod' könyvtárat is be kellett töltenem.

### 3.1.8. Adatelőkészítés

R adathalmazról lévén szó, az adatbeolvasás könnyen elvégezhető volt az 'insuranceData' könyvtár betöltése után, amely kifejezetten biztosítási adatok elemzési céljából készült. Ebből a 'data' függvénnyel kinyerhető az általam választott 'dataCar' adathalmaz is. Ezután az 'X_OBSTAT_' változót azonnal elhagytam, hiszen, ahogy az Adatok című fejezetben megállapítottam, ezekre nincs szükség a modellezéshez.

Bár a Kárgyakoriság becslése Poisson-kárszámmodellben című részben részletesen levezettem, hogyan lehet a kategorikus változókat is bevenni egy általánosított lineáris modellbe, valójában erre nem volt szükségem. Az R sokszor magától is felismeri, melyek az ilyen típusú változók, és az egyik szint elhagyása nélkül is könnyedén tudja ezeket kezelni. Esetemben ez nem így volt az életkorkategóriákkal és a kárbekövetkezést jelző bináris változóval, de még ezzel együtt sem kellett

mást tennem, mint az 'as.factor()' függvénnyel kikényszerítenem, hogy kategorikus változóként tekintsen ezekre.

Ha ezek után meghívjuk a 'str()' függvényt az adathalmazra, akkor ellenőrizhető, hogy most már minden változót megfelelően kezel az R.

### 3.1.9. Eredmények

#### Kárszámbecslés

Az általánosított lineáris modellek esetében két gyakran használt mérőszám a jól ismert átlagos négyzetes eltérés négyzetgyöke (Root Mean Square Error - RMSE) és az átlagos abszolút eltérés (Mean Absolute Error - MAE). A megbízhatóbb eredmények érdekében 80%-os tanítóadat rész mellett tízszeres keresztvalidációt (cross-validation) végeztem, ahol ezekre a mértékekre rendre a 0,2770 és 0,1356 értékeket kaptam. Viszonyítási alapnak először felállítottam egy úgynevezett nullmodellt, amely minden szerződőhöz nulla kárszámot rendel. Ezzel ugyanezek az értékek hasonlóan alakultak: az átlagos négyzetes eltérés gyökére 0,2770-et, az átlagos abszolút eltérésre 0,1356-ot kaptam. Tehát a nullmodell becslései hasonlóan alakultak a matematikai statisztika alapjain nyugvó általánosított lineáris modelléhez, sőt az átlagos abszolút eltérés még javult is. Ez igencsak visszavetette lelkesedésemet, ám kicsivel később eltöprengtem, mi lehet ennek az oka.

Az eredmények értelmezésekor arra jutottam, hogy a két gyakran, az R által is alapbeállításként szolgáló mérőszám nem azt méri, amire egy aktuárius kíváncsi. Egy biztosítói állomány vizsgálatakor ugyanis nem egyéni szinten szeretnénk eltalálni, hogy hány kárt okoz valaki. Amint láthattuk, erre nem is vagyunk képesek. Ám a nagy számok törvénye alapján az állomány összkárszámát azért jó volna eltalálni. Ehhez új mérőszámként bevezettem a predikált és valós összkárszám arányát. Ezzel a mértékkel szeretnénk egyhez minél közelebbi értéket kapni, hiszen ekkor vagyunk közel a pontos összkárszámhoz állományszinten. Mindjárt ellenőrizhetjük, hogy a nullmodellt ez az új mérőszám biztosan elutasítja, hiszen nullát kapunk vele eredményül.

Amikor ugyanazt a Poisson-regressziót illesztettem az egész adatbázisra, akkor az új mértékkel azt tapasztaltam, hogy pontosan eltalálja az állomány összkárszámát! Elsősorban a gépjárművezető életkora és a vázszerkezet változó bizonyult statisztikailag szignifikánsnak. Az előbbivel azonos keresztvalidációval pedig az átlagos találati arány 0,9994, azaz rettentően pontos. Ráadásul a találati arány szórása csekély, csupán 0,0245. Kezdeti csüggedtségem után üdítőleg hatottak ezek az eredmények.

#### Kárnagyságbecslés

Több eloszlás és kapocsfüggvény kombinációjával próbálkoztam általánosított lneáris modellt illeszteni az adatokra. Kezdeti modelljeim gyenge teljeítményt nyújtottak, sőt gamma és inverz-Gauss eloszlásokkal nem futott le semmilyen kapocsfüggvény mellett sem az illesztés. Ennek oka, hogy a 67 856 jegyzett kötvénytulajdonos közül csupán 4 624, azaz 6,8% jelentett be legalább egy kárigényt, vagyis temérdek 0 érték szerepel a célváltozó valós értékei között, ám a felsorolt eloszlások folytonosak. Ezt végiggondolva nem is csoda, hogy nem sikerült folytonos eloszlást illeszteni egy olyan eloszlású változóra, amely a nulla értéket pozitív valószínűséggel veszi fel.

Éppen ezt a problémát oldja meg a Tweedie-eloszlás, amelyről egy korábbi elméleti összefoglalóban részletesen írok. Ott olvashatjuk, hogy abban az esetben, hogyha az eloszlás $p$ hatványparamétere 1, akkor még diszkrét az eloszlás, és 1-nél nagyobb, de 2-nél kisebb értékek esetén nemnegatív, pozitív súllyal 0-ban, amilyet feladatom is megkíván. Mivel az adathalmazban csekély számú szerződésnél jelentettek be kárt, ezért igyekeztem 1-hez, azaz a diszkrét eloszláshoz minél közelebbi értéket beállítani hatványparaméternek, amelyet R-ben a 'var.power' paraméter jelöl. Végül az 1,01 beállítás mellett roppant pontos eredményekhez jutottam.

A kárszámbecslésnél leírt megfontolásból ugyanazt a mérőszámot használtam kárnagyságbecslésre is. Tweedie-eloszlást és log-kapocsfüggvényt használó általánosított lineáris modellel a predikált-valós arány 1,0003 lett az egész adathalmazon, amely jelentős pontosság. Elsősorban a gépjárművezető életkora és a tengelymetszet, azaz a konstans bizonyult statisztikailag szignifikánsnak, amelybe beépítettük a kitettséget. Továbbra is az adatok 80%-án tanuló, tízszeres keresztvalidációval kiértékelve 1,0083 volt a találati arányok átlaga, szórása 0,0708.

#### Konklúzió

Kárszám- és kárnagyságbecslés esetén is kiemelkedő pontossággal bírt az általánosított lineáris modell. Ehhez azonban fontos volt az aktuáriusi háttértudás a megfelelő mérőszám kiválasztásához a validálásban, illetve a matematikai statisztikai ismeret a kimeneteleket helyesen leíró eloszlás kiválasztásában.

### 3.1.10. Új módszerre való igény

**Definíció.** Ismérvtervezés (feature engineering): Az ismérvtervezés egy fontos folyamat az adatbányászat terén, amely magában foglalja a változók nyers ada-

tokból való kinyerését és átalakítását abból a célból, hogy az így kapott ismérvek megfeleljenek a modellezési feladatoknak és javítsák a modellek teljesítményét.

Bár az általánosított lineárs modellek hatékonynak bizonyultak a vezetői kockázat azonosításában és a díjkalkulációban, ezeknek a modelleknek vannak korlátai. A lineáris prediktorok nehezen találnak optimális megoldásokat, amikor komplex vagy magasdimenziójú adatokból tanulnak. Ez azt eredményezi, hogy az általánosított lineáris modell nem ragadja meg hatékonyan az adathalmazban lévő korrelációkat, ami gyenge általánosításhoz vezet[16]. Ez a korlátozás azt jelenti, hogy a lineáris prediktorokat használó modellek alkalmatlanok lehetnek a kockázatos viselkedések előrejelzésére nagy dimenziójú, például telematikai adatokból. Ahhoz, hogy egy általánosított lineáris modellből kinyerjük a releváns korrelációkat és a magyarázóváltozók hatásait az ütközési adatokra vagy kárigényekre, bonyolult kézi előfeldolgozás, ismérvtervezés és modellépítési folyamatok szükségeltetnek[11].

## 3.2. Együttes módszerek - XGBoost

Az együttes módszerek (ensemble methods) olyan technikák, amelyek célja a modellekben elért eredmények pontosságának javítása több modell kombinálásával egyetlen modell használata helyett. Az együttes modellek jelentősen növelik az eredmények pontosságát, emiatt népszerűvé váltak a gépi tanulásban.

Az együttes módszerek úgynevezett gyenge tanulókat - alaposztályozókat (base classifiers) vagy regressziókat - kombinálnak, hogy egy optimális előrejelző modellt hozzanak létre. Ezek a gyenge tanulók általában egyszerűek, gyenge prediktív teljesítményt nyújtanak önállóan, de együttesen, az együttes módszer alkalmazása révén pontosabb előrejelzést nyújtanak. Például a döntési fa egy fa alapú strukturális algoritmus, amelyet egyszerű döntési szabályok megtanulására használnak egy osztály vagy érték előrejelzésére. Egyetlen döntési fa azonban hajlamos a túlillesztésre és a változók kiválasztásának elfogultságára (selection bias), azaz egy olyan hibára, amely a megfigyelések nem véletlenszerű kiválasztását jelenti, és ennek eredményeként a vizsgálati eredmények nem reprezentálják megfelelően a mintát[29]. Döntési fák összekapcsolása egy olyan egyesített módszerben, mint a véletlen erdő (Random Forest) vagy a gradiensfokozás (Gradient Boosting), javítja ezeknek az alaposztályozóknak a teljesítményét, csökkentve a túltanulás hajlamot, és jelentősen növelve ezeknek a modelleknek a pontosságát.

### 3.2.1. Együttes módszerek a biztosításban

Az együttes gépi tanulási algoritmusok népszerűsége megnőtt a biztosítási szektorban. Az ezen alapuló modellek képesek bonyolult adatstruktúrákat megtanulni kevésbé alapos ismérvtervezés szükségessége nélkül, valamint értelmezhető eredményeket szolgáltatnak a biztosítók számára[10].

Az együttes módszerek, amelyeket a vezetési viselkedés kockázatértékelésében használnak, jelentősen javították az árazómodellek előrejelzési pontosságát, felülmúlva ezzel az általánosított lineáris modelleket ugyanebben a feladatban. Az együttes módszerek mellett szóló érvek között az egyik leghangsúlyosabb Guelman 2012-es tanulmánya[10], amelyben összehasonlította az általánosított lineáris modellt és a gradiensfokozó fák (Gradient Boosted Trees - GBT) módszerét az autóvezetők kockázati osztályozásában. A gradiensfokozó famodell képes lehet jobb teljesítményt nyújtani a hagyományos kockázati tényezők és baleseti adatok alapján történő kockázatárazásban, mint az általánosított lineáris modell. A Noll, Salzmann és Wuthrich által 2018-ban írt átfogó tanulmány[26] összehasonlította az együttes módszerek változatait a neurális hálózatokkal és az általánosított lineáris modellel. A tanulmány azt mutatta, hogy az együttes módszerek jobban teljesítettek, mint az általánosított lineáris modell, amikor a hagyományos árazási jellemzők alapján előrejeleztek kárigényeket. Különösképpen a gradiensfokozó fák teljesítménye múlta felül a véletlen erdőét ugyanazon az adathalmazon. A Pesantez-Narvaez és munkatársai által 2019-ben[28] és Maillart által 2021-ben[19] készített tanulmányok az együttes módszerek előrejelző képességét kombinálják a vezetői viselkedési adatokkal. Mailart cikkében az együttes módszereket összehasonlította az általánosított lineáris modellel. Az előbbi pontos és magyarázható eredményeket nyújtott, miközben kevesebb erőfeszítést igénylő előfeldolgozásra és ismérvtervezésre volt szükség.

### 3.2.2. Működési elv

Az XGBoost (eXtreme Gradient Boosting) egy hatékony és népszerű gépi tanulási módszer, amelyet gyakran használnak regressziós és osztályozási feladatokhoz egyaránt. Főként döntési fákon alapuló módszer, és matematikai háttere a gradiensfokozás koncepcióján alapul, amely a gradiens alapú optimalizáció és az additív modellépítés együttes használata. Az XGBoos ezt a technikát alkalmazza a predikciók javítására, amelyre neve is utal. Az alábbiakban leírom működési elvének néhány kulcsfontosságú pillérét, beleértve a fontos matematikai összetevőket és formulákat[4].

### Veszteségfüggvény

Adott egy $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ adathalmaz, ahol $x_i$ a bemeneti adat, $y_i$ pedig a hozzá kapcsolódó célérték. A modell célja, hogy megtanuljon egy $F(x)$ függvényt, amely minimalizálja a veszteségfüggvényt.

Az $L(y, F(x))$ veszteségfüggvény méri a modell hibáját. Gyakori veszteségfüggvény a négyzetes hiba regressziós feladathoz és a logaritmikus veszteség osztályozási feladathoz.

### Additív Modellépítés

Az XGBoost egy additív modellépítési folyamatot használ, ahol több gyenge tanulóból álló modell épül fel. Egy adott iterációban egy új modell (fa) hozzáadódik a meglévő modellekhez úgy, hogy csökkentse a hibát vagy a veszteséget.

Az additív modellépítés során egy meglévő modellhez hozzáadunk egy új komponenst, hogy csökkentsük a veszteséget. Ez matematikailag a következőképpen írható le:

$$F_{m+1}(x) = F_m(x) + \gamma_m \cdot h_m(x),$$

ahol $F_m(x)$ a meglévő modell, $h_m(x)$ a gyenge tanuló, és $\gamma_m$ a lépésméret vagy tanulási ráta (learning rate), amely gyakorlatilag azt mutatja, hogy az új hozzáadott gyenge tanulót milyen mértékben vegye figyelemben a modell.

### Gradiensfokozás és hibakorrekció

A gradiensfokozás lényege, hogy több gyenge tanulót (általában döntési fát) egyesít egy erősebb modell létrehozásához. A folyamat során minden egyes új fa a korábbi fák hibáit próbálja korrigálni. Ez a korrekció a gradiens mentén történik, azaz az új gyenge tanuló az előző modell gradiensét veszi alapul, hogy meghatározza, milyen irányban és mértékben kell módosítani a predikciót. A gradiens a veszteségfüggvény deriváltja a meglévő modell kimenete szerint:

$$g_i = \frac{\partial L(y_i, F_m(x_i))}{\partial F_m(x_i)},$$

ahol $g_i$ a gradiens az $i$-edik adatponthoz. Az új fák tehát azokban a régiókban tesznek javaslatokat, ahol a korábbi fák a leggyengébbek voltak.

### Gyenge tanuló illesztése

Az XGBoost által használt gyenge tanulók döntési fák, de a lényeg az, hogy egy fa által meghatározott struktúrát találjanak, amely csökkenti a veszteséget. Ezek a fák lépésről lépésre választják szét az adatokat a legmegfelelőbb elválasztási szabályok alapján. Az XGBoost különböző optimalizálási technikákat használ a döntési fák hatékonyabb létrehozásához és alkalmazásához.

### Regularizáció és paraméterek

Az XGBoost erősen fókuszál a regularizációra, ami segít elkerülni a túlillesztést (overfitting). A regularizáció különböző paramétereket alkalmaz: Pythonban például a fa mélységét korlátozó 'max_depth' vagy a fa súlyait szabályozó 'lambda' és 'alpha' értékeket. A regularizáció az alábbi formában jelenik meg a veszteségfüggvényben:

$$L(y, F_m(x)) + \Omega(h),$$

ahol $\Omega(h)$ a regularizációs kifejezés, amely tartalmazhat L1 és L2 regularizációs tagokat.

### Metszés (pruning)

A metszés egy olyan technika, amely a fa méretének csökkentésével javítja az általánosítást.

### Parallelizáció és hatékonyság

Az XGBoost egyedi tervezésének köszönhetően képes párhuzamosan végrehajtani több feladatot, ami gyorsabb működést eredményez. Ez különösen hasznos nagy adathalmazok esetén, mivel a tanulási folyamat időigényes lehet.

### Korai megállás és keresztvalidáció

Az XGBoost támogatja a korai megállást (early stopping), ami azt jelenti, hogy a modell tréningje leáll, ha egy bizonyos számú iteráció után nem mutat javulást a validációs adatokon. Emellett a keresztvalidáció (cross-validation) használata segíti a modell értékelését és az általánosítást.

**Összegzés**

Az XGBoost matematikailag a gradiensfokozás elvén alapul, ahol egy additív modellépítési folyamat során gyenge tanulókat (általában döntési fákat) adnak hozzá a meglévő modellhez, hogy csökkentsék a veszteséget. A gradiens alapú optimalizáció és a regularizáció kulcsfontosságú elemek a modell hatékonyságának és általánosításának biztosításában.

Ezek a főbb elvek és mechanizmusok, amelyek mentén az XGBoost működik, és amelyeknek köszönhetően a módszer gyorsan népszerűvé vált a gépi tanulási közösségben. Az XGBoost alkalmazása előtt fontos figyelmet fordítani a megfelelő paraméterek beállítására és a modell validálására, hogy elkerüljük a túlillesztést vagy az alulillesztést.

### 3.2.3. Modellimplementáció

Az XGBoost használatához Pythonban az 'xgboost' nevű könyvtárra volt szükségem. Ezzel a könyvtárral különféle feladatok végezhetőek, beleértve a döntési fa-alapú osztályozást és regressziót, valamint a modell kiértékelését és paraméterhangolását.

### 3.2.4. Adatelőkészítés

A Python XGBoost regresszora egy speciális adatmátrixszal dolgozik, amely tudja kezelni a kategorikus változókat, ezért erre itt sem kellett külön figyelmet fordítanom. A döntési fák nem érzékenyek a skálázásra, azonban a regularizációs tag miatt sztenderdizáltam az adatokat a 'scikit-learn' könyvtár 'StandardScaler()' függvényével. A sztenderdizálással nem ronthattam a modell előrejelző képességét.

### 3.2.5. Hiperparaméter-optimalizálás

A hiperparaméter optimalizáláshoz továbbra is a már általánosított lineáris modellnél bevezetett becsült-valós arányt használtam validáló mérőszámként az összkárszám és az összkárnagyság becslése esetén. Mivel a 'num_boost_round' paramétert ötezerre állítottam, ezért 5000 alkalommal hajtottam végre gradiensfokozást, azaz ennyi fából áll az általam összeállított XGBoost modell. A keresztvalidációt a 'scikit-learn' könyvtár 'KFold()' függvényével végeztem. Ezzel a 'learning_rate' (vagy 'eta') tanulási rátát, a 'max_depth' maximális famélységet, az 'objective' célfüggvényt, valamint a fa súlyait szabályozó 'lambda' és 'alpha' pa-

ramétereket finomhangoltam. Ez utóbbi kettő görög betű közül az előbb felsorolt alapbeállításként 1, az utóbb felsorolt 0. Ahogy növeljük ezeket az értékeket, úgy egyre konzervatívabb lesz a modell, vagyis egyre kevésbé hajlamos a túltanulásra.

Optimális értéknek az alábbiakat kaptam.

- Kárszámbecslésre
  - 'learning_rate': 0.7,
  - 'max_depth': 5,
  - 'lambda': 1,
  - 'alpha': 0,
  - 'objective': 'count:poisson', azaz Poisson-regresszió.
- Kárnagyságbecslésre
  - 'learning_rate': 0.3,
  - 'max_depth': 1,
  - 'lambda': 1,
  - 'alpha': 0
  - 'objective': 'reg:tweedie', azaz Tweedie-regresszió,
    - 'tweedie_variance_power': 1,01.

Tehát a fa súlyait szabályozó hiperparaméterek optimalizálására nem is lett volna szükség, és a kárnagyság becslése esetén még a tanulási ráta módosítására sem. Bár az 'objective' célfüggvény beállítására próbálkoztam a 'reg:squarederror' négyzetes veszteséggel is, az általánosított lineáris modellnél látott eloszlásokból származó és paraméterű célfüggvények mutatkoztak leghatékonyabbnak.

### 3.2.6. Eredmények

#### Kárszámbecslés

Tízszeres keresztvalidáció mellett 0,9991 átlagos becsült-valós találati arányt sikerült elérnem. Ez azt jelenti, hogy a modell szinte hibátlanul találja el az állomány összkárszámát csakúgy, mint az általánosított lineáris modell. Az arány szórása 0,047. A legfontosabb változók az XGBoost számára is a gépjárművezető kora, majd a vázszerkezet. Ezt a 'gain' nevű score-ból lehet megtudni, amely azt mutatja, hogy mennyi volt az átlagos nyereség azokban a fametszésekben, amelyekben az ismérvet használtuk.

#### Kárnagyságbecslés

Szintén tízszeres keresztvalidációval sikerült elérni átlagosan 0,29%-os relatív hibát az állomány összkárának előrejelzésében: 1,0029. A becsült-valós arány szórása pedig 0.0597 lett. Ez ugyancsak kellően nagy pontosságra vall. Az ismérvfontosság itt is a a GLM-hez hasonlóan alakult.

#### Konklúzió

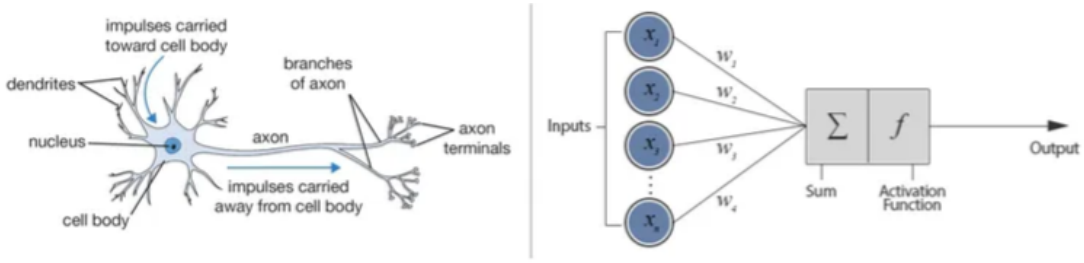
Az XGBoost hasonló eredményeket adott a GLM-hez, ám a hiperparaméteroptimalizáció több időt igényelt, a modell értelmezhetősége sem annyira egyértelmű, mert a változók szignifikanciájára nincs statisztikai próba.

### 3.2.7. Új módszerre való igény

Az együttes módszerek használatának is vannak korlátai a biztosítói kockázatárazásban, mert csupán alapjaikban értelmezhetőek. Ebből kifolyólag az elégséges modellértelmezhetőség elérése ellenőrzési vagy pénzügyi szabályozási célokra igencsak bonyolult feladatnak mutatkozik ezekkel a modellekkel a biztosítók számára[12]. Az együttes módszerek tanítása is kihívást jelenthet. A hiperparaméterek finomhangolása olykor nehéz feladat, és ezért az extra erőfeszítésért járó jutalom esetenként elhanyagolható egyszerűbb modellekhez, például a logisztikus regresszióhoz viszonyítva.[28].

## 3.3. Neurális hálózatok

A mesterséges neurális hálózatok (Artificial Neural Networks - ANNs) a gépi tanulás alapkövei, amelyeket az emberi agy szerkezetének és működésének utánzásával terveztek, hogy komplex mintázatokat modellezzenek az adatokban. Ebben a részben mélyrehatóan foglalkozom a neurális hálózatok matematikai alapjaival, különös tekintettel a visszaterjesztési (backpropagation) algoritmusra és annak kiterjesztéseire.




3.3.1. ábra. Biológiai és mesterséges neuron
Forrás: `https://towardsdatascience.com`

### 3.3.1. Neurális hálózatok a biztosításban

A mélytanulás drasztikusan megváltoztatta a természetes nyelvfeldolgozás, a képfelismerés és az önvezető autók területét. Ezeknek a hatékony modelleknek azonban korlátozott a felhasználása a biztosítók árazási és kockázatbesorolási feladataiban. A mélytanulás egy többrétegű megközelítés a tanuláshoz, ahol minden réteg látens jellemzőket nyer ki, és frissíti a kapcsolódó pontokat és súlyokat annak függvényében, hogy mennyire relevánsak az értékelésben[8]. Az előre irányuló és visszaterjesztési lépések kombinációja biztosítja, hogy minden súly és eltolás (bias) megfelelően állíttasson be a hálózat rejtett rétegei között, ahol minden réteg kimenetele a megtanult fontos ismérvek egy vektora, és ezt a kimeneti vektort továbbítja a következő rétegnek.

Csak korlátozott számú tanulmány foglalkozik a mélyháló-modellek alkalmazásával a biztosítási kockázatok osztályozásában és árazásában. Ám egyre szélesebb körben válnak elfogadottá ezek a kiemelkedően pontos modellek. A mélyháló-modellek elterjedése előtt számos tanulmány vizsgálta a mélytanulás megvalósíthatóságát a biztosítói kockázatárazásban. Paefgen és munkatársai 2013-ban bemutatták, hogy a mélytanulás milyen hatékony lehet a telematikai adatokból előrejelezhető baleseti kockázatokat tekintve[27]. A szerzők egy mélyháló-modellt hasonlítanak össze logisztikus regresszióval és döntési fa modellekkel. Bár a mélyháló-modell különböző metrikákban felülmúlta a többi modellt, Paefgen és munkatársai 2013-ban az alacsony interpretálhatóság miatt inkább a logisztikus regressziót választották legmegfelelőbbnek. Egy hasonló tanulmányban Baecke és Bocca 2017-ben telematikai adatok felhasználásával hasonlított össze egy mélyháló-modellt egy véletlen erdővel és a logisztikus regresszióval a sofőrkockázatok osztályozására[3]. Ahogy Paefgenék, Baecke és Bocca is úgy döntöttek, hogy a logisztikus regresszió volt a legjobb választás, bár a mélytanulás több kategóriában felülmúlta a többi modellt.

Az említett tanulmányok mellett számos szerző vezetett be kifejezetten biztosítási célokra szabott mélytanulási modelleket. Például Noll és társai 2018-ban egy modelleket összehasonlító tanulmányukban egy neurális hálózat architektúrát mutattak be, amelyet egy telematikai adatállományra alkalmaztak biztosítási célokból[26]. Nollék jelentős változtatásokat hajtottak végre ezen a modellen, és a legjobb teljesítményt egy, csak a kitettséget mint ismérvet figyelembe vevő réteg hálózatba történő beemelésével érték el, amely felülmúlta az általánosított lineáris modellt és az együttes módszereket a kárigény-előrejelzés feladatában.

Egy alternatív mélyháló-architektúra, amelyet Siami, Naderpour és Lu mutattak be 2021-es cikkükben[31], háromlépéses megközelítést alkalmaz a vezetői viselkedés kinyerésére a későbbi kockázatelemzéshez. Tanulmányukban Siami és társai egy önszerveződő térkép[2] (self-organising map - SOM) alkalmazásával csökkentik a telematikai adatok összetettségét. Egy kilencrétegű mély autoencoder kinyeri a releváns ismérveket, mielőtt egy K-közép algoritmus klasztterezné az adathalmazt az utolsó két lépésben. A kapott klaszterek azonosítják a jellemző kockázatos vezetői viselkedéseket vagy mintákat, amelyek hozzájárulnak a vezető összesített kockázatához.

A táblázatos adatokból való tanulás kihívást jelenthet a mélytanulási modellek számára az optimális megoldás megtalálásában, mivel a ritka és heterogén táblázatos adatkészletek korlátozzák a mélytanulási modell képességét abban, hogy megfelelő következtető elfogultságot[3] (inductive bias) találjon[1]. Emellett a szabályozások miatti szemcsés telematikai adatokhoz való hozzáférés korlátozása, valamint az alacsony értelmezhetőség komoly akadályt jelent a mélytanulási modellek széles körű elfogadásában a biztosítási kockázatárazás terén[3].

[2]Az önszerveződő térkép vagy önszerveződő jellemzőtérkép egy felügyelet nélküli tanulási technika, amellyel egy magas dimenziós adatkészlet alacsony dimenziós (általában kétdimenziós, innen a térkép elnevezés) ábrázolását állítják elő, miközben megőrzik az adatok topológiai szerkezetét. Ez egy mesterséges neurális háló, amelyet megalkotója, a finn Kohonen professzor után szoktak még Kohonen-térképnek vagy Kohonen-hálónak is nevezni.

[3]A következtető elfogultság egy adattudományokban használt fogalom, amely azokra az előfeltevésekre utal, amelyeket egy tanulási algoritmus használ, hogy általánosításokat eszközöljön a tanuló adatokból az ismeretlen adatokra vonatkozóan. Mivel a tanuló algoritmusoknak véges mennyiségű adatból kell tanulniuk, szükségük van bizonyos előfeltevésekre ahhoz, hogy helyes következtetéseket vonjanak le a jövőbeli, ismeretlen adatokra vonatkozóan. Példa erre a lineáris regresszióban az a feltételezés, hogy a válaszváltozó és a bemeneti változók közötti kapcsolat lineáris; vagy a neurális hálóknak az architektúrája, mivel többek között az aktivációs függvény, a rétegek száma és a neuronok kapcsolódási mintái befolyásolják, hogy milyen típusú függvényeket tud megtanulni hatékonyan.

### 3.3.2. A neurális hálózatok alapfogalmai

Egy neurális hálózat csúcsok vagy neuronok összekapcsolt rétegeiből áll. Minden ilyen kapcsolat egy súlyt reprezentál, amely a tanulás során folyton frissül. A jellemző rétegek a következők:

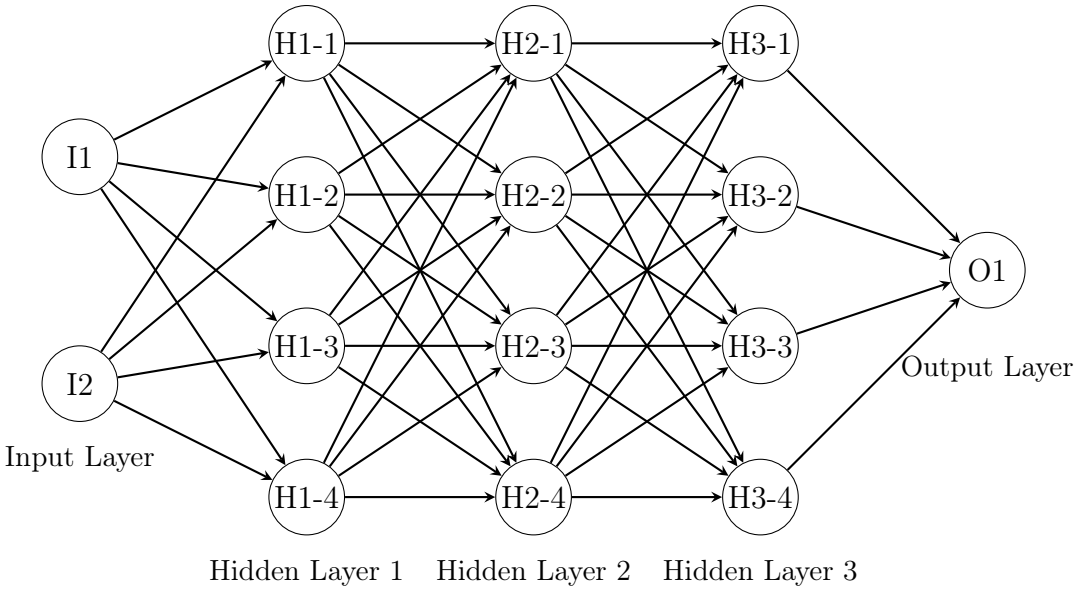
- Bemeneti réteg (input layer): Fogadja az adatok jellemzőit.
- Rejtett rétegek (hidden layer): Közbenső feldolgozó rétegek.
- Kimeneti réteg (output layer): Előállítja a becslést vagy az osztályozást.
- Korszak (epoch): akkor zárul le egy korszak egy tanulási algoritmus során, amikor az összes tanulóadatot egyszer áteresztettük a modellen.

#### Matematikai modell

Minden neuron egy neurális hálózatban egy $y$ kimenetet számít ki a bemeneti $x_i$ értékek és egy $b$ eltolás (bias) súlyozott összegéből. Az eltolásparamétert gyakran nem tekintjük külön, hanem eltolássúlyként együtt kezeljük a többi súllyal. A kimenetet tipikusan egy nemlineáris $f$ aktivációs függvényen keresztül vezetik át. Az $l$ réteg $j$ neuronja esetén a kimenet a következőképpen adható meg:

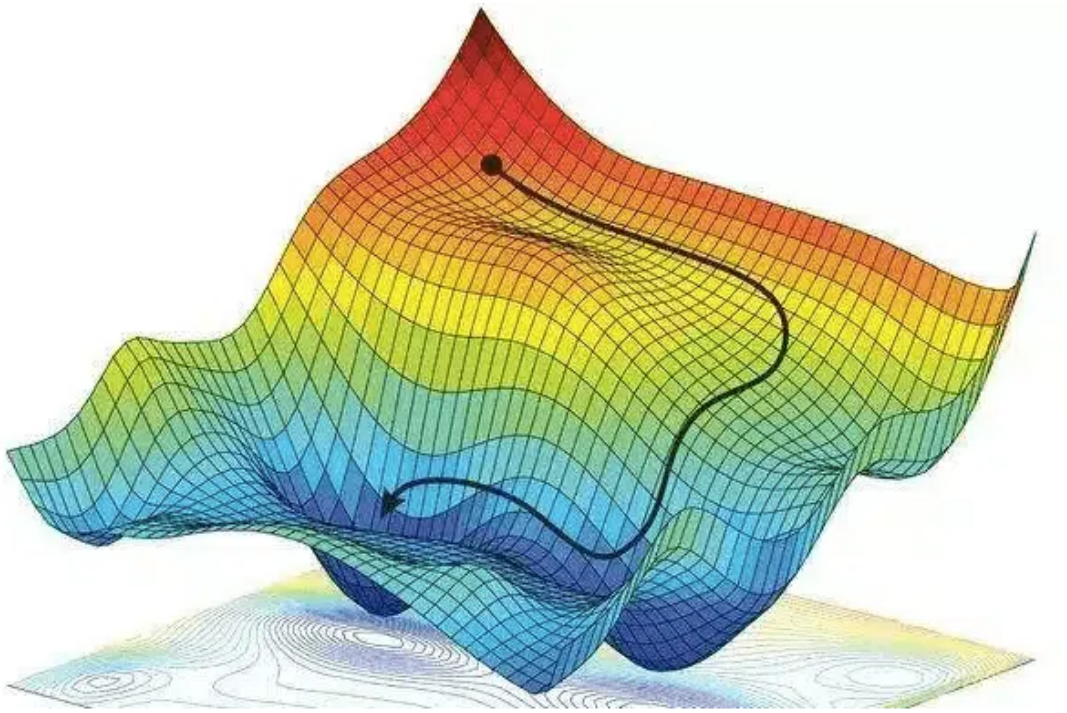
$$y_j^{(l)} = f\left(\sum_i w_{ij}^{(l)} x_i^{(l-1)} + b_j^{(l)}\right) \tag{3.1}$$

ahol $w_{ij}^{(l)}$ az $(l-1)$ réteg $i$ neuronját az $l$ réteg $j$ neuronjával összekötő súly.



### 3.3.3. Gradiensereszkedés

A gradiensereszkedés optimalizáló algoritmus egy differenciálható függvény lokális minimumának megtalálására. A legtöbb neurális háló ezt a módszert használja a veszteségfüggvény minimalizálásához. Az algoritmus azon alapszik, hogy ha egy $f$ többdimenziós függvény differenciálható egy $\mathbf{x}$ pont környezetében, akkor ebből a pontból $f$ úgy csökkenthető a lehető leggyorsabb ütemben, ha az $\mathbf{x}$ pontbeli gradienssel ellentétes irányba mozdulunk el.



3.3.2. ábra. A gradiensereszkedés folyamata
Forrás: `https://easyai.tech/en/ai-definition/gradient-descent/`

**Definíció.** Többváltozós derivált: Tegyük fel, hogy $f : \mathbb{R}^d \to \mathbb{R}^k$ értelmezve van $\mathbf{x}$ egy $B_r(\mathbf{x})$ környezetében. Azt mondjuk, hogy $f$ deriválható $\mathbf{x}$-ben, és az itt vett deriváltja $S : \mathbb{R}^d \to \mathbb{R}^k$ lineáris, ha van olyan $r : \mathbb{R}^d \to \mathbb{R}^k$ függvény, hogy elég kis $||\boldsymbol{\Delta}||$ esetén

$$f(\mathbf{x}+\boldsymbol{\Delta}) = f(\mathbf{x}) + S \cdot \boldsymbol{\Delta} + r(\boldsymbol{\Delta}), \text{ ahol } \lim_{\boldsymbol{\Delta}\to\mathbf{0}} \frac{r(\boldsymbol{\Delta})}{||\boldsymbol{\Delta}||} = \mathbf{0}.$$

Tehát állításként megfogalmazhatjuk a következőt.
**Állítás.** Ha a $\gamma \in \mathbb{R}^+$ tanulási ráta elég kicsi, akkor $\mathbf{x_{n+1}} = \mathbf{x_n} - \gamma \nabla f(\mathbf{x_n})$ esetén egyrészt $f(\mathbf{x_n}) \geq f(\mathbf{x_{n+1}})$, másrészt minden $\mathbf{v}$ irány közül, ahol $||\mathbf{v}||$ kicsi,

a gradiens irányával ellentétes elmozdulás jelenti a legmeredekebb ereszkedést, azaz $f(\mathbf{x_n} + \mathbf{v}) \geq f(\mathbf{x_{n+1}})$.

**Bizonyítás.** A bizonyítás kulcsa a Cauchy–Bunyakovszkij–Schwarz-egyenlőtlenség. Azt látjuk be, hogy a gradiensirányú elmozdulás a legmeredekebb egy valós számokba képező többdimenziós függvény esetén. A definíció szerint az $\mathbf{x}$ pontból $\mathbf{v}$ vektorral elmozdulva:

$$f(\mathbf{x} + \mathbf{v}) - f(\mathbf{x}) = \langle \nabla f(\mathbf{x}); \mathbf{v} \rangle + r(\mathbf{v}).$$

Ha $||\mathbf{v}|| = 1$, akkor $\langle \nabla f(\mathbf{x}); \mathbf{v} \rangle$ adja meg a függvényfelület $\mathbf{v}$ irányban vett meredekségét. Ez becsülhető a Cauchy–Bunyakovszkij–Schwarz-egyenlőtlenséggel:

$$|\langle \nabla f(\mathbf{x}); \mathbf{v} \rangle| \leq ||\nabla f(\mathbf{x})|| \cdot ||\mathbf{v}|| = ||\nabla f(\mathbf{x})||.$$

Az egyenlőtlenség akkor és csak akkor teljesül egyenlőséggel, azaz a bal oldal akkor és csak akkor maximális, ha $\mathbf{v}$ a gradiens irányát adja meg, azaz

$$\mathbf{v} = \nabla f(\mathbf{x}) \cdot \frac{1}{||\nabla f(\mathbf{x})||}. \quad \square$$

Ez az alapja sok numerikus szélsőértékkereső módszernek: a következőkben bemutatott visszaterjesztésnek és valójában a gradiensfokozásnak is, amelyet az XGBoostnál használtunk.

### 3.3.4. A visszaterjesztési algoritmus

A visszaterjesztés egy módszer, amelyet a neurális hálózathoz tartozó veszteségfüggvény súlyok szerinti gradiensének kiszámítására használnak. Ez kulcsfontosságú a neurális hálózatok gradienssereszkedéssel történő tanításához.

#### Előre irányuló lépés

Az előre irányuló lépés során a bemeneti adatok áthaladnak a hálózaton, hogy egy kimeneti eredményt generáljanak. Minden neuron esetében a következők számítódnak:

$$z_j^{(l)} = \sum_i w_{ij}^{(l)} x_i^{(l-1)} + b_j^{(l)}$$

$$y_j^{(l)} = f(z_j^{(l)})$$

#### Veszteség- vagy költségfüggvény

Az $L$ veszteségfüggvény méri a predikált kimenetek és a tényleges célértékek közötti eltérést. Ezt szeretnénk a súlyok és eltolások beállításával minimalizálni. Gyakori választás regresszióhoz az átlagos négyzetes eltérés (Mean Squared Error - MSE) és osztályozáshoz a keresztentrópia (Cross-Entropy Loss).

#### Hátrafelé irányuló lépés

A veszteségfüggvény minden egyes súly szerinti parciális deriváltja a láncszabály segítségével kerül kiszámításra a háló kimenetelétől visszafelé lépkedve, innen ered a visszaterjesztés elnevezés. A veszteségfüggvény $w_{ij}^{(l)}$ súly szerinti parciális deriváltja az alábbi módon számítható:

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial y_j^{(l)}} \cdot \frac{\partial y_j^{(l)}}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}},$$

ahol

$$\frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}} = x_i^{(l-1)}.$$

#### A súlyok frissítése

Ezután a súlyokat a gradiensereszkedésnél elmondottaknak megfelelően frissítjük, hogy csökkentsük a veszteséget. A $t$-edik lépésben a következőképpen kapjuk az új súlyvektort:

$$w_t = w_{t-1} - \gamma \nabla L,$$

ahol $\gamma$ a tanulási ráta (learning rate), $\gamma \nabla L$ a lépésméret (step size). A súlyoknak egy ilyen frissítése az algoritmus során egy lépés (step).

### 3.3.5. A visszaterjesztés fajtái

#### Kötegelt gradiensereszkedés (Batch Gradient Descent - BGD)

Vanília gradiensereszkedésként is ismert, a gradienssüllyedés legegyszerűbb változata. Kötegelt gradiensereszkedés esetén a teljes tanuló-adathalmazt használják a veszteségfüggvény egyes paraméterek szerinti parciális deriváltjainak kiszámítására. Az összes tanulóadatra kiszámolt gradiensek átlagát vesszük, majd ezt az átlagos gradienst használjuk a paramétereink frissítéséhez. Tehát egy korszakban csak egy lépést teszünk a gradienssüllyedésben. Ez számításilag költséges lehet

nagy adatkészletek esetén, de garantálja a konvergenciát a veszteségfüggvény helyi minimumához.

### Sztochasztikus gradienssereszkedés (Stochastic Gradient Descent - SGD)

A sztochasztikus gradienssereszkedés a gradienssüllyedés egy olyan változata, amely frissíti a modell paramétereit az adathalmaz minden egyes tanulóadatához. A kötegelt gradienssüllyedéstől eltérően, amely a teljes adatkészletet használja a gradiensek kiszámításához, a sztochasztikus gradienssereszkedés egy véletlenszerűen kiválasztott tanulóadat alapján frissíti a paramétereket. Ez gyorsabb konvergenciához vezethet, mivel a frissítések gyakoribbak, de a frissítések véletlenszerűsége miatt több ingadozással is járhat a veszteségfüggvényben.

### Minikötegenkénti gradienssereszkedés (Mini-batch Gradient Descent)

A minikötegenkénti gradienssereszkedés az előző két módszer ötvözete. A tanulóadatokat egyforma méretű minikötegekre (mini-batch) osztja fel véletlenszerűen, vagyis ez is sztochasztikus megközelítés, és a gradienst minden lépésben újabb minikötegen becsli. Ez szintén nagy mintaelemszám esetén válhat a hasznunkra, mert így egy lépésben nem az összes adatot vesszük figyelembe, ezért gyorsítja az eljárást. Mivel azonban most sem az egész mintát használjuk egy lépésben, ezért ingadozhat a veszteségfüggvényünk a frissítések során, de kisebb mértékben, mint az előbbi technika esetén. Ezt képletekkel is megfogalmazhatjuk.

Vegyünk ismét egy $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ adathalmazt, ahol $x_i$ a bemeneti adat, $y_i$ pedig a hozzá kapcsolódó célérték. Legyen $\hat{y}_i$ a háló becslése, amely a bementi adatoknak és a háló súlyainak egy $g$ függvénye. Jelölje $B$ a kötegbe tartozó adatok halmazát, melynek elemszáma $|B|$. Ekkor a $TL$ teljes veszteségfüggvény gradiense a $t$-edik korszakban az alábbi:

$$\nabla TL(w_t) = \sum_{i=1}^{n} \nabla L(y_i; g(x_i; w_t)) \approx \frac{n}{|B|} \sum_{i \in B} \nabla L(y_i; g(x_i; w_t)).$$

Mivel egy korszakban minden tanulóadatot egyszer használunk fel, ezért egy epoch során minden köteget is egyszer lát a modell. Mivel az utolsó egyenlet csak közelítőleg teljesül, ezért a paraméterek fluktuálva tartanak az optimális értékükhöz.

### 3.3.6. A visszaterjesztés hatékonyságának fejlesztése

#### Lendület (momentum)

A lendület olyan technika, amely a gradiensvektorokat a helyes irányokba lendíti, így gyorsabb konvergenciát eredményez. Ráadásul a lendület segíthet a becsülendő paramétereknek kiszabadulni a lokális minimumból. A súlyok a következőképpen számíthatóak:

$$v_{t+1} = \gamma v_t + \eta \frac{\partial L}{\partial \theta_t}$$

$$\theta_{t+1} = \theta_t - v_{t+1},$$

ahol $\gamma$ a lendület együtthatója, $\eta$ a tanulási ráta, és $\theta$ a modell paramétereit jelenti.

#### Gyök átlagos négyzetes terjesztés (Root Mean Square Propagation - RMSProp)

Ez a technika a gradienssereszkedés ingadozását hivatott csökkenteni egy adaptív tanulási ráta által. A módszer gyors zuhanás esetén fékez, lassú ereszkedés esetén viszont gyorsít. Matematikai felírása a következő.

Írjuk fel a négyzetes gradiensek időben lecsengő súlyozott átlagát:

$$h_t = \beta h_{t-1} + (1 - \beta)(\nabla L(\theta_t))^2,$$

ahol $\beta$ a lecsengési együttható. Ezután az adaptív tanulási ráta a következőképpen számolható:

$$\eta_t = \frac{\eta}{\sqrt{h_t} + \varepsilon},$$

ahol $\varepsilon > 0$ egy kis szám a nullával való osztás elkerülése végett. Ezzel az új paraméterek:

$$\theta_{t+1} = \theta_t - \eta_t \nabla L(\theta_t).$$

#### Adaptív lendületbecslés (Adaptive Moment Estimation - Adam)

Az Adam optimalizáló kombinálja a lendület és a gyök átlagos négyzetes terjesztés előnyeit, biztosítva a gyors és egyben stabil konvergenciát a sztochasztikus gradienssereszkedési módszerek esetén is különféle problémák megoldása során. Matematikai képletekkel a következőképpen fogalmazható meg[15].

Jelölje az $L$ veszteségfüggvény gradiensét a modell aktuális $\theta_t$ paramétereire

vonatkozóan:

$$g_t = \frac{\partial L}{\partial \theta_t}.$$

Ezzel az Adam becsli először az $m_t$ első momentumot a lendülettel, amely a gradiensek mozgó átlaga:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t,$$

majd a $v_t$ második momentumot az RMSProp módszerrel, amely a gradiensek négyzetének mozgó átlaga:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2.$$

Mivel az $m_t$ és $v_t$ becslések nullákból indulnak, torzíthatnak az iterációk korai szakaszában. Az Adam ezért korrigálja ezeket az értékeket:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

Végül az Adam frissíti a modell paramétereit az alábbiak szerint:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

A paraméterek jelentése az alábbi:

- $\eta$: tanulási ráta, amely meghatározza a lépésméretet.
- $\beta_1$ és $\beta_2$: a momentumbecslések exponenciális súlyozásának paraméterei, Pythonban alapbeállításként $\beta_1 = 0.9$ és $\beta_2 = 0.999$.
- $\epsilon$: egy kis konstans, amely stabilizálja a számítást, elkerülve a nullával való osztást.

Az adaptív tanulási ráták és a lendület alkalmazása miatt az Adam hatékonyan működik nagy adatbázisok és magas dimenziós paraméterterek kezelésekor, amely nagyon sikeressé tette. Többek között a ChatGPT is használja. Ennek ellenére az Adam sem konvergál garantáltan globális optimumba.

### 3.3.7. Túllillesztés elleni módszerek

Neurális hálóknál is lehetőségünk adódik a korai megállásra. Ha adott számú epoch során a validációs adatokon nem javul a teljesítmény, akkor a tanulás leáll. Ezt egy úgynevezett türelmi, Pythonban 'patience' néven elérhető paraméterrel lehet beállítani.

Neurális hálóknál egy másik speciális módszer túllillesztés ellen a lemorzsolódási (dropout) réteg. Ez a megelőző réteg neuronjainak értékeit minden korszak során adott $q$ valószínűséggel nullázza, ezzel kényszerítve a hálózatot alternatív megoldások megtalálására. A megmaradó neuronok kimenetét skálázza az $\frac{1}{1-q}$ tényezővel kompenzációként. Fontos azonban, hogy a következtetési fázis során - vagyis amikor új, nem látott adatokra vonatkozó előrejelzéseket készítünk - nem alkalmazunk lemorzsolódást. Tehát a kiesett neuronok nélküli teljes hálózatot használják az előrejelzésekhez.

### 3.3.8. Modellimplementáció

Mélyhálómodellek építéséhez Pythonban a 'keras' nevű könyvtárra volt szükségem. Ezzel könnyedén készíthetőek előrecsatolt mesterséges neurális hálók, és alkalmazhatóak a fentebb leírt túllillesztés elleni és gradienssereszkedést segítő módszerek. Az adatokat ismét a 'scikit-learn' könyvtár 'StandardScaler()' függvényével sztenderdizáltam.

### 3.3.9. Adatelőkészítés

A mesterséges neurális hálózatok adatainak előfeldolgozása során a skálázás kulcsfontosságú lépés. Ez biztosítja, hogy az összes bemeneti funkció hasonló nagyságrendű legyen, ami elősegítheti a modell gyorsabb konvergálását, és segít elkerülni a kiugró ismérvek felé torzítást.

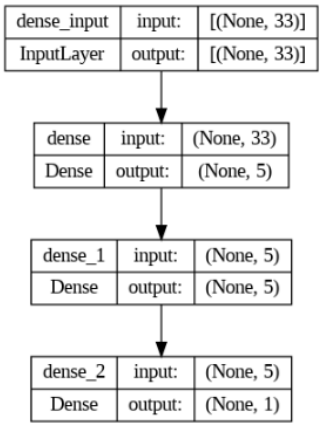
Ezen kívül most elkerülhetetlen volt a kategorikus változók szintenkénti elkódolása (one-hot encoding). Ez azt jelenti, hogy minden kategorikus változóhoz annyi új változót készítünk, ahány kategóriát fed le. Ha egy adat az egyik kategóriába esik, akkor az ennek megfelelő változó 1-es értéket vesz fel, különben 0-t. Ehhez ugyanennek a könyvtárnak a 'OneHotEncoder()' objektumát használtam.

Végül a keresztvalidációt szintén a 'scikit-learn' 'KFold()' függvényével végeztem, ahogy korábban is tettem.

### 3.3.10. Eredmények

#### Kárszámbecslés

Veszteségfüggvényem a legkisebb négyzetes hiba 'mean_squared_error' volt. Aktivációs függvényt a kimeneti rétegen nem alkalmaztam, tehát ott az identitás, a többi rétegen pedig a ReLU (rectified linear unit) az, amely egyszerűen a pozitív részét veszi az inputnak. Az architektúra a következő volt:

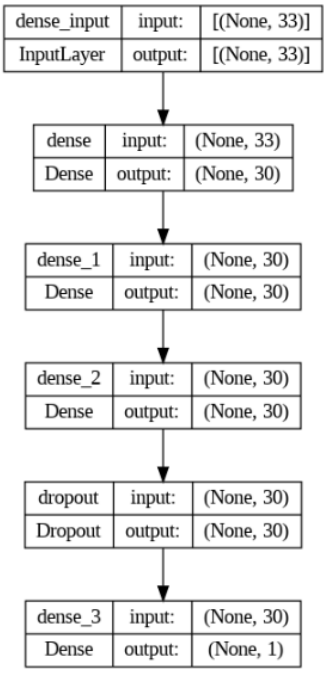



3.3.3. ábra. Mélyhálómodell kárszámbecslésre.

Ezzel az egyszerű modellel is 0,9626 lett a predikált-valós találati arányom.

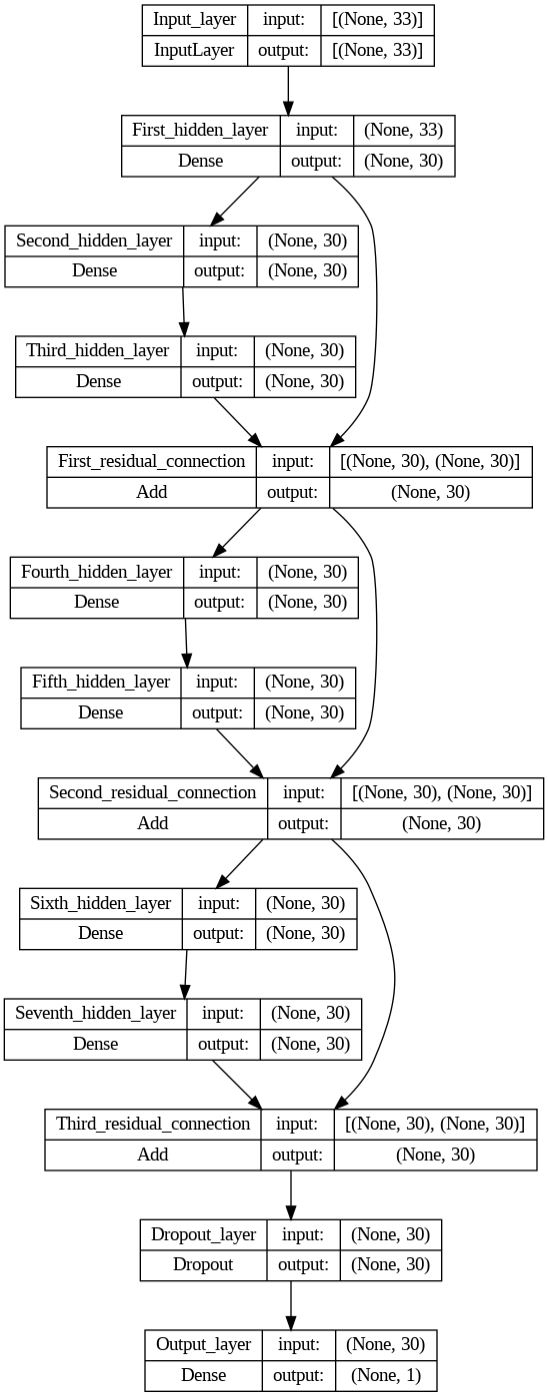
#### Kárnagyságbecslés

A kárnagyság becsléséhez ugyanazokat az eszközöket használtam, mint a kárszámbecslésnél, azzal a különbséggel, hogy az adatokat nem csupán egy 5-, hanem egy 30-dimenziós térbe ágyaztam be, valamint alkalmaztam egy lemorzsolódási réteget is, de a szokásos célértékem csupán a 0,8749 arány lett.




3.3.4. ábra. Mélyhálómodell kárnagyságbecslésre.

Próbálkoztam az alábbi, bonyolultabb struktúrájú, reziduális blokkokból álló hálóval. Ekkor a becsült-valós állományszintű kár aránya 0,9435 lett, ami javulás az egyszerű előrecsatolt hálóhoz képest, de ez is gyengébb teljesítmény, mint az eddigi modelleké.




3.3.5. ábra. Reziduális blokkokból épült mesterséges neurális hálózat

### Konklúzió

A saját készítésű, hagyományos mesterséges neurális hálókkal nem sikerült utolérnem a GLM és az XGBoost teljesítményét. Az eredmények azonban figyelemre méltóak, és tovább javíthatóak az architektúra precizitásával.

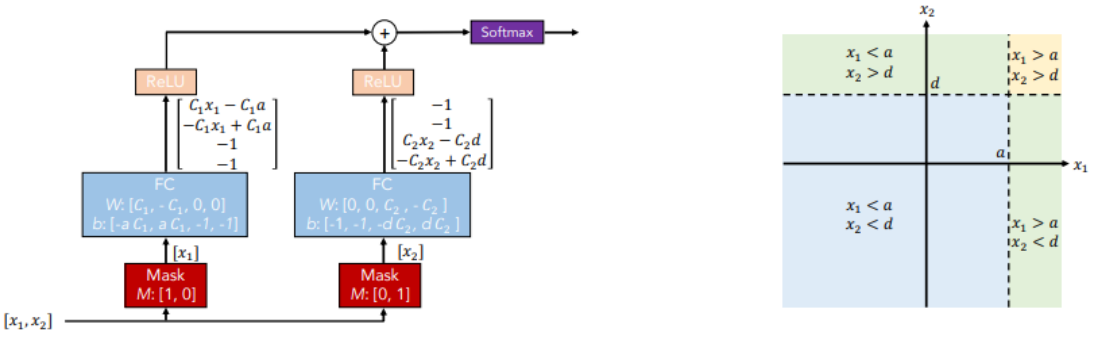
## 3.4. TabNet (Tabular data learning neural Network)

A TabNet egy modern neurális hálózati architektúra, amelyet kifejezetten táblázatos adatokkal való munkára fejlesztettek ki. A hagyományos neurális hálózatokkal ellentétben, amelyek gyakran gyengén teljesítenek táblázatos adatokon, a TabNet számos innovatív mechanizmust alkalmaz, hogy jobban kihasználja az ilyen adatok sajátosságait.

Bár a TabNet is mesterséges neurális hálózat, ennek a modellnek külön részt szentelek szakdolgozatomban. Fontos megjegyeznem, hogy a működési elvet leíró matematikai formulákban a mátrixokkal történő műveletek, mint például a mátrixszorzás, elemenként értendő, mivel Arik és Pfister[1] is ezzel a jelöléssel él.

### 3.4.1. A Tabnet felépítése és a mögöttes elképzelés

Láttuk, hogy a döntési fák sikeresen alkalmazhatók táblázatos adatok tanulására. Speciális kialakítással a hagyományos mélyháló-modellek építőelemei használhatók döntési faszerű kimeneti terek meghatározására. Ezt reprezentálja a 3.4.1 ábra.



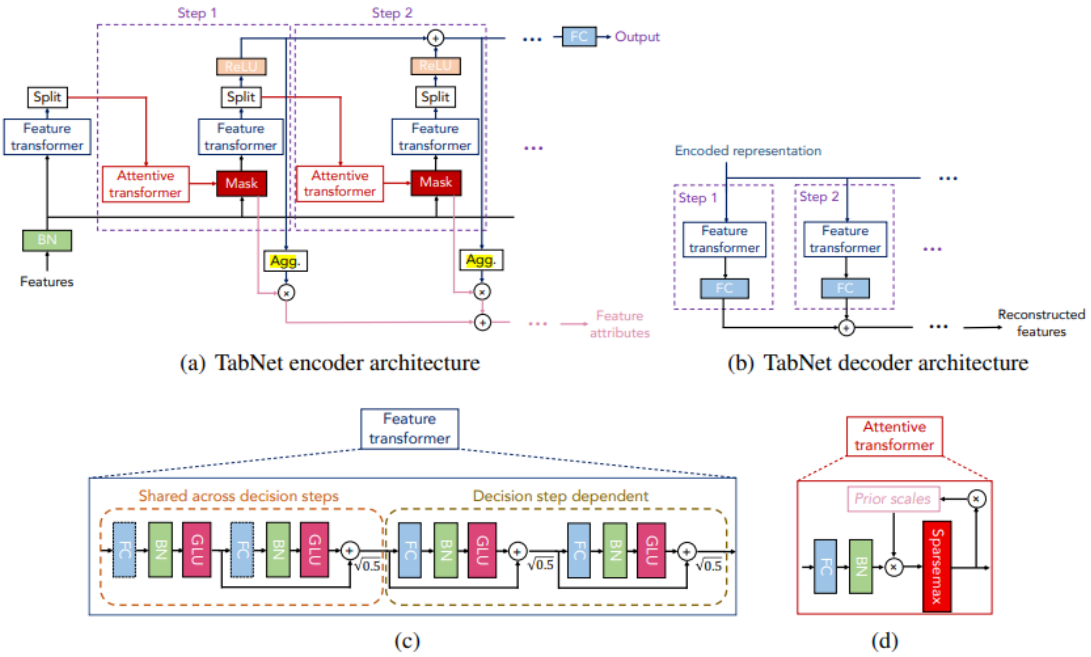

3.4.1. ábra. A döntési faszerű osztályozás illusztrációja hagyományos mélyháló blokkokkal (balra) és az ezzel meghatározott döntési tér (jobbra)[1].

A releváns jellemzők kiválasztása a bemeneteken található úgynevezett ritka maszkokkal való szorzással történik. A kiválasztott jellemzők lineáris transzformációja és egy eltolás hozzáadása után a ReLU a régiók nullázásával régiókiválasztást hajt végre. A 3.4.1 ábrán például csak a jobb felső régió marad nem 0. Látható, hogy a döntéshatárokat az eltolások állítják be. Több régió összesítése összeadáson alapul.

Ilyen kialakításban az egyes jellemzők kiválasztása kulcsfontosságú a döntési határok hipersíkbeli formájának eléréséhez. A TabNet ilyen funkcionalitásra épül,

és a cikk szerzői szerint felülmúlja a döntési fákat, miközben kihasználja azok előnyeit a következők használatával:

- adatból tanult ritka, megfigyelésalapú jellemzőkiválasztást alkalmaz;
- szekvenciális, több lépésből álló architektúrát épít, ahol minden lépés hozzájárul a döntés egy részéhez a kiválasztott jellemzők alapján;
- növeli a tanulási kapacitást a kiválasztott jellemzők nemlineáris feldolgozásával; és
- a magasabb dimenziók és több lépés révén utánozza az együttes módszereket.




3.4.2. ábra. TabNet architektúra[1]

A 3.4.2 ábra (a) része a TabNet architektúrát mutatja be táblázatos adatok kódolásához. Nyers numerikus jellemzőket vagy ismérveket használ, és a kategorikus jellemzők leképezését tanulható beágyazásokkal veszi figyelembe. Nem alkalmaz globális jellemző normalizálást, csupán kötegenkénti normalizálást (batch normalisation - BN), amelyet a következő bekezdésekben elmagyarázok. A TabNet ugyanazokat az $f \in \mathbb{R}^{B \times D}$ D-dimenziós jellemzőket (features) kapja minden döntési lépésben, ahol $B$ a köteg mérete. A TabNet kódolása szekvenciális, több lépésből álló feldolgozáson alapul $N_{steps}$ darab döntési lépéssel. Az $i$-edik lépés

bemenete az $(i-1)$-edik lépés feldolgozott információját használja annak eldöntésére, hogy mely jellemzőket vegye figyelembe. Az $i$-edik lépés kimenete pedig a feldolgozott jellemzők reprezentációját adja válaszul, hogy aztán az összesített döntés része lehessen.

### Kötegelt normalizálás (Batch Normalisation - BN)

A kötegelt normalizálás olyan technika, amelyet a neurális hálózatok tanításának felgyorsítására és stabilizálására használnak. Ezt a módszert Sergey Ioffe és Christian Szegedy vezette be 2015-ben[14]. Lényege, hogy minden minikötegben (mini-batch) normálja a rejtett rétegek aktivációit, azaz a bemeneti adatok eloszlását normalizálja az egyes rétegeken belül. Az eljárásnak számos előnye van, beleértve a tanulási folyamat gyorsítását, a gradienseltűnés (vanishing gradient) és robbanás (exploding gradient) problémáinak csökkentését, valamint a modell általánosítási képességének javítását. A kötegelt normalizálás lépései a következők.

**Bemeneti aktivációk gyűjtése** Az adott réteg bemeneti aktivációi $\mathbf{X} = \{x_1, x_2, \ldots, x_m\}$, ahol $m$ a miniköteg mérete.

**Átlag és variancia kiszámítása** Az aktivációk minibatch átlaga:

$$\mu_{\mathbf{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i.$$

Az aktivációk minibatch varianciája:

$$\sigma_{\mathbf{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathbf{B}})^2.$$

**Normálás** Minden aktiváció normálása a minibatch átlaga és varianciája alapján:

$$\hat{x}_i = \frac{x_i - \mu_{\mathbf{B}}}{\sqrt{\sigma_{\mathbf{B}}^2 + \epsilon}},$$

ahol $\epsilon$ egy kis konstans a numerikus stabilitás érdekében.

**Skálázás és eltolás** A normált aktivációk skálázása és eltolása tanulható $\gamma$ és $\beta$ paraméterekkel:

$$y_i = \gamma\hat{x}_i + \beta.$$

A következő réteg bemenete tehát az így nyert $y_i$ érték lesz.

**Előnyök**

- Gyorsabb konvergencia: A BN csökkenti az aktivációk változékonyságát a rétegek között, ami stabilabb és gyorsabb tanulást eredményez.
- Nagyobb tanulási ráták: Mivel a BN stabilizálja az aktivációkat, nagyobb tanulási ráták alkalmazása válik lehetővé, ami tovább gyorsítja a tanulást.
- Regularizációs hatás: A minibatch-ek változékonysága enyhe regularizációs hatást fejt ki, csökkentve a túltanulás (overfitting) kockázatát.

**Implementáció** A kötegelt normalizációt tipikusan a teljesen összekötött rétegek (fully connected layers - FCs) és a konvolúciós rétegek (convolutional layers) után alkalmazzák. Pythonban a 'keras' könyvtárban a 'BatchNormalization()' objektummal végezhető.

### 3.4.2. Jellemzőkiválasztás

**Definíció.** Lágy maximum (softmax): A softmax függvény bemenete egy $K$ darab valós számból álló $\mathbf{z}$ vektor, amelyeket $K$ darab valószínűségből álló valószínűségi eloszlásba normalizál. Formálisan softmax : $\mathbb{R}^K \to (0;1)^K$, ahol $K \geq 1$. Egy $\mathbf{z} = (z_1; \ldots; z_K) \in \mathbb{R}^K$ input esetén a softmax$(\mathbf{z}) \in (0;1)^K$ output egy komponensét a következőképpen határozzuk meg:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}.$$

**Definíció.** Ritka maximum (sparsemax): Legyen $\Delta^{K-1} := \{\mathbf{p} \in \mathbb{R}^K | \mathbf{1^T p} = 1, \mathbf{p} \geq \mathbf{0}\}$ a $(K-1)$-dimenziós szimplex. Ekkor

$$\text{sparsemax}(\mathbf{z}) = \underset{\mathbf{p} \in \Delta^{K-1}}{\text{argmin}} \, ||\mathbf{p} - \mathbf{z}||^2.$$

Szavakba öntve a sparsemax a $\mathbf{z}$ bemeneti vektor valószínűségi szimplexre vett euklideszi vetületét adja vissza csakúgy, mint a softmax, ám a softmax csak nemnegatív értékeket ad vissza, a sparsemax viszont nem feltétlenül. Ezzel képesek lehetünk ritka eloszlásokat is modellezni, amikor némely eseményhez 0 valószínűséget szeretnénk társítani[20].

A TabNet egy $M[i] \in \mathbb{R}^{B \times D}$ tanulható maszkolási mátrixot használ a releváns jellemzők lágy kiválasztásához. A legjelentősebb jellemzők ritka kiválasztása révén a döntési lépés tanulási kapacitása nem pazarlódik el irreleváns jellemzőkre, így

a modell paramétereiben hatékonyabbá válik. A maszkolás szorzás alapú, azaz $M[i] \cdot f$. A TabNet figyelő transzformert (lásd 3.4.2 ábra (d) része) használ a maszkok előállításához az előző lépés $a[i-1]$ feldolgozott jellemzői alapján:

$$M[i] = \text{sparsemax}(P[i-1] \cdot h_i(a[i-1])).$$

A sparsemax normalizáció itt a szorzatmátrixra soronként értendő. Ez a ritkaság lehetővé tételével az euklideszi projekciót a valószínűségi szimplexre vetíti, ami jobb teljesítményt nyújt és pontosabban igazodik a ritka jellemzőkiválasztáshoz, így képesek leszünk eredményeinket megmagyarázni. Vegyük észre, hogy a sparsemax miatt

$$\sum_{d=1}^{D} M[i]_{b,d} = 1 \quad \forall\, 1 \leq b \leq B.$$

$h_i$ tanulható függvény, amelyet a 3.4.2 ábra (d) részén egy teljesen összekötött réteg (FC) és egy kötegelt normalizáció (BN) követ. A $P[i]$ a priorskála, amely jelzi, hogy egy adott jellemzőt korábban milyen mértékben használtunk:

$$P[i] = \prod_{j=1}^{i} (\gamma - M[j]),$$

ahol $\gamma$ egy lazítási vagy relaxációs paraméter - ha $\gamma = 1^{B \times D}$, akkor egy jellemző csak egy döntési lépésben kerül felhasználásra, és ahogy $\gamma$ nő, úgy több lépésben is felhasználhatóvá válik. $P[0]$ kezdetben csupa egyes, $1^{B \times D}$, azaz nincs előzetes feltételezésünk a maszkolt jellemzőkre vonatkozóan. Ha néhány jellemző nincs használatban - mint ahogy önálló tanulásnál (self-supervised learning), amelyről egy későbbi alfejezet szól -, a megfelelő $P[0]$ elemeket nullára állítjuk, hogy segítsük a modell tanulását. A kiválasztott jellemzők ritkaságának további szabályozására a szerzők az entrópia formájában történő ritkaság-regularizációt javasolják Grandvalet és Bengio 2004-es tanulmánya alapján[9]:

$$L_{\text{sparse}} = \sum_{i=1}^{N_{steps}} \sum_{b=1}^{B} \sum_{d=1}^{D} \frac{-M_{b,d}[i] \log(M_{b,d}[i] + \epsilon)}{N_{steps} \cdot B},$$

ahol $\epsilon$ egy kis szám a numerikus stabilitás érdekében. A ritkaság-regularizációs tagot hozzáadják az összes veszteséghez $\lambda_{\text{sparse}}$ együtthatóval. A ritkaság kedvező következtető elfogultságot (inductive bias) biztosít olyan adatbázisok esetén, ahol a legtöbb jellemző redundáns.

### 3.4.3. Jellemzők feldolgozása

A maszkkal szűrt jellemzőket egy jellemzőtranszformátor segítségével dolgozzuk fel (lásd a 3.4.2 ábra (c) részét), majd felosztjuk a döntési lépés kimenetére és a következő lépés bemeneti információira:

$$[d[i], a[i]] = f_i(M[i] \cdot f),$$

ahol $d[i] \in \mathbb{R}^{B \times N_d}$ a döntési lépés kimenete és $a[i] \in \mathbb{R}^{B \times N_a}$ a következő lépés bemeneti információja, valamint $f_i$ a tanulható jellemzőtranszformátor függvény. A paraméterhatékony és nagy kapacitású robusztus tanulás érdekében a jellemző transzformátornak olyan rétegeket kell tartalmaznia, amelyek megosztottak minden döntési lépés között (mivel ugyanazok a jellemzők kerülnek bemenetre különböző döntési lépésekben), valamint a döntési lépésektől függő rétegeket is. A 3.4.2 ábra (c) része mutatja a megvalósítást két megosztott réteg és két döntési lépéstől függő réteg összefűzéseként. Minden teljesen összekötött (FC) réteget kötegelt normalizálás (BN), majd aktivációs függvényként GLU (gated linear unit) nemlinearitás követ[5]. Végül az így kapott kimenet egy normált reziduális kapcsolattal csatlakozik a következő réteghez. A $\sqrt{0,5}$-tel való normalizálás segít stabilizálni a tanulást azáltal, hogy biztosítja, hogy a hálózatban a variancia nem változik drasztikusan[7]. Független minta esetén a mögöttes heurisztika az, hogy magasdimenziós térben nagyobb valószínűséggel választunk két merőleges változót, amelyek szórása a kötegelt normalizálás miatt nagyjából egységnyi, így a $\sqrt{0,5}$-ös korrekcióval az összeg szórása szintén közelítőleg egységnyi marad.

A gyorsabb tanulás érdekében a TabNet nagy kötegméreteket használ a kötegelt normalizálásnál. Ezért, kivéve az input jellemzőkre alkalmazott BN-t, ghost BN formát használ[13], virtuális $B_V$ batch méret és $m_B$ átlag alkalmazásával. Az input jellemzők esetében az alacsony varianciájú átlagolás előnyei miatt a TabNet kerüli a ghost BN-t. Végül a 3.4.1 ábrán bemutatott döntési fa-szerű aggregáció inspirációjára az összesített döntési beágyazást a

$$d_{out} = \sum_{i=1}^{N_{steps}} \mathrm{ReLU}(d[i])$$

formában hozza létre. Erre egy $W_{final} d_{out}$ lineáris leképezést alkalmaz, hogy megkapja a kimeneti leképezést. A diszkrét kimenetekhez emellett softmaxot is alkalmaz a tanítás és argmaxot a következtetés során.

### 3.4.4. Értelmezhetőség

A TabNet jellemzőkiválasztási maszkjai megmutathatják az egyes lépésekben kiválasztott jellemzőket. Ha $M_{b,d}[i] = 0$, akkor a $b$ minta $d$ jellemzője nem járul hozzá a döntéshez. Ha $f_i$ egy lineáris függvény lenne, akkor az $M_{b,d}[i]$ együttható az $f_{b,d}$ jellemző fontosságának felelne meg. Bár minden döntési lépés nemlineáris feldolgozást alkalmaz a GLU aktivációs függvény miatt, a kimeneteiket később lineáris módon kombinálják. A cél az, hogy egy aggregált jellemzőfontosság is számszerűsítve legyen az egyes lépések elemzése mellett. A különböző lépések maszkjainak kombinálása egy együtthatót igényel, amely képes súlyozni az egyes lépések relatív fontosságát a döntésben. Erre a TabNet egyszerű megoldása, hogy $\eta_b[i] = \sum_{c=1}^{N_d} \mathrm{ReLU}(d_{b,c}[i])$ jelezze az $i$-edik döntési lépés aggregált döntési hozzájárulását a $b$-edik mintára. Ez a mérőszám összhangban van az intuícióval, miszerint ha tetszőleges $1 \leq b \leq B$ és $1 \leq c \leq N_d$ számokra $d_{b,c}[i] < 0$, akkor az $i$-edik döntési lépés összes jellemzőjének 0 hozzájárulása van az összesített döntéshez. Továbbá nem csak az intuíciónak, hanem a tényleges hozzájárulásnak is megfelel, hiszen a $d_{out}$ összesített döntés meghatározásakor is ReLU az aktivációs függvény, így a negatív elemek nem fogják befolyásolni a TabNet döntését, amelyen a végső $W_{final}$ lineáris leképezés sem változtat. Ahogy a $d_{b,c}[i]$ érték növekszik, úgy egyre nagyobb szerepet játszik az összesített lineáris kombinációban. Mindezek után az összesített ismérvfontossági maszkot úgy kaphatjuk, hogy a döntési maszkot skálázzuk minden döntési lépésnél $\eta_b[i]$-vel, vagyis a $b$-edik minta $d$-edik jellemzőjének az összesített fontossága:

$$M_{\mathrm{agg}-b,d} = \frac{\sum_{i=1}^{Nsteps} \eta_b[i] M_{b,d}[i]}{\sum_{j=1}^{D} \sum_{i=1}^{N_steps} \eta_b[i] M_{b,j}[i]}.$$

A nevezőben lévő normáló tényezőre azért van szükség, hogy $\sum_{d=1}^{D} M_{\mathrm{agg}-b,d} = 1$ legyen, azaz egy ismérv esetén a jellemzők fontossága 1-re összegződjön.

### 3.4.5. Önálló tanulás

#### Az önálló tanulásról

Az önálló tanulás (Self-Supervised Learning - SSL) olyan gépi tanulási technika, amely az adatok belső struktúrájából származó, felügyelet nélküli jeleket használ fel a modellek tanítására. Ez a módszer hasznos, mert lehetővé teszi a modellek számára, hogy előzetes ismereteket szerezzenek nagy mennyiségű címkézetlen adaton, mielőtt specifikus feladatokra finomhangolnák őket felügyelt tanulással. Az

önálló tanulás egyik legnagyobb előnye, hogy kevesebb címkézett adatra van szükség a modell hatékony tanításához, valamint új, ismeretlen adatokon is jobban általánosítanak. Az önálló tanulás működése a következő pontokba szedhető[30].

**Feladat definiálása** Az önálló tanulás keretében a rendszer először definiál egy mesterséges feladatot, amelyet az adatból származó címkék nélkül tud megoldani. Például képeknél az egyik feladat lehet a kép egy részének kitöltése vagy a szomszédos képkockák előrejelzése egy videóban.

**Képzési adatok generálása** Az adatokból generálnak olyan címkéket, amelyek segítenek a mesterséges feladat megoldásában. Például egy képnek eltávolíthatják egy részét, és a feladat az lesz, hogy a modell megtanulja rekonstruálni az eltávolított részt.

**Modell tanítása** A modellt a mesterséges feladat megoldására tanítják. Ezáltal a modell megtanulja az adat belső struktúráját és összefüggéseit.

**Finomhangolás (fine-tuning)** Miután a modell elsajátította az adat belső struktúráját, felügyelt tanulással finomhangolják specifikus feladatokra, mint például osztályozás vagy előrejelzés.

### Példák önálló tanulási feladatokra

- Képkitöltés (Image Inpainting): A kép egy részét eltávolítják, és a modell megtanulja kitölteni az üres részt.
- Jelzaj-elválasztás (Noise-Contrastive Estimation): A modell megtanulja megkülönböztetni a zajos és a nem zajos adatokat.
- Kódolás-dekódolás (Autoencoding): A modell megtanulja az adatot egy kisebb dimenziós reprezentációba kódolni és onnan visszafejteni.

A TabNet ez utóbbit használja előtanításhoz (pre-training), amelyet a következőkben részletezek.

### Önálló táblázatos tanulás TabNettel

A mesterséges neurális hálózat része egy dekóder architektúra, amely a TabNet által kódolt reprezentációkból rekonstruálja a táblázatba foglalt jellemzőket. A dekóder ismérvtransformerekből áll, amelyeket minden döntési lépésnél egy teljesen

összekötött réteg követ, ahogy azt a 3.4.2 ábra (b) része mutatja. Az eredmények összeadódnak, hogy megkapjuk a rekonstruált jellemzőket. A TabNet mesterséges feladata a hiányzó jellemző oszlopok előrejelzése a többi oszlop alapján.

Vegyünk egy bináris maszkot $S \in \{0,1\}^{B\times D}$. Ekkor $(1-S)\cdot\hat{f}$ az enkóder bemenetei, és az $S\cdot\hat{f}$ rekonstruált jellemzők a dekóder kimenetei. $P[0]$-t $(1-S)$-ként inicializáljuk az enkóderben, hogy a modell csak az ismert jellemzőkre koncentráljon, és a dekóder utolsó FC rétegét megszorozzuk $S$-sel, hogy az ismeretlen jellemzőket kapjuk meg. Az önálló tanulási fázisban a veszteségfüggvény az átlagos rekonstrukciós abszolút eltérésként értelmezhető:

$$L_{SSL}(\hat{f};f)=\frac{1}{B\cdot D}\sum_{b=1}^{B}\sum_{d=1}^{D}\left|\frac{(\hat{f}_{b,d}-f_{b,d})\cdot S_{b,d}}{\sqrt{\frac{1}{B}\sum_{b=1}^{B}(f_{b,d}-\frac{1}{B}\sum_{b=1}^{B}f_{b,d})^2}}\right|.$$

A ground truth, azaz a valós értékek szórásával való normalizálás előnyös, mivel a jellemzők eltérő nagyságrendekkel rendelkezhetnek. Minden iterációnál egy $p_s$ paraméterű Bernoulli-eloszlásból függetlenül mintavételezzük az $S_{b,d}$ értékeket.

### 3.4.6. Modellimplementáció

A TabNet használatához Pythonban a PyTorch könyvtárra volt szükségem. Ez a Torch könyvtáron alapuló gépi és mélytanulási könyvtár, amelyet akár olyan feladatokhoz is használnak, mint a számítógépes látás és a természetes nyelvi feldolgozás. Innen installáltam magát a 'TabNetRegressor' modellt illetve az előtanításához szükséges 'TabNetPretrainer' modellt is.

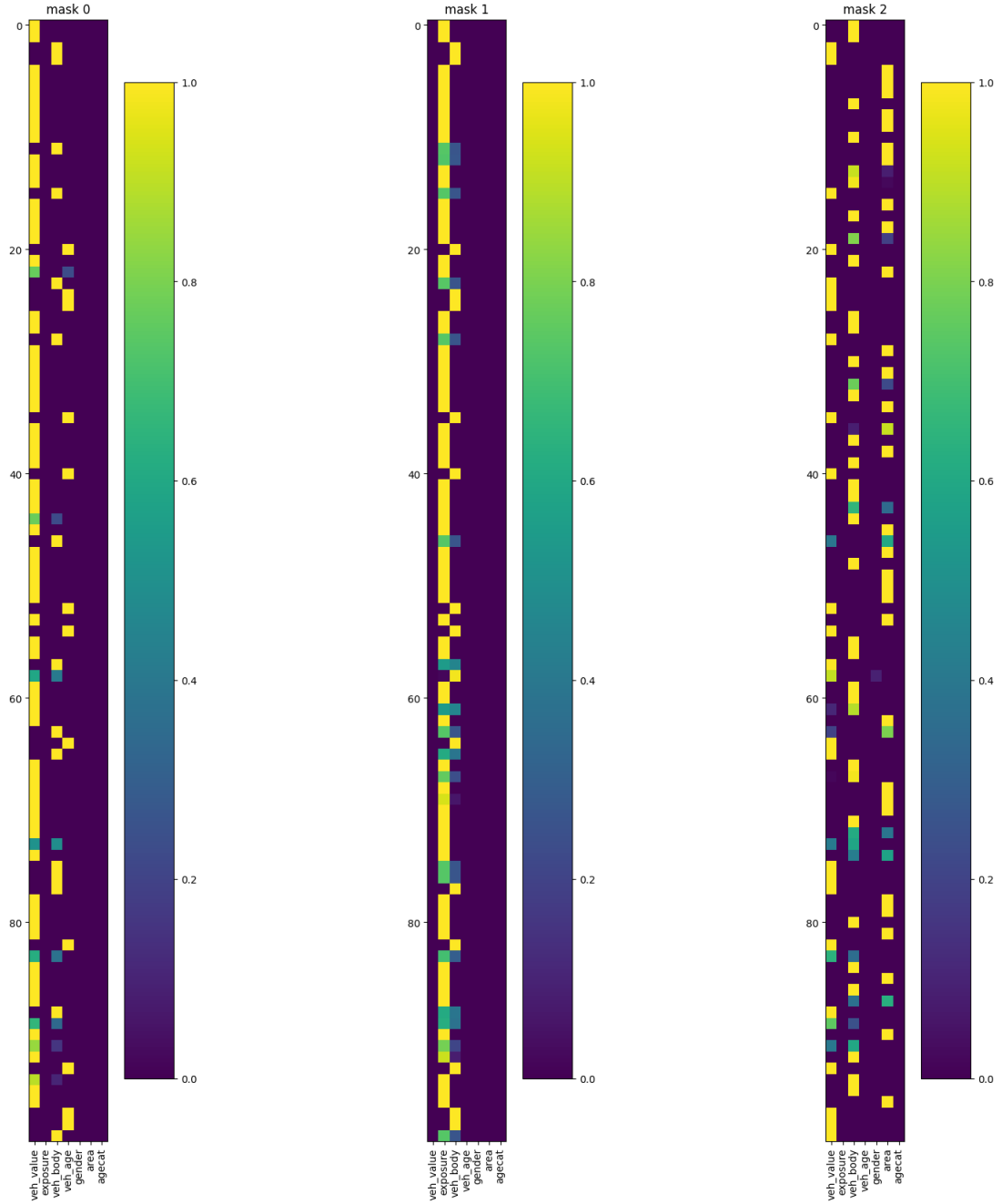
### 3.4.7. Adatelőkészítés

A TabNet ugyan képes kezelni a nyers adatokat, mégis több adatelőkészítő lépést eszközöltem. Egyrészt, bár a kategorikus változókat is tudja használni, elvárja a hálózat, hogy paraméterként állítsuk be, melyek ezek. Meg kell adni ezen ismérvek indexét (hányadik oszlopban vannak a táblázatban), dimenzióját (hány kategóriaszint van) és hogy hány dimenziós térbe akarjuk beágyazni. Mivel a kategorikus változók dimenziója eredendően 'cat_dims' = [13, 4, 2, 6, 6], ezért a 'cat_emb_dim' = [5, 3, 2, 3, 3] beágyazást választottam. Másrészt az eredeti cikk hangsúyozza, hogy a TabNetnek az adatok normalizálására sincs szüksége globálisan, mert kötegelt normalizálást használ. Én a gyorsabb konvergencia reményében sztenderdizáltam a numerikus változókat.

### 3.4.8. Kárszámbecslés

Az előtanítás során sikerült jobban teljesítenie a modellnek, mintha csupán az átlaggal becsülnék, mivel a normált célfüggvény 1-nél kisebb érték alá ment. A legjobb értékként 0,6322-t sikerült elérnem azzal, hogy a figyelő és az ismérvtranszformer dimenzióját 'n_a' = 'n_d' = 6-ra állítottam. A többi paramétert hiába módosítottam az alapbeállításhoz képest. Ugyanezekkel a beállításokkal a legjobb elért eredményem kárszámbecslésre 1,0253, ami jócskán gyengébb teljesítmény, mint a GLM vagy az XGBoost általi. Ehhez a szintén alapbeállításként használt veszteségfüggvényt, az átlagos négyzetes hibát használtam. Próbálkoztam itt is Poisson-regresszióval a PyTorch-ban implementált Poisson-veszteségfüggvénnyel, ám ekkor még negatív eredményt is predikált a modell, ami semmivel sem magyarázható.

A négyzetes veszteségfüggvény esetén a következő maszkok rajzolódtak ki sorra a döntési lépésekkor:




3.4.3. ábra. TabNet maszkjai kárszámbecslésnél az első 100 megfigyelésre

Ezek alapján, és a cikktől eltérően implementált globális ismérvfontosság alapján - aggregált maszk helyett 1-re összegződő fontossági súlyok rendre a változók szerint: [0,2766; 0,1946; 0,3824; 0,0361; 0,0014; 0,1085; 0,0005] - is látszik, hogy a TabNet leginkább a gépjárművek vázszerkezetét, majd az árát és a kitettséget tartotta fontosnak döntéshozáskor.

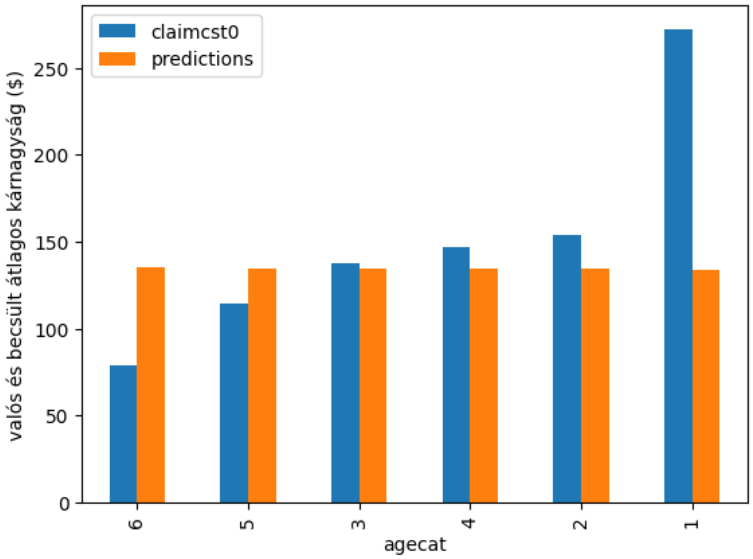
### 3.4.9. Kárnagyságbecslés

Az előtanítás most is sikerrel járt, a TabNet tanult, mert a veszteségfüggvény értéke 1 alatti, így jobban becsül, mintha csak az átlagot használná.

A regressziókor a gradienssereszkedésre használtam egy skálázót, amely 30 lépés után a tanulási rátát tizedére csökkenti, mert a veszteségfüggvény alakulásán az látszott, hogy ott elkezd erősen ingadozni. A kisebb lépések az optimum közelében segítik a konvergenciát.

Ezzel a változtatással 0,9089 értékű lett a validáló adathalmazon a becsült-valós összárnagyság aránya. Ez sajnos a nullmodellt sem szárnyalja túl. Ha mindent az átlaggal becslünk, akkor ugyanez az arány 93% felett van.

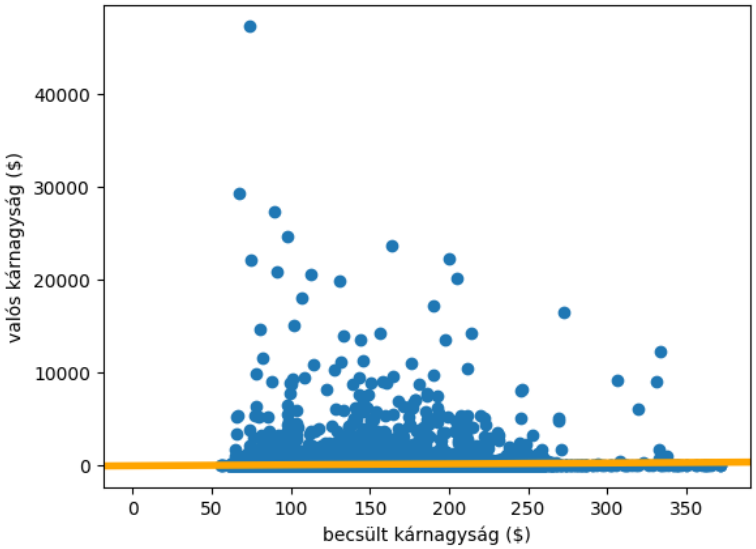
Tehát a TabNet ebben a feladatban nehezen tanult. Minden változó szerint az átlagos kárt próbálta megtanulni, például életkor-kategóriák szerinti bontásban:




3.4.4. ábra. Tényleges és becsült átlagos kárnagyság életkor-kategóriák szerint

Ennek a problémának a gyökere a veszteségfüggvényre vezethető vissza. Erre a következő fejezetben kitérek.

A validáló adathalmazon a predikciók függvényében a valódi értékek ábrája is arra utal, hogy nem sikerült a tanulás. A narancssárga $x = y$ egyenletű egyenes mentén szeretnénk látni a pontokat.




3.4.5. ábra. Valós kárnagyságok a becsült értékek függvényében

# 4. fejezet

# Az eredmények értelmezése és jövőbeli célok

A neurális hálózatok gyenge teljesítményének egyik oka lehet, hogy kevés magyarázóváltozó áll rendelkezésre, és ezek közül is a legtöbb kategorikus. Emiatt az adatmátrix nagyon ritka, amelyet nehezen tud megtanulni a hálózat[1]. Ahogy többször hangsúlyoztam, ezek a modellek magasdimenziójú adatbázisok esetén múltak felül más módszereket.

Érdemes lehet azonban a jövőben több időt szentelni a hiperparaméter optimalizációnak. Erről McDonnell és társai[23] is azt nyilatkozták, hogy rengeteg időt vett igénybe és nehézségeket okozott. Ezen kívül a kárnagyság becslését esetlegesen pontosítani lehetne egy saját veszteségfüggvénnyel Tweedie-regresszióhoz. Továbbá megfontolandó tesztelni a TabNet működését más adathalmazokon, amelyeken eredményesebben teljesíthet.

A saját építésű mesterséges neurális hálózatok optimalizálása sem elvetendő, hiszen bár gyengébb teljesítménnyel bírtak, mint a GLM és az XGBoost, figyelemreméltó teljesítményt értek el a TabNet mellett.

Biztosítási kockázatárazáshoz végső soron az általánosított lineáris modellt használnám legszívesebben, mivel könnyedén illeszthető az adatokra, és eredményeinek értelmezése nem jelent problémát.

# Irodalomjegyzék


[1] Arik, S. Ö., & Pfister, T. (2021). TabNet: Attentive Interpretable Tabular Learning. *Proceedings of the AAAI Conference on Artificial Intelligence, 35*(8), 6679–6687.

[2] Ayuso, M., Guillen, M., & Nielsen, J. P. (2019). Improving automobile insurance ratemaking using telematics: Incorporating mileage and driver behaviour data. *Transportation, 46*(3), 735–752. `https://doi.org/10.1007/s11116-018-9890-7`

[3] Baecke, P., & Bocca, L. (2017). The value of vehicle telematics data in insurance risk selection processes. *Decision Support Systems, 98*, 69–79. `https://doi.org/10.1016/j.dss.2017.04.009`

[4] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

[5] Dauphin, Y. N., Fan, A., Auli, M., & Gran- gier, D. (2016). Language Modeling with Gated Convolutional Networks. arXiv preprint arXiv:1612.08083.

[6] Dunn, P. K., & Smyth, G. K. (2005). Series evaluation of Tweedie exponential dispersion model densities. *Statistics and Computing, 15,* 267-280.

[7] Gehring, J., Auli, M., Grangier, D., & Dauphin, Y. N. (2017). Convolutional Sequence to Sequence Learning. arXiv preprint arXiv:1705.03122.

[8] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

[9] Grandvalet, Y., & Bengio, Y. 2004. Semi-supervised Learning by Entropy Minimization. In *NIPS*.

[10] Guelman, L. (2012). Gradient boosting trees for auto insurance loss cost modeling and prediction. *Expert Systems with Applications, 39*(3), 3659–3667. https://doi.org/10.1016/j.eswa.2011.09.058

[11] Henckaerts, R., Antonio, K., Clijsters, M., & Verbelen, R. (2018). A data driven binning strategy for the construction of insurance tariff classes. *Scandinavian Actuarial Journal, 2018*(8), 681–705. https://doi.org/10.1080/03461238.2018.1429300

[12] Henckaerts, R., Côté, M.-P., Antonio, K., & Verbelen, R. (2021). Boosting Insights in Insurance Tariff Plans with Tree-Based Machine Learning Methods. *North American Actuarial Journal, 25*(2), 255–285. https://doi.org/10.1080/10920277.2020.1745656

[13] Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv preprint arXiv:1705.08741.

[14] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167 https://arxiv.org/abs/1502.03167

[15] Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

[16] Klein, N., Denuit, M., Lang, S., & Kneib, T. (2014). Nonlife ratemaking and risk management with Bayesian generalized additive models for location, scale, and shape. *Insurance: Mathematics and Economics, 55*, 225–249. https://doi.org/10.1016/j.insmatheco.2014.02.001

[17] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444. https://doi.org/10.1038/nature14539

[18] Ma, Y.-L., Zhu, X., Hu, X., & Chiu, Y.-C. (2018). The use of context-sensitive insurance telematics data in auto insurance rate making. *Transportation Research Part A: Policy and Practice, 113,* 243–258. https://doi.org/10.1016/j.tra.2018.04.013

[19] Maillart, A. (2021). Toward an explainable machine learning model for claim frequency: A use case in car insurance pricing with telematics data. *European Actuarial Journal.* https://doi.org/10.1007/s13385-021-00270-5

[20] Martins, A. F. T.; and Astudillo, R. F. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. arXiv:1602.02068

[21] Masello, L., Sheehan, B., Murphy, F., Castignani, G., McDonnell, K., & Ryan, C. (2022). From Traditional to Autonomous Vehicles: A Systematic Review of Data Availability. *Transportation Research Record, 2676*(4), 161–193. https://doi.org/10.1177/03611981211057532

[22] McCullagh, P., & Nelder, J. A. (2019). In *Generalized Linear Models* (2nd ed.). Routledge. https://doi.org/10.1201/9780203753736.

[23] McDonnell, K., Murphy, F., Sheehan, B., Masello, L. Castignani, G. (2023). Deep learning in insurance: Accuracy and model interpretability using TabNet *Expert Systems with Applications* 217, 119543

[24] Molnar, C. (2020). *Interpretable Machine Learning.*

[25] Nelder, J. & Wedderburn, R. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society.*

[26] Noll, A., Salzmann, R., & Wuthrich, M. V. (2018). Case Study: French Motor Third-Party Liability Claims. *SSRN Electronic Journal.* https://doi.org/10.2139/ssrn.3164764

[27] Paefgen, J., Staake, T., & Thiesse, F. (2013). Evaluation and aggregation of pay-as-you-drive insurance rate factors: A classification analysis approach. *Decision Support Systems, 56,* 192–201. https://doi.org/10.1016/j.dss.2013.06.001

[28] Pesantez-Narvaez, J., Guillen, M., & Alcaniz, M. (2019). Predicting Motor Insurance Claims Using Telematics Data—XGBoost versus Logistic Regression. *Risks, 7*(2), 70. https://doi.org/10.3390/risks7020070

[29] Quan, Z., & Valdez, E. A. (2018). Predictive analytics of insurance claims using multivariate decision trees. *Dependence Modeling, 6*(1), 377–407. https://doi.org/10.1515/demo-2018-0022

[30] Ren, X., Wei, W., Xia, L., & Huang, C. (2024). A Comprehensive Survey on Self-Supervised Learning for Recommendation. arXiv preprint arXiv:2404.03354.

[31] Siami, M., Naderpour, M., & Lu, J. (2021). A Mobile Telematics Pattern Recognition Framework for Driving Behavior Extraction. *IEEE Transactions on Intelligent Transportation Systems, 22*(3), 1459–1472. https://doi.org/10.1109/TITS.2020.2971214

[32] Verbelen, R., Antonio, K., & Claeskens, G. (2018). Unravelling the predictive power of telematics data in car insurance pricing. *Journal of the Royal Statistical Society. Series C: Applied Statistics, 67*(5), 1275–1304. Scopus. doi:10.1111/rssc.12283.

[33] Wüthrich, M. V. (2020). Bias regularization in neural network models for general insurance pricing. *European Actuarial Journal, 10*(1), 179–202. https://doi.org/10.1007/s13385-019-00215-z