|  |  |
| --- | --- |
| Miskolci Egyetem  Gépészmérnöki és Informatikai Kar  Általános Informatikai Intézeti  *3515 Miskolc-Egyetemváros* |  |

**Diplomamunka**

*Feladat címe:*

Mesterséges intelligencia alapú oktatást támogató alkalmazás tervezése

*Készítette:*

Szilvási Péter

*MSc szintű mérnökinformatika szakos*

*Alkalmazásfejlesztő szakirányos hallgató*

*Témavezető:*

Dr. Kovács László

*tanszékvezető, egyetemi tanár*

**Miskolc, 2020.**

# Bevezetés

## Téma választás ismertetése

placeholder

## Miért kell támogatni az oktatást?

placeholder

## Oktatás fontossága

placeholder

## Mesterséges intelligencia és az oktatás

placeholder

# Gépi tanulás menete

**Gépi tanulás**

Ha elmélyülünk a gépi tanulás definíciójában, akkor azt találjuk, hogy „*A gépi tanulás az algoritmusok és statisztikai modellek tudományos vizsgálata, amelyet a számítógépes rendszerek egy adott feladat végrehajtásához használnak kifejezett utasítások nélkül, mintákra és következtetésekre támaszkodva*”. [1]

Ha elemezzük az előző definíciót, találunk néhány kiemelt szempontot. Hatalmas tudományos kutatás és erőfeszítés van a növekedés és fejlődés támogatására.

***Algoritmusokra*** és statisztikai modellekre támaszkodik, amelyek azt mondják nekünk, hogy ha el akarjuk sajátítani a gépi tanulást, akkor az algoritmusok, a statisztikák és a valószínűségek átfogó megértését kell kialakítanunk. Célja egy meghatározott feladat végrehajtása. Tehát a gépi tanulási megoldásnak van egy bizonyos hatóköre, például előrejelzés, ajánlás vagy osztályozás.

Kifejezett utasítások nélkül, ami azt mondja nekünk, hogy a gépi tanulási megoldást nem szabad pontosan beprogramozni a dolgok megtanítására. ***Mintákra*** és következtetésekre támaszkodik. Tehát a gépi tanulási megoldásunk a történeti adatokból bizonyos mintákra támaszkodik a helyes működés megtanulásához.

**Megoldható problémák gépi tanulással** [2]

Most tárgyaljuk meg a gépi tanulás néhány érdekes, valós alkalmazását a mindennapokban.

* A ***képfelismerés***. Hogyan tudja a mobiltelefon feloldani magát azzal, hogy ha a gazdája megmutatja az arcát a kamerának, miközben nem oldja fel magát más emberek számára? Egy képfelismerési technológiával érik el ezt a funkcionalitást. Ahhoz, hogy fel tudjuk ismerni a kívánt objektumot, a képi adatok (pixeleket) egy gépi tanulási algoritmusra kell kiterjeszteni.
* A ***rosszindulatú programok észlelése***, ahol arra vagyunk kíváncsik, hogy az antivírusok hogyan tudják hatékonyan megismerni az új vírusokat, mielőtt frissítéseket kapnának. Ismét a gépi tanulás a motor, amely ezt működteti.
* A ***hangfelismerés***. Egy másik jól ismert példa, amikor hang alapján azonosítani tudunk egy személyt. Legyen szó mobiltelefonról vagy okos otthonról, a gépi tanulás lehetővé teszi a felhasználó hatékony azonosítását.

A gépi tanulás terén szerzett kompetenciák és készségek fejlesztésével fantasztikus megoldásokat tudunk kidolgozni, amelyek a valós problémákkal foglalkoznak és értéket képviselnek az emberiség számára.

Végül érdemes megvitatni, hogy a gépi tanulás miért vált manapság ilyen vonzóvá. Nem is gondolná az ember, hogy már a 60-as évek óta létezik. Az elmúlt években azonban volt néhány mozgatóerő.

**Adat túlcsordulás**

Először is, a problémák nagy méretűvé váltak: ***nagy dimenziószám***. A sok dimenzióval rendelkező adat, amelynek túl sok jellemzője van, ami szinte lehetetlenné teszi a hagyományos programozással történő elemzést. A nagy dimenziós adatra példa a páciens egészségi állapota, ahol ***számos jellemző***, például az immunrendszer állapota, a genetikai háttér, a táplálkozás, az operációk, a drog- és dohányfogyasztás.

Hatalmas ***adat túlcsordulás*** van, például hírcsatornák, webszolgáltatások, adatbázisok, e-mailek, felmérések adatforrásain. A vállalkozások a közelmúltban észrevették, hogy hatékony módszert kell találni ezen hatalmas adatok megértésére, amelyekből értéket kell teremteni.

A ***számítástechnika fejlődése***, valamint a számítási és tárolási erőforrások folyamatos csökkenése, például a felhőalapú számítástechnikában hatékonyabbá tette a gépi tanulási algoritmusok megvalósítását. Általában a gépi tanulási algoritmusok erőforrás-igényesek. A kutatások jelentősen megnövekedtek az elmúlt évtizedben. A vállalkozások elkezdték megérteni ennek értékét, ezért számos algoritmust kutattak és fejlesztettek ki.



**1. ábra: A gépi tanulás mozgatórugói** [3]

Tehát, figyelembe véve korábbi információkat, magabiztosan kijelenthetjük, hogy:

*„Az adat az új üzemanyag.”* [4]

## Miben különbözik a gépi tanulás?

Nos, egy jó kérdés merülhet fel bennünk, hogy miért nem programozzuk be üzleti szabályainkat és tartomány (domain) ismeretünket a gépi tanulási modelljeinkbe úgy, mint a hagyományos programozásnál? Más szóval, mi a tényleges különbség a gépi tanulás és a hagyományos programozás között?

**Hagyományos programozás és gépi tanulás** [5]

A hagyományos programozás során a teljes tartományú üzleti logikát olyan ***üzleti szabályok*** formájában rögzítjük, amelyek kifejezetten programozhatók kód szinten. Például, ha egy adott vállalat fizetését szeretnénk kiszámítani, akkor minden forgatókönyvre világos üzleti szabályunk lesz, és pontosan tudjuk, hogy mi fog történi. ***Teljes mértékben megértjük*** az üzleti területet.

Míg a gépi tanulás során nem teljesen értjük a tartományt, valamennyire tudjuk, hogyan reagál a rendszer bizonyos bemenetekre és kimenetekre. **Nem** igazán ***értjük teljesen*** a mögöttes bonyolult kapcsolatokat a bemenet és a kimenet között. Például egy gépi tanulási algoritmus a vásárlási viselkedés hasonlósága alapján különböző csoportokba sorolhatja az ügyfeleket, annak ellenére, hogy nem értjük teljesen az alapul szolgáló komplex összefüggést.

A hagyományos programozásban a rendszer kidolgozott. A ***változtatások***at kifejezetten a forráskód vagy a konfiguráció megváltoztatásával kell végrehajtani. Például, ha új szabályt szeretnénk hozzáadni a fizetési rendszerünkhöz, módosítanunk kell a forráskódot. De a gépi tanulásban a rendszer ***dinamikus*** és folyamatosan tanul. A bemenetek és az új adatok alapján igazodik. Az ügyfelek ápolására szolgáló algoritmus idővel jobbá válhat, mivel több adat trendet tanul meg.

Egy másik különbség az, hogy a hagyományos programozásnál nincs szükségünk semmilyen **történelmi adat**ra vagy tanulási fázisra. A logika pontosan tudja, mit kell tennie. Bérszámítási rendszerünk megfelelően működhet anélkül, hogy tanító fizetési adatokat biztosítanánk. Míg a gépi tanulás során sok történelmi adatra van szükségünk ahhoz, hogy tanulhassunk belőle. Például egy ügyfél megfelelő osztályozásához, a gépi tanulási algoritmusnak elegendő mintát kell látnia a kapcsolatok megértéséhez. Az embernél is hasonló a tanulás, múltbeli eseményekből vonnunk le következtetéseket, majd próbáljuk megjósolni a jövőt.

## Keretrendszerek

A hagyományos programozásban a logika egyszerű és egyértelmű. Könnyen elolvashatjuk a forráskódot, hogy megtaláljuk az igazságot. A gépi tanulás homályos és bonyolult bizonyos matematikai algoritmusok alapján. Általában könyvtárak vagy olyan keretrendszerek, például a scikit-learn, tensorflow és a Keras segítségével elvonatkoztathatunk a gépi tanulás mögötti bonyolultságtól. A keretrendszerek ***absztrakciós*** ***rétegek***, amelyek hozzájárulnak a gyors és egyszerű fejlesztéshez.

**Tensorflow** [6]

A TensorFlow egy erőteljes könyvtár a numerikus számításhoz, különösen jól illeszkedik és a nagyszabású gépi tanuláshoz. A Google Brain csapata fejlesztette ki. Számtalan projekt használja a TensorFlow-t mindenféle gépi tanulási feladathoz, például képosztályozáshoz, természetes nyelv feldolgozásához (NLP), ajánló rendszerekhez, idősoros előrejelzésekhez és még sok máshoz.

Tehát mit kínál a TensorFlow valójában?

* Magja nagyon hasonlít a NumPy-hez, de ***GPU támogatással*** rendelkezik.
* Támogatja az elosztott számítást (több eszközön és szerveren is).
* Egyfajta just-in-time (***JIT***) fordítót tartalmaz, amely lehetővé teszi a számítások optimalizálását a sebesség és a memóriahasználat szempontjából. Úgy működik, hogy a Python függvényből kivonja a számítási grafikont, majd optimalizálja azt.
* A TensorFlow sokkal több olyan funkciót kínál, amelyek ezekre az alapvető jellemzőkre épülnek: a legfontosabb természetesen a Keras, de rendelkezik ***adatbetöltési***, ***előfeldolgozási***, ***képfeldolgozási*** operációkkal.

A legalacsonyabb szinten az egyes TensorFlow műveletek rendkívül hatékony C++ kód felhasználásával valósulnak meg. Számos műveletnek több megvalósítása is van, úgynevezett ***kernelek***: mindegyik kernel egy adott eszköztípusra van címezve, például ***CPU***, ***GPU*** vagy ***TPU*** (Tensor Processing Units).



**Keras** [7]

A ***Keras*** egy mély tanulási ***API*** a TensorFlow gépi tanulási platform tetején fut. Azért fejlesztették, hogy gyors kísérletezéseket érjünk el. Az ötletből eredményeket érjünk el, amilyen gyorsan csak lehetséges. Számos mély tanulási keretrendszer áll rendelkezésre ma. Miért érdemes a Keras-t használni, mint bármely másikat?

* A Keras egy API, amelyet emberek, nem pedig gépek számára terveztek. A Keras a kognitív terhelés csökkentésének bevált módszereit követi: következetes és ***egyszerű*** API-kat kínál, minimalizálja a gyakori használati esetekhez szükséges felhasználói műveletek számát, és ***egyértelmű*** és cselekvőképes ***visszajelzést*** ad a felhasználói hibákról.
* Ez megkönnyíti a Keras megtanulását és használatát. Keras felhasználóként produktívabb vagy, lehetővé téve, hogy több ötletet kipróbálhassunk gyorsabban.
* Ez az egyszerű használat nem jár csökkentet rugalmassággal: mivel a Keras mélyen integrálódik az alacsony szintű TensorFlow funkcionalitásokhoz. Lehetséges olyan feltörhető munkafolyamatok kikifejleszteni, ahol a funkcionalitás bármely része ***testre szabható***.

## Főbb tanulási típusok

Fontos tisztában lenni a gépi tanulási algoritmusok különböző típusaival. Minden algoritmuscsoportnak meg van a maga módszere a tanulásra. Alapvetően három csoportot különíthetünk el: a ***felügyelt*** tanulási, a ***felügyelet nélküli*** és a ***megerősítő*** tanulási algoritmusok. Nézzük meg, hogyan működik mindegyikük!

**Felügyelt tanulás** [8]

Kezdjük a felügyelt tanulással. A felügyelt tanulási algoritmusok úgy működnek, hogy egy ***adott bemenetre*** () egy ***megfelelő kimenetet*** () eredményeznek. A bemeneti érték egy függvényen () keresztül leképezésre, ami egy kimeneti értéket produkál.

Tehát a kimenet függ a bemenettől. A cél az, hogy felfedezzük a (bemenet) és (kimenet) közötti összefüggéseket. Az függvényt ***leképezési függvénynek*** (mapping function) is nevezik, ami leírja az és kapcsolatát.

Vegyünk egy konkrét példát. Tegyük fel, hogy van egy olyan üzletünk, ahol van vásárlás szám, vásárlás dátuma és elköltött összege. Felügyelt gépi tanulással szeretnénk megtudni a várható értékesítési és vásárlói típusokat. A felügyelt algoritmus azokat a történelmi adatokat veszi fel, amelyek tartalmazzák az bemenetet. Azaz a vásárlások számát, a vásárlás dátumát és az elköltött összegeket. Az kimenet a probléma tartomány, a várható értékesítés vagy ügyféltípus. A felügyelt gépi tanulási algoritmus rétegek, és ennek eredménye egy gépi tanulási modell. Ezt követően a betanult modellnek új inputot adhatunk, és az új input alapján előre jelezhetjük a jövőbeni értékesítési és vásárlói profilokat. A felügyelt tanulásban előzetesen tudjuk a helyes választ, ami azt jelenti, hogy megfelelő eladásokkal és kategóriákkal rendelkezünk, amelyeket algoritmusunk tanulhat és fejlődhet belőle.



A felügyelt gépi tanulási algoritmusok két jól ismert típusa a ***regressziós algoritmus***, amikor az előre jelzett érték folyamatos egy bizonyos tartományban. Az előrejelzések bármilyen értéket felvehetnek: 50, 100, 150 vagy akár milliót.

A másik típus az ***osztályozási algoritmusok***, amikor az előre jelzett ***érték diszkrét*** jellegű. Szóval az előre jelzett érték lehetséges ***opcióival rendelkezik***. Például egy kiskereskedelmi üzlet minősítheti az ügyfeleket kiemelt, közepes és normális vásárlóknak.

**Felügyelet nélküli tanulás** [9]

Felügyelet nélküli algoritmusok azok, amikor ***csak bemeneti adatok*** vannak, és nincsenek kimeneti adatok. Nincs alapfeltevés, tehát az adott bemenetre nincs hozzátartozó kimenet. A cél általában az adatszerkezet megértése az adatok helyes felhasználása érdekében. Ezeket ***önszerveződő*** algoritmusoknak is nevezzük. Tegyük fel, hogy van egy bizonyos adatkészletünk, amelyet A – L betűkkel jelölünk, és szeretnénk megismerni ezeknek az adattagoknak a kapcsolatát és asszociációit. Az algoritmus a következő csoportot hozhatja létre az általa felfedezett kapcsolatok alapján. [A, I, J]; [B, K, H]; [C, H, F]; [E, G, L]. Az általunk használt felügyelet nélküli tanulási algoritmus típusától függ, hogy miként szervezi ezeket a kapcsolatokat és asszociációkat.

Vegyünk egy példát. Tegyük fel, hogy három különböző autók vannak, hétköznapi autók, sportautók és elektromos autók. Az autókat betöltjük egy felügyelet nélküli tanulási algoritmusba, amely kiszámítja az egyes autókkal kapcsolatos összefüggéseket. Ebből lesz egy gépi tanulási modellünk. A gépi tanulási modell felhasználható a jövőbeli minták előrejelzésére. Ennek értelmében a felügyelet nélküli algoritmusok ahelyett, hogy a visszajelzésekre támaszkodnának, mint a felügyelt algoritmusok. A felügyelet nélküli algoritmusok a közös vonások azonosításán alapulnak. Akár képes felismerni az adatkészletek közötti különbségeket is, bizonyos tulajdonságok hiányát vagy jelenlétét.



A gépi tanulási algoritmusok egyik jól ismert típusa a ***csoportosító algoritmusok***. Ezt a típusú algoritmust akkor használják, ha egy adott adatot csoportosítani kívánunk. Például az ügyfelek csoportok szerinti szétválasztása, hogy a hirdetéseket megfelelő módon címezzék. A klaszterező algoritmus felismeri az ügyfél jellemzői közötti mintázatot. Számos különböző csoportosító algoritmus létezik, mint például a K-csoportosítás. A másik algoritmus csoport az ***asszociáció***. Az ilyen típusú ***algoritmusok*** célja, hogy asszociációkat hozzanak létre nagy adatbázis adatelemei között. Célja a változók közötti hasznos kapcsolatok feltárása.

**Megerősítő tanulás** [10]

A gépi tanulási algoritmus harmadik típusa a megerősítő tanulás, másnéven célorientált tanulás. Az ilyen típusú algoritmusok célja egy közvetítő előállítása, úgynevezett az ***ágens***. A tanulási algoritmus az időben ***változó jutalmak*** alapján megtanulják a környezetükből a feladatot. Az ágens végrehajt egy műveletet a környezetben, majd tanul ebből a műveletből. A tanulás visszavezethető ***környezeti állapot*** és jutalom formájában. Ennek felhasználásával az ***ágens*** törli és ***frissíti belső szabályrendszerét***, hogy egyre több tapasztalatot szerez. Az ilyen típusú algoritmusok célja a sok lépéses komplex problémák megoldása. Például egy összetett játék megoldása. A megerősítő tanulási algoritmusok üres állapotból indulnak ki. A megfelelő jutalmak alkalmazásával fokozatosan javítják a teljesítményt. A megerősítő algoritmusokat általában olyan autonóm rendszerekben használják, amelyek ***emberi útmutatás nélkül*** döntenek.



**Tanulási módok**

A gépi tanulás módjai két nagy csoportba válaszhatjuk szét: [6]

**Offline**

Kötegelt vagy ***offline*** tanulási módszerek. ***Nem*** képes ***fokozatosan*** tanulni: az összes rendelkezésre álló adat felhasználásával kell kiképezni. Ez általában sok időt és számítási erőforrásokat igényel. Először a rendszert betanítják, majd elindítják a termelésbe. Tanulás nélkül futnak csak alkalmazzák a tanultakat.

**Online**

Az ***online*** tanulási módszerek. ***Inkrementálisan*** tanul az adatpéldányok egymás utáni adagolásával mini kötegek (***batch***) formájában. Minden tanulási lépés gyors és olcsó, így a rendszer menet közben tanul az új adatokból.

Az online tanulási rendszerek egyik fontos paramétere, hogy milyen gyorsan kell alkalmazkodniuk a változó adatokhoz: ezt nevezzük tanulási aránynak (***learning rate***). Ha magas tanulási arányt állítunk be, akkor a rendszere gyorsan alkalmazkodni fog az új adatokhoz, de hajlamos gyorsan elfelejteni a régi adatokat is. Az online tanulás nagy kihívása, hogy ha ***rossz adatokat*** táplálnak be a rendszerbe, akkor a rendszer teljesítménye fokozatosan csökken.

## A gépi tanulás folyamata

Az előző bekezdésekben megértettük, hogy mi a gépi tanulás és annak különböző alkalmazásai. Majd részleteztem a különböző algoritmusok, amelyeket a különböző üzleti problémákra alkalmaznak. Itt az ideje, hogy egy kicsit elmélyüljünk és megértsük, hogyan valósítjuk meg valójában a gépi tanulást. A következő részben bemutatásra kerül a gépi tanulás folyamata.

**Gépi tanulás menete** [11]

A folyamat négy nagyobb szakaszra lehet osztani:

* ***Probléma meghatározás***: A folyamat egy probléma meghatározásával kezdődik. Itt azonosítjuk és elemezzük üzleti problémánkat.
* ***Adatszerzés***: Ezután folytatjuk az adatszerzés szakaszát, és előkészítjük az adatokat. Ezt követően szét kell választanunk az adatot tanító és tesztelő halmazra.
* ***Modell tanítás***: Miután meg vannak a megfelelően előkészített adatok, neki láthatunk a modell betanítására. A tanítás befejezése után kiértékeljük a modellt, hogy mennyire felel meg az adott probléma megoldására.
* ***Modell telepítés***: Az utolsó fázis a modell telepítés. Ekkor kerül használatba a betanított modell. Ide tartozik még a modell karbantartása is.



**Probléma meghatározása**

A probléma-meghatározási szakasz az első szakasz a gépi tanulási folyamatban. A gépi tanulás problémájának első és legfontosabb lépése az ***üzleti kérdés*** egyértelmű meghatározása, amelyre választ szeretnénk adni. Megpróbáljuk előre jelezni a következő hónap eladásait a készletelőzmények alapján? Vagy a demográfiai információk alapján kategorizálni az ügyfélkörünket.

Az adattudomány vagy a gépi tanulás öt féle típusú kérdésre tudd választ adni.

* Választási kérdések: Ez „A” vagy „B”? Ez az ügyfél vásárol vagy sem?
* Eldöntendő kérdések: Ez jó? Ez rossz? Egészséges? Van-e rendelleneség?
* Mennyiségre vonatkozó kérdések: Hány darab terméket fogok eladni a következő negyedévben?
* Csoportosítással kapcsolatos kérdések: Hogyan van ez megszervezve? Melyek a különböző vásárlói kategóriák?
* Előrejelzésre releváns kérdések: Mit tegyek ezután? Mi a következő lépés?

A probléma megfogalmazása egy kérdéssel kezdődik. A kérdésre való választ, pedig a gépi tanulás során fogjuk megkapni. Előfordul, hogy a folyamatosan fejlődő világban ugyan arra a kérdésre később más válasz társul. Esetemben több kérdés forma is megmutatkozik. „X” vagy „Y” tananyagot/forrás válaszam a tanulásra? Napi hány óra tanulást töltsek gyakorlattal vagy elmélettel? Milyen típusú tanuló vagyok? Az idő függvényébe ezek különböző válaszokat produkálhatnak.

**Adatszerzés** [6]

A második szakasz az adatforrással kapcsolatos lépésekkel foglalkozik. A gépi tanulási algoritmus olyan adatokon tevékenykedik, amelyeket megtanul és felhasznál a problémák megoldására. Az adatokat egy gépi tanulási folyamatba kell táplálnunk. Tehát először a probléma megoldásához szükséges ***adatbegyűjtés*** feladatával kell foglalkozni.

Jellemzően a szervezeti tudás vagy a történeti ***adatok*** több különböző rendszerben vannak ***szétszórva***. Például egy HR-rendszer, amely beszélhet REST-ről, egy 30 éves pénzügyi rendszer, amely csak a fájlintegrációt érti, egy CRM-rendszer, amely SOAP-t protokollon keresztül kommunikál. Tegyük fel egy konfigurációkezelő rendszer, amely az adatbázist közvetlenül SQL nézeteken keresztül teszi elérhetővé vagy akár érzékelők küldik az adataikat a saját eltérő formátumaiban.

Miután begyűjtöttük az adatokat a következő lépés az ***adatelőkészítés*** egy tanulási algoritmus bemenetére. Az adatok előkészítése az egyik legfontosabb rész az adatszerzés folyamatába. Az előkészítés azért is nélkülözhetetlen, mivel az adatok eltérő forrásokból származnak. Különböző műveleteket hajtunk végre ebben a lépésben: [12]

* a ***hiányzó adatok*** azonosításával és kezelésével
* az adatokban lévő felesleges ***attribútumok*** ***eldobása***
* a ***kiemelkedő adatok*** azonosításával
* az ***adatok transzformációja***

Ennek a lépésnek a fő célja, hogy az adatokat a gépi tanulási algoritmusokhoz illesszük. A gépi tanulás algoritmusnak konkrét elvárásaik vannak a beléjük adott adatformátumok tekintetében. Például egyes gépi tanulási algoritmusok normálisan elosztott adatokat igényelnek. Meglepő módon ezt a lépést tartják a ***legidőigényesebb lépésnek*** egy gépi tanulási folyamatban. Néhány adatkutató azt állítja, hogy idejének 60-80% -át tölti az adatbányászattal és adatelőkészítéssel.

Amíg nem megfelelőek az adatok, addig nem érdemes elkezdeni a modell tanítását. Hiszen a nem egyértelmű vagy használhatatlan adatokkal való tanítás, értelmetlen és haszontalan modellt eredményezz. Informatikában gyakran használt kifejezés erre, hogy „***Szemét be szemét ki***”.

Előkészítést követő feladat az ***adatok szétválasztása***. Az adatszegregáció abból áll, hogy az adatokat két halmazra osztjuk. ***Képzési készlet***, amelyet a gépi tanulási algoritmus megtanulására és megtanítására használnak. A gépi tanulási algoritmus kiképzése az algoritmus bizonyos belső paramétereinek beállításának folyamata. ***Tesztkészlet***, amelyet a gépi tanulási algoritmus teljes teljesítményének és általánosításának értékelésére használnak. A teszt adatokat elrejtjük a gépi tanulási algoritmus elől, hogy azokat később ki tudjuk értékelni. Ahhoz, hogy megtudjuk milyen jól teljesít a modell, olyan adatokra van szükség, amelyekkel még nem találkozott. Kétségtelen, hogy az algoritmus jobban fog teljesíteni a képzési adatokon mint a tesztadatokon. Voltaképp memorizálta a megoldást, nem pedig „megértette” a megoldáshoz vezető utat. A modell értékelése alapvető irányelv.

**Modell tanítás** [13]

Miután az adatok rendelkezésre állnak, következhet a modellképzése. A gépi tanulás folyamatának lényege a modell tanítás. A gépi tanulási algoritmusok itt illeszkednek a képzési adatainkhoz és előreláthatóan a megadott paraméterek alapján ***módosítják belső paramétereit***. A gépi tanulási algoritmusok általában a polcon található receptek, amelyeket a megfelelő üzleti problémákra, például osztályozásra, lineáris regresszióra és klaszterezésre vonatkozó problémákra alkalmaz. Ez a lépés magában foglalhatja egynél ***több*** gépi tanulási ***algoritmus kipróbálását***, mivel nem tudjuk előre, hogy melyik algoritmus illik valójában a mögöttes adat jellemzőkre.

A tanítás után történik a ***modell kiértékelése***. Miután betanítottuk a gépi tanulási modellünket, itt az ideje, hogy teszteljük és értékeljük a teljesítményünket. Elég gyakori, hogy több gépi tanulási modellünk van, amelyek közül szeretnénk választani, és összehasonlítjuk teljesítményüket. Ezért hasznos, ha az adatot szétválasztjuk több ***validációs adatkészletre***. Vannak ***teljesítményértékelési*** mutatók, amelyek objektíven felhasználhatók a gépi tanulási algoritmusok teljesítményének összehasonlítására. Ilyen például az osztályozási táblázat és osztályozási algoritmusok, valamint az RSK (Robinson–Schensted–Knuth) és a lineáris regressziós algoritmusok.

**Modell telepítés** [11]

Az utolsó szakasz a modell telepítésé. Miután kiválasztottuk a gépi tanulási modellünket és elégedettek vagyunk a teljesítményével, eljött az ideje, hogy ezt a felhasználók és az üzleti élet számára is felhasználjuk. A gépi tanulási modell általában elfogadja az független rendszerek bemenetét (***upstream***), alkalmazza a gépi tanulási technikákat. A kimenetet és az eredményeket átadja a komplex rendszereknek (***downstream***).



Lehetőség van ***kiterjeszt***eni a gépi tanulási modelljeinket ***API***-k, háttérfeldolgozási feladatok vagy bármilyen más megfelelő mechanizmus formájában.

Végül a ***modell figyelése***. A modell üzembe helyezése és működése után a történet még nem fejeződött be. A gépi tanulási modellek nem jelentenek kivételt azoktól az általános szoftveres megoldásoktól, amelyek ***karbantartást*** és ***üzemeltetési*** gondozást igényelnek. Ebben az utolsó lépésben a modell teljesítményünket a teljesítmény helyessége alapján mérjük. Például mennyire pontosak az előrejelzései, a válaszidő, a használt funkciók eloszlása ​​stb. Ez az értékes információ visszajuttatható gépi tanulási folyamatunkba annak jövőbeni teljesítményének javítása érdekében. A modellek kiépítését és felügyeletét együttesen gépi tanulás üzemeltetésnek (***MLOps***) nevezik. Célja, hogy növelje az automatizálást és javítsa a gyártás minőségét.

# Probléma meghatározása

Itt az ideje bemutatni azt az ***üzleti problémát***, amelyet a gépi tanulási megoldásunk megvalósításához fogunk használni. Diákokkal kapcsolatos adatkészletet fogunk igénybe venni.

A cél az, hogy az adatokból kinyerjük, hogy milyen kontextusba tanulnak a legmegfelelőben a diákok. Ha felidézzük a korábban bevezetett gépi tanulási folyamatot, akkor ez a múltbéli adatokból való tanulás.

**<<Ha lesz egyetemi adathalmaz azt bemutatni>>**

**<<Különösen milyen típusú kérdésre próbálunk választ adni?>>**

**<<Milyen üzleti probléma típus? (regresszió, klasszifikáció)>>**

**<<Milyen tanulási algoritmus/algoritmusokkal?>>**

Miután meghatároztuk problémánkat, az diákok adatkészlete képezi majd az alapot, amelyre a gépi tanulási folyamatot alkalmazzuk. A gépi tanulás folyamatát lépésről lépésre fogjuk használni az adatkészletünkön egy gépi tanulási megoldás megvalósításához.

## Az adatforrás a kezdetek kezdete

**Mik lehetnek az adatok?** [14]

Lehet bármilyen feldolgozatlan tény, érték, szöveg, hang vagy kép, amelyet nem értelmeznek és elemeznek. Az **adat**ok az összes adatelemzés, gépi tanulás, mesterséges intelligencia legfontosabb része. Adatok nélkül nem képezhetünk egyetlen modellt sem és minden korszerű kutatás és automatizálás hiábavaló lesz. A nagyvállalkozások rengeteg pénzt költenek csak azért, hogy minél több biztos adatot gyűjtsenek.

Meg kell különböztetni az adat és információ közti különbséget. Az **információ** olyan adatok, amelyeket értelmeztek és manipuláltak, és amelyek most a felhasználók számára érdemi következtetéseket vonnak le. Tehát az adatunk valamilyen transzformáción és adattisztítási lépéseken esett át.

A következő lépcsőfok a **tudás**, ami az információ után következik. A megszerzett tudás a következtetett információk, tapasztalatok, tanulás és meglátások kombinációja. Eredménye egy koncepció kiépítése.



Vegyünk egy példát! Egy vásárlói központ tulajdonosa felmérést készített. Hosszú listát állított fel az ügyfelektől feltett kérdések és válaszok alapján. A ***kérdések és válaszok listája*** maga az adat. Minden alkalommal, amikor bármire következtetni akar, akkor az ügyfelek miden egyes kérdéseit és válaszait át kell nézni. Az összes adat végig nézése nem hasznos és nagyon időigényes folyamat. Az időveszteség csökkentése és a munka megkönnyítése érdekében nyújtanak segítséget a szoftverek, matematikai számítások, grafikonok. Ezek a támogató eszközök az adatokat manipulálják, megtisztítják, transzformálják. A manipulált ***adatokból levont következtetés*** az információ. Tehát az adatokból nyerjük az információkat. A tudásnak szerepe van abban, hogy ***különbség***et tegyünk ***két*** egyforma ***információ***val rendelkező egyén ***között***. A tudás valójában nem technikai tartalom, hanem az emberi gondolkodási folyamathoz kapcsolódik.

## Adatok strukturáltságának változatai

Az adatokat strukturáltság szempontjából három nagy kategóriába sorolhatjuk: a strukturált, a félig strukturált és a strukturálatlan adatok.

**Strukturált adatok** [2]

A strukturált adatok olyan információk, amelyeket ***formáztak*** és jól definiált adatmodellekké alakítottak. A nyers adatok előre megtervezett mezőkbe vannak leképezve, amelyeket később kibonthatnak és könnyedén beolvashatnak. Az SQL relációs adatbázisok, amelyek sorokból és oszlopokból álló táblákból állnak, tökéletes példa a strukturált adatokra.

A strukturált adatok relációs modellje kihasználja a memóriát, mivel ez minimalizálja az adatok redundanciáját. Ez azonban azt is jelenti, hogy a strukturált adatok jobban függenek egymástól és ***kevésbé rugalmas***ak.

A strukturált adatokat emberek és gépek egyaránt generálják. Számos példa található a gépek által létrehozott strukturált adatokra. Például egy bankszámlakivonat, amely tartalmazza a dátumot, az időt, az összeget stb. A strukturált adatok szervezése miatt ***könnyebb elemezni***, mint a félig strukturált és a strukturálatlan adatokat.

**Félig strukturált adatok** [2]

Az ilyen adatok félig strukturáltak. A félig strukturált adatok olyan adattípusok, amelyek bizonyos jellemzőkkel bírnak. ***Nem*** korlátozódnak olyan ***merev*** struktúrába, mint amilyen a relációs adatbázisokhoz szükséges.

A félig strukturált adatokra példa a fényképek. A képnek sincs előre meghatározott struktúrája. Ennek ellenére strukturált attribútumai vannak (helyazonosító, eszközazonosító, dátum). Tárolás után a képekhez hozzárendelhetők címkéket is, mint a „háziállat” vagy a „kutya”, hogy struktúrát adjanak.

Bizonyos esetekben a strukturálatlan adatokat félig strukturáltnak minősítik, mert egy vagy több osztályozó attribútummal rendelkeznek. Ilyenek lehetnek például a naplófájlok, JSON fájlok, érzékelő adatok, csv fájlok stb.

**Strukturálatlan adatok** [2]

Az abszolút ***nyers*** formában jelen lévő ***adatok***at strukturálatlannak nevezzük. Ezeket az adatokat összetett elrendezése és formázása miatt ***nehéz feldolgozni***. A strukturálatlan adatok sokféle formát ölthetnek, beleértve a közösségi média bejegyzéseket, csevegéseket, műholdas képeket, Internet of Things (IOT) érzékelő adatokat és e-maileket.

Nem rendszerezett adat (strukturálatlan) nem illeszkedik a relációs adatbázis sorokba és oszlopokba. Például ilyenek lehetnek a szöveges fájlok, képek, videók, hangposták, hang fájlok stb.

**Különbségek a strukturált, félig strukturált és strukturálatlan adatok között** [15]

Az interjúk analógiája alapján különböztessük meg ezt a három típusú adatstruktúrát. Tegyük fel, hogy háromféle állásinterjú létezik: strukturálatlan, félig strukturált és strukturált interjúk.

Egy strukturálatlan formátumú interjúban a feltett ***kérdések*** teljesen ***az*** ***interjúztató*** ***választása***. Ő eldöntheti azokat a kérdéseket, amelyeket fel akar tenni, és azok sorrendjét. A strukturálatlan kérdések népszerű példái: „mesélj magadról” és „írd le ideális szerepedet”.

Egy másik típus a strukturált interjú. Ebben az esetben a kérdező szigorúan követni fogja a ***HR*** osztály által létrehozott ***forgatókönyv***et, és ugyanazt a forgatókönyvet fogja használni minden pályázó számára.

A harmadik típus félig strukturált. Egy félig strukturált interjúban az interjúkészítő egyesíti a strukturálatlan és a strukturált interjú elemeit. Ez magában foglalná a strukturált interjúhoz hasonlóan a kötelező elemeket. Ugyanakkor ***rugalmas***an alkalmazhatja a ***kérdések***et a helyzetnek megfelelően, ami a strukturálatlan interjú egyik aspektusa.

A következő pontok rámutatnak a strukturált, félig strukturált és strukturálatlan adatok közötti különbségekre:

* ***Szervezet***: A strukturált adatok rendelkeznek a legmagasabb szintű szervezettséggel. Míg a félig strukturált adatok részben szervezettek. Végül a strukturálatlan adatokat egyáltalán nem rendezik szervezettséggel.
* ***Rugalmasság és méretezhetőség***: A strukturált adatok kevésbé rugalmasak és nehezen méretezhetők. Míg a félig strukturált adatok rugalmasabbak és méretezhetők. A strukturálatlan adatoknak nincs sémájuk, amely a legrugalmasabbá és méretezhetőbbé tenné a másik kettő közül.
* ***Változatkészítés***: A strukturált adatok relációs adatbázison alapulnak, verziószámcsomagok állnak rendelkezésre. Másrészről félig strukturált adatokban csak részleges adatbázist támogat. Végül a strukturálatlan adatokat az adatbázis nem támogatott.
* ***Tranzakciókezelés***: A strukturált adatokban jelen van a tranzakció kezelés és az egyidejűség. Míg a félig strukturált adattranzakciók a DBMS-től függenek és az adatok egyidejűsége nem áll rendelkezésre. Végül a strukturált adatokban sem a tranzakciókezelés, sem az adatok egyidejűsége nincs jelen.

## Hogy szerezünk adatokat?

**Az adatkészletek típusa** [16]

Az adattudományi projekt kialakítása során többen azt feltételezik, hogy létre kell hozni az algoritmusok változatait. Majd megbecsülni a modell teljesítményét a képzési adatokon. Végül értelmezni az előre jelzett eredményeket. Mielőtt azonban ezeket a lépéseket elkezdnénk, nemcsak a megfelelő adatot kell megszereznie, hanem azt is ellenőriznie kell, hogy az megfelelő címkézéssel van ellátva. Ha nincs szükség konkrét ***adatgyűjtés***re, akkor is sok időbe telhet egy olyan adatforrás megtalálása, amely a legjobban felel meg a projektnek.

Több ezer nyilvános adatkészlet különböző témákban érhető el online.

* ***Adatkészletek az általános gépi tanuláshoz***: Ebben az összefüggésben az „általános” gépi tanulás alatt a regressziót, az osztályozást és a csoportosítást értjük. Ezek a leggyakoribb gépi tanulási feladatok.
* ***Adatkészletek a mély tanuláshoz (deep learning)***: Az általános gépi tanulástól némileg különböző a mély tanulás. Alapvetően a mély tanulás olyan, mint a neurális hálózat. A különbség, hogy a neuronok itt számos rétegekbe szerveződnek. Leginkább a kép, szöveg vagy hangadatok feldolgozására használják a mély tanulást.
* ***Adatkészletek a természetes nyelv feldolgozásához***: A természetes nyelv feldolgozása (N.L.P.) a szöveges adatokról szól. Az olyan rendezetlen adatokhoz, mint a szöveg, különösen fontos, hogy az adatkészletek valós alkalmazásokból származzanak. Ennek érdekében könnyedén tudunk elvégezni az adat realitásának és életszerűségének az ellenőrzését (sanity check).
* ***Adatkészletek az idősor elemzéshez***: Az idősor elemzéséhez időbélyeggel ellátott megfigyelésekre van szükség. Más szavakkal, minden adatrekordot az idő folyamán követnek nyomon és rögzítenek.
* ***Adatkészletek az ajánló rendszerek számára***: Az ajánló rendszerek átvették a szórakoztató és az e-kereskedelmi iparágakat. Remek példa az ajánló rendszerekre a Facebook, az Amazon, a Netflix és a Youtube.

Ahhoz, hogy minél kevesebb időt fordítsunk a megfelelő adatkészlet keresésére, tisztában kell lenni, hogy hol keressük meg. A továbbiakban részletezem, hogy hol tudunk a gépi tanulási projektekhez ***nyilvános***an elérhető ***adatforrás***t találni.

**Adatforrás gyűjtemények helye** [17]

1. Google adatkészlet-kereső

A Google a keresőmotorok óriása. A gépi tanulással foglalkozó szakembereknek segítenek megtalálni a megfelelő adathalmazokat. A ***keresőmotor*** nagyszerű munkát végez a kulcsszavakhoz kapcsolódó adatkészletek megszerzésében különböző forrásokból. Ideértve a kormányzati weboldalakat, a Kaggle-t és más nyílt forrású adattárakat.

1. .gov adatkészletek

Mivel az Egyesült Államok, Kína és még sok más ország mesterséges intelligencia nagyhatalommá válik, az adatok demokratizálódnak. Az ezen adatkészletekre vonatkozó szabályok és előírások általában szigorúak. A ***nemzet*** különböző szektoraiból származnak az ***adatok***. Ezért óvatos használat ajánlott. Van néhány olyan ország, amelyek nyíltan megosztják adataikat.

1. Kaggle adatkészletek

A Kaggle a gépi tanulás és a mély tanulási kihívások tárhelyéről ismert. A Kaggle relevanciája, hogy ***adatkészlet***eket nyújt, és egyúttal a tanulók és az gépi tanulási szakemberek ***közösség***ét is biztosítja. Nagyban elősegíti a fejlődésüket a jövendő adattudósoknak. Minden ***kihívás***nak külön adatkészlete van és általában megtisztítják az adatokat, hogy ne kelljen a tisztítást elvégezni. Helyette az algoritmus finomítására összpontosíthatunk. Az adatkészletek könnyen letölthetők. Az erőforrások részben előfeltételek és linkek találhatók az anyaghoz. A Kaggle egy fantasztikus webhely a kezdők számára, akik belevághatnak a gépi tanulás és a mélytanulás alkalmazásaiba. Továbbá részletes erőforráskészletet nyújt a gépi tanulás haladó gyakorlói számára is.

1. Amazon adatkészletek

Az Amazon nyilvántartásba vette a szervereiken elérhető néhány adatkészletet. Egyik nagy előnye, hogy amikor Amazon Web Services (***AWS***) erőforrásokat használunk a modellek kalibrálásához és módosításához, akkor ***lokális adatok***at használunk. A helyileg elérhető adatkészleteknek a használata többszörösen ***gyors***ítja az ***adatbetöltés***i folyamatot. A nyilvántartás számos adatkészletet tartalmaz az alkalmazások területe szerint, például műholdas képek, ökológiai erőforrások stb.

1. UCI Machine Learning Repository

Az UCI Machine Learning Repository könnyen kezelhető és tisztított adatkészleteket kínál. Ezek már ***régóta*** az egyik leggyakrabban ***használt*** adathalmazok a tudományos területeken.

**Hogyan keressünk könnyeden adatkészletet?** [18]

A mai világban sok tudományterület tudósai és egyre több elemző, újságíró gyűjt és állít elő adatokat. Az interneten sok ezer adattár található, amelyek milliónyi adatkészlethez nyújtanak hozzáférést. A helyi és nemzeti kormányok világszerte közzéteszik adataikat is. Az adatokhoz való könnyű ***hozzáférés*** érdekében elindították az ***adatkészlet keresés***t. Elősegítve, hogy a tudósok, az újságírók, az adatkezelők vagy bárki más megtalálhassa a munkájához szükséges adatokat.

A ***Google Tudós*** működéséhez ***hasonlóan*** az adatkészlet keresés lehetővé teszi az adatkészletek megtalálását, bárhol is tárolják őket. Legyen szó kiadói webhelyről, digitális könyvtárról vagy a szerző személyes weboldaláról. Az adatkészlet keresés létrehozásához ***irányelvek***et dolgoztak ki az adatforrás szolgáltatók számára. Olyan értelemben írják le az adataikat, hogy a Google (és más keresőmotorok) jobban megértsék oldalaik tartalmát. Ezek az irányelvek ***kiemelkedő információk***at tartalmaznak az adatkészletekről:

* Ki hozta létre az adatkészletet?
* Mikor jelent meg?
* Hogyan gyűjtötték az adatokat?
* Milyen feltételek vonatkoznak az adatok felhasználására?

Ezután összegyűjtött információkat elemzik, hogy ugyanazon verziók hol változtak. Találhatunk olyan publikációkat, amelyek leírják vagy megvitatják az adatkészletet. Az információk leírása ***nyílt szabvány*** alapú. Aki adatokat publikál, leírhatja az adatkészletét a szabvány alapján. Arra ösztönözve az adathalmaz-szolgáltatókat, hogy fogadják el ezt a közös szabványt, hogy minden adatkészlet része legyen ennek a robusztus ökoszisztémának.

Továbbá megtalálható ***több*** környezeti és társadalomtudományi adatkészlet, valamint más ***tudományág***ak adatai. Ideértve a kormányzati és a hírszervezetek által szolgáltatott adatokat. Egyre több adattár használja a közös szabványt az adatkészletek leírására. A felhasználók által az adatkészlet keresésben megtalált adathalmazok változatossága és lefedettsége a mai napig is növekszik.

Az adatkészlet keresés ***több nyelven*** működik. Egyszerűen be kell írni, amit keresünk. Majd a keresőmotor segít a letárolt adatforrás megtalálásához.

Például, ha elemezni szeretnénk a diákok teljesítményére vonatkozó rekordokat, kipróbálhatjuk ezt a lekérdezést az adatkészlet keresésben:



Látni fogjuk a Kaggle által szolgáltatott adatait, valamint olyan tudományos adattárakat, mint például a Harvard Dataverse és a Statista. Számos adattárak célja, hogy támogassa a projektet és segítse, hogy számos adatkészletét kereshetővé tegyen ebben az eszközben.

Ez az adatkészlet keresés bevezetés azon kezdeményezések egyike, amelyek célja az adatforrásokat könnyebben beépítsük a projektjeinkbe és termékeinkbe. Remélhetőleg sokan a nyílt szabványokat használják az adatok leírására. Lehetővé téve a felhasználók ​​számára, hogy megtalálják azokat az adatokat, amelyeket keresnek.

## Saját adatforrásom ismertetése

Több adatforrással is fogok foglalkozni a diplomamunka során. Mindegyik adatforrás diákokkal kapcsolatos információt tartalmazzák.

**Matematikai teljesítmény**

Az adatkészlet az ***Kaggle*** adattárból származik. Az adatállomány ***395*** ***rekord***ot ír le az előrejelzésekről a diák teljesítményére mutatnak. Az adatforrás a **student-mat.csv** néven található meg.

Ez az adatkészlet a tanulók végeredményeit tartalmazza a matematikai periódusok végén, számos olyan funkcióval, amelyek befolyásolhatják vagy nem befolyásolhatják hallgatók jövőbeli eredményeit. Az attribútumok leírását az alábbi linken elérhető: <https://www.kaggle.com/janiobachmann/math-students>.

Az eredeti adatforrás számos kategorikus értékkel rendelkezik. Továbbá sok jellemzővel rendelkezik összes ***33*** ***oszlop***a van az adatkészletnek. Numerikus jellemzőkből 16, binárisból 13 és szövegesből pedig 4 darab található.

**Diák bizalom**

Az adatkészlet a ***data.gov*** adattárban található. Az adatállomány ***1829 rekord***ból áll és megmutatja mekkora bizalma van a diáknak az adott iskolával szemben. Az adatok a **2019\_PublicDataFile\_Students.csv** fájlban fellelhetők. Link: <https://catalog-next.data.gov/dataset/2019-public-data-file-students>

Elősegíti a családok felfogásának, a diákok és a tanárok megértésének megkönnyítését iskolájukkal kapcsolatban. A 6-12. osztály tanulói vettek részt a felmérésben.

Továbbá ***11 oszlop***ot, jellemzőt tartalmazz az adatkészlet. Ebből 9 numerikus és 2 darab szöveges értékből áll össze. Számos nem definiált, üres értkéket tartalmaznak. Ezektől az értékektől az adattisztítás során kell megszabadulni.

**<<Ha van még megemlítendő adatforrás az mehet ide>>**

**A Képek ereje**

Nehéz következtetéseket leszűrni az adatokból. Akár napokig is nézhetjük a csv fájlokat, de valós konzekvenciát nem fogunk kinyerni belőle. Az ember egyik hasznos fegyvere az ***intuíció***. Ez a fegyver néha félre, máskor rávezetnek a probléma megoldáshoz.

Ahhoz, hogy megtudjunk bizonyosodni a megérzésünk érvényeségéről ***vizualizáció***t és adatelemzési lépéseket alkalmazunk. A következő fejezetben az adatok vizualizációját fejtem ki. Hiszen:

*„Többet mond egy kép*

*mint száz bekezdés”*

# **Reprezentációs módszerek**

## Miért hasznos az adatok megjelenítése?

Az adattudományban az egyik legfontosabb készség az adatok eloszlásának és összetettségének vizualizálása és megértése. Manapság elsősorban a gépi tanulásra és az algoritmus működésére összpontosít.

Ez nem azt jelenti, hogy a gépi tanulás ismerete elhanyagolandó. Nyilvánvalóan a legjobb adattudós könnyedén eligazodik a gépi tanulási algoritmusok nagy részével. Mindenesetre az adattudomány nem korlátozódik le a gépi tanulásra, inkább az egy képesség. A készség alatt azt kell érteni, hogy mennyire érted a probléma területet (domain), mennyire tudsz elmélyülni az adatokban. Elkapni a ***rejtett összefüggések***et és megtalálni az adatokban rejlő üzenetet. Az adat beszél magáról. A legjobb mód arra, hogy szóra bírjuk az ***adatmegjelenítés***. [19]

**Mi az adatmegjelenítés?**

Az adatmegjelenítés az információ (adatok) felvételének és vizuális kontextusba, például térképbe vagy grafikonba helyezésének módszere. A fő cél a nagy adatkészletek vizuális formába történő megjelenítés, hogy lehetővé tegye az adatokon belüli összetett ***kapcsolatok*** könnyű ***megértését***. [12]

Tehát a vizualizáció választ ad azokra a kérdésekre, amelyeket nem tudunk.

**Miért fontos az adatmegjelenítés?**

Az adatmegjelenítést egyre inkább minden sikeres adatközpontú elemzési stratégia alapvető utolsó lépésének tekintik. Amikor az adattudósok egy összetett projekt közepén vannak, szükségük van egy módra az összegyűjtött adatok megértésére. Az adatmegjelenítés megkönnyíti az adatcsoportok mintáinak, ***előítélet***einek és ***kiugró érték***einek ***észlelés***ét is.

Továbbá vezeti a célközönséget, hogy felfedezzék a figyelmet igénylő területeket. Korábban észrevétlen kulcsfontosságú ***tények feltárása*** az adatforrásokkal kapcsolatban. Segít az érdekelt feleknek minőségi információkkal szolgálni azáltal, hogy hatalmas mennyiségű adatokat ***könnyen érthető kép***pé és grafikává alakít.

**Az adatmegjelenítés előnyei** [19]

Figyelembe véve, hogy az adatok milyen hatást gyakorolnak a vállalkozás növekedésére, íme néhány előny

* Segít felismerni a legújabb ***trend***eket az üzleti nyereség növelése érdekében.
* Az adatmegjelenítések megkönnyítik a kis és nagy adatok ***megértés***ét az emberi agy számára, ami jobb elemzéshez vezet.
* Segít megérteni a történetet - Az emberi agy nem képes egyszerre nagy mennyiségű számot vagy szöveget megérteni, sőt csak elképzelni. Szüksége van egy vizuális ábrázolásra, hogy értelmezze őket, és ennek következtében a nyers adatokat kézzelfogható fogalommá alakítsa.

## Ábrázolási formák

A történelem számos ábrázolási formát használt az ember. A számítógépek megszületése előtt a barlang rajzoktól a Mona Lisáig számos dolgot próbáltunk meg vizualizálni. Ezeknek a kifejezési formáknak a célja az, hogy gyorsan előtudjunk hívni egy ismétlődő vagy komplex (szavakban nehezen kifejezhető) információt.

Az évek során az ember leginkább két dimenzióba ábrázolt információkat. A mélység bevezetésével a három dimenziót is képesek vagyunk imitálni. Az agyunk maximum ***három dimenzió***t képes ***értelmezni***. Ennek ellenére az adatokban rejlő dimenzió számtalan lehet.

Gondoljunk egy emberre, mint ***adatdimenziók***ra. A dimenziók ebben az esetben év, név, nem. Viszont ez a három jellemző nem sok mindent mond el az emberről, ezért sokkal több attribútumot vizsgálunk (vérkép, családi körülmény, étkezés, magasság, sport tevékenység, stb).

**Ábrázolási típusok**

Négy fő vizualizációk típust tudunk elkülöníteni.

* **Összehasonlító vizualizációk**: Az első típus az összehasonlító vizualizációk, amelyek segítenek az értékek összehasonlításában egy adott dimenzióban.
* **Kapcsolati vizualizációk**: A második típus a kapcsolati vizualizációk, amelyek segítenek felderíteni az adathalmaz potenciális ok-okozati összefüggéseit.
* **Kompozíció vizualizáció**: A kompozíció vizualizációi több, azonos típusú grafikonból állnak, azzal a céllal, hogy összetett információkat közvetítsenek.
* **Eloszlási vizualizáció**: Végül az eloszlási vizualizációk segítenek megérteni a mögöttes adateloszlást.

**Összehasonlító vizualizációk**

Az összehasonlító grafikonok egyik típusa az ***oszlopdiagram***. Az oszlopdiagramon az x tengely különböző kategóriáiból áll például a diák neme, az y tengely értéket mutat például a számosság diák nemenként. Az oszlopdiagram arra szolgál, hogy összehasonlítsuk a különböző ***értékeket egy adott dimenzió felett***. Az adatelemzési kontextusban segít összehasonlítani egy adott tulajdonságot a megfigyelésekkel. [13]



student\_mat.csv

Az összehasonlítási vizualizációk másik típusa a ***vonaldiagram***, amely általában az idő előrehaladásával társul. Például a pontok az adott félévben. Az x tengelyen a periódusok, míg az y tengelyen a pontok vannak. Érdekes módon azt a tendenciát láthatjuk, hogy a pontok száma az ***idő haladás***ával se nem nő se nem csökken. A vonaldiagramok alapvetően ***trendvonal***ak, és az adatelemzés keretében segítenek azonosítani az idő hatását egy adott tulajdonságra.



student\_mat.csv

**Kapcsolat vizualizációk**

A vizualizációk második kategóriája a kapcsolati vizualizációk, amelyek segítenek azonosítani ***két vagy több változó*** közötti lehetséges ok-okozati ***összefüggés***t. Az egyik példa a ***hőtérkép*** (heatmap). A hőtérkép egyszerűen két változó ***korrelációs mátrix***ából áll. De ahelyett, hogy korrelációs értékeket adna az érték mezőbe, egy színt használunk a korreláció erősségének kijelölésére. Általában minél sötétebb a szín, annál erősebb a korreláció. Vessünk egy pillantást az itt látható hőtérképre! [20]



student\_mat.csv

Az x és y tengelyen látható az összes diák jellemző. Gyorsan észrevesszük, hogy az eredmények periódusonként (G1, G2, G2) erősen összefüggenek. Ez érthető, aki egyik időszakban jó jegyeket produkált az nagy valószínűséggel a következőben is jókat fog. Látható, hogy az anya és apa oktatási szintje is összefügg. A magasabb végzettségű szintű szülők hasonló végzettségű partnert fognak választani. Tehát egy hőtérkép megmutatja nekünk a két változó közötti kapcsolat erősségét. Ezért megkönnyíti számunkra az összefüggő jellemzők azonosítását, amelyek véleményünk szerint a legfontosabbak képzési céljaink szempontjából. A hőtérkép szép tulajdonsága, hogy olvashatóbb, mint a korrelációs mutatók által bemutatott egyszerű szám.

Egy másik kapcsolat-megjelenítés a ***szóródási ábra*** (scatter plot). Például van ez a grafikon, ahol az x tengely tartalmazza végső eredményt, míg az y tengely tartalmazza a korábbi osztálybukások száma. Minden egyes pont az adott bukások számán az elért végső pontot jelöli. Az általános tendencia az, hogy a bukások számának csökkenésével nő a végső érdemjegy pontja. Ezt általában ***trendvonal***on keresztül hangsúlyozzák, megmutatva a kapcsolat irányát. Tehát a szóródási ábrák segítenek megérteni a két változó kapcsolatát, és az adatelemzés összefüggésében felhasználják őket a ***két tulajdonság*** közötti ***lineáris kapcsolat*** természetének megértésére.



student\_mat.csv

**Kompozíció vizualizációk**

A vizualizáció harmadik típusa az összetett vizualizáció, amely megkönnyíti számunkra a dolgok kombinálását. Gyakori példa a ***kördiagram***. A kördiagram, amint azt a neve is jelzi, általában különböző kategóriákat képviselnek százalékos arányban, arányos méretben egy különböző színű tortában, és minden szín más kategóriát képvisel. Tehát a kördiagramok megkönnyítik a ***százalékos eloszlás*** megértését, és megkönnyítik számunkra annak megértését, hogy az adott kategória különböző értékei hogyan oszlanak meg az adatkészletünkön.

**Eloszlási vizualizációk**

A vizualizáció utolsó kategóriája az ***eloszlási vizualizáció***k, amelyek lehetővé teszik számunkra, hogy megértsük, hogyan oszlik meg adatkészletünk különböző értéktartományok között [20]. Például láthatjuk, hogy a 8 és 12 pont közötti értékek fordultak elő a legtöbbször, míg a 0 pont fele annyiszor fordult elő. Látszódik, hogy az eloszlás az átlag 10.41 felé tendál. Tehát a hisztogram hasznos a teljes ***adateloszlás*** megértéséhez és az értékek nagy részének megismeréséhez. Különösen hasznos a ***kiugró érték***ek ***felderítés***ére.



**Adat statisztika leíró ábrázolás**

Az egyik vizualizáció, amely összefoglalja a leíró statisztikákat, az a ***daru és doboz diagram*** (whisker box). A diagram egyetlen adatmegjelenítésben összefoglalja az adatok több aspektusát, a medián, a ***felső*** kvartilis (75%), az ***alsó*** kvartilis (25%). A felső szélső érték, amelyet felső negyed plusz 1,5 interkvartilis tartományként határozunk meg. Az ***interkvartilis*** tartomány a felső és az alsó kvartilis közötti intervallum. A szélső érték, amely az alsó ***kvartilis*** mínusz 1,5 interkvartilis tartomány. A daru szakasz a felső és a felső kvartilis, valamint az alsó és az alsó kvartilis között húzódó vonal. Végül a kiugró értékek a felső és az alsó szélsőségeken kívül eső értékek. A daru és a doboz diagram segít ***összefoglal***ni a ***leíró statisztikát***. Hasznos az adatkészlet potenciális ***kiugró érték***einek felderítése is. [21]

****

## Következtetések az ábrákból

**Összehasonlítás következtetés**

A diák adatkészletnél (student\_mat.csv) a jellemzők érték típusának számossága jól kiegyensúlyozott. Az egyedüli jellemző, aminek nincs jó aránya az extra oktatási támogatás attribútum (***schoolsup***). Itt az arány közelítőleg ***85% nem*** vesz igénybe plusz segítséget, míg ***15% igen***. Adatkészlet ***kiegyensúlyozás***ához túl vagy alul mintavételezési technikák alkalmazhatók.

**Kapcsolat következtetés**

A kapcsolat feltérképezése a jellemzők között nagyon fontos. Alapvetően kettő vagy több jellemző közötti ***korreláció*** megmutatja, hogy az egyik jellemző milyen valószínűséggel idézi elő a másik jellemzőt. Például a hőmérséklet a fagylalt eladások ára.

Ha az attribútumok erősen (pozitívan) függenek egymástól az kihat a modell teljesítményére. A probléma az úgynevezett ***multikollinearítás***. Bekövetkezik, ha egy bementi érték előre jelezz több értéket nagy precizitással. Ez ***torz*** vagy félrevezető ***eredmények***hez vezethet. [22]

Szerencsére a hőtérképről egyszerűen leolvasható, hogy az attribútumok többsége nem függ nagyon erősen egymástól.

**Eloszlás következtetés**

Tudni, hogy milyen eloszlása van az adatunknak alapvető. Miért jó tudni, hogy a minták mennyire térnek el az átlagtól?

Első sorban nagyon jól lehet detektálni a ***kiugró érték***eket.

Továbbá újabb és újabb minták gyűjtésénél, egy minta átlaga ***konvergál*** az összes minták ***átlag***ával. Ez azt jelenti, hogy pontos előrejelzéseket tudunk adni a normális eloszlás alapján.

**Statisztikai elemzés**

A képek és az ábrák nagyon jól segítenek egy összképet adni az adatokról. Jól tudjuk vele reprezentálni az attribútumok összefüggését, eloszlását, kategóriáját. A képekkel való ábrázolás az érme egyik oldala.

Szükségünk van statisztikai leírásokra is. Mivel a ***gép*** az ábrákból nem tudd kinyerni magas szintű következtetéseket, így maradnak neki a ***számok***. Ahhoz, hogy a gép által is értelmezhető összefüggéseket készítsünk, statisztikai elemzési technikákat alkalmazhatunk.

# Adatelemzés lépései és adathalmazok ismertetése

## Adat a gépi tanulásban

**Az adat** [23]

A gépi tanulási módszerek példákból tanulnak. Fontos, hogy jól megértsük a bemeneti adatokat és az adatok leírásakor használt különféle terminológiát. Ebben a szakaszban megismerhetjük a gépi tanulásban használt definíciókat, amikor az adatokra hivatkozik.

Ha az adatokra gondolok, akkor ***sor***okra és ***oszlop***okra gondolok, például egy adatbázis táblára vagy egy Excel táblázatra. Ez egy hagyományos adatstruktúra, és ez általános a gépi tanulás területén. Egyéb adatok, például képek, videók és szöveg, az úgynevezett strukturálatlan adatok egyelőre nem vesszük figyelembe.



***Példány***: Egyetlen adatsort nevezünk példánynak. Ez egy megfigyelés az adatkészletből.

***Jellemző***: Egyetlen adatoszlopot nevezünk jellemzőnek. Ez egy megfigyelés összetevője, és adatpéldány attribútumának is nevezik. Egyes jellemzők lehetnek bemenetek egy modellhez, mások pedig kimenetek vagy a megjósolt jellemzők.

***Adattípus***: A jellemzők adattípussal rendelkeznek. Lehetnek valós vagy egész értékűek, vagy lehetnek kategorikus vagy sorszámértékek. Lehetnek karakterláncai, dátumai, időpontjai és összetettebb típusai is. Ezek tipikusan valós vagy kategorikus értékekre redukálódnak a hagyományos gépi tanulási módszerek használatánál.

**Címke**: A célváltozó vagy a címke az az érték, amelyet a modellünk megjósol. Például a mi esetünkben a matek dolgozat eredménye, az olvasási készség pont, vizsgán elért eredmény stb.

***Adatkészletek***: A példányok gyűjteménye egy adatkészlet, és amikor gépi tanulási módszerekkel dolgozunk, általában néhány adatkészletre van szükségünk különböző célokra.

***Képzési adatkészlet***: Adatkészlet, amelyet a gépi tanulási algoritmusunkba töltünk be modellünk betanításához.

***Tesztelési adatkészlet***: Olyan adatkészlet, amelyet a modellünk pontosságának ellenőrzésére használunk, de nem használjuk a modell képzésére. Nevezhetjük értékelési adatkészletnek.

***Validációs adatkészlet***: Az adatnak az a része, amelyet a modell gyakori kiértékeléséhez használnak. Illeszkedik a képzési adatkészletbe, valamint javítja az érintett híper-paramétereket (kezdetben beállított paraméterek a modell tanulás előtt). Ezek az adatok akkor játszanak szerepet, amikor a modell ténylegesen a tanulás fázisába van.



Előfordulhat, hogy példányokat kell gyűjtenünk az adatkészletek létrehozásához, vagy kapunk egy véges adatkészletet, amelyet fel kell osztanunk részadatokra.

**Az adatok tulajdonságai** [13]

***Méret***: Adatok skálája. A világ népességének és technológiájának a növekedése óriási mennyiségű adatot produkál minden másodpercben. A felhőalapú számítástechnikai, az Internet of Things (IoT) szolgáltatások, a mobilforgalomnak tovább növelik az adatok mennyiségét. Mostanra már az adat méreteinek redukálására kell helyezni a hangsúlyt.

***Változatosság***: Különböző adatformák - egészségügy, képek, videók, hangfelvételek. Strukturált, félig strukturált és strukturálatlan adatokra utal, különféle adatforrások, akár emberek, akár gépek generálják.

***Sebesség***: Az adatok halmozódásának sebességére vonatkozik. Ennek oka elsősorban az IoT, a mobil adat, a közösségi média stb.

***Érték***: Az adatok értelmessége olyan információk szempontjából, amelyekből a kutatók következtetni tudnak. Csak azért, mert rengeteg adatot gyűjtöttünk, nincs értéke, hacsak nem szerzünk belőle betekintést. Az érték arra utal, hogy az adatok mennyire hasznosak a döntéshozatalban. Megfelelő elemzéssel kell kinyernünk az adat értékét.

***Valódiság***: Az adatok minőségének, integritásának, hitelességének és pontosságának biztosítására utal. Mivel az adatokat több forrásból gyűjtik össze, ellenőriznünk kell az adatok pontosságát, mielőtt üzleti ismeretekre felhasználnánk őket.

## Adatelemzési lépések

**Alapvető statisztikai leírók** [12]

Ha gyorsan összefoglalni szeretnénk az összes olyan adatsort, amellyel dolgozunk, akkor az első mérőszám az ***adatok átlaga***. Bármely adat átlagos értéke az a szám, amely az ***összes adatpontot*** a legjobban ***képviseli***. Bármely adatpont átlaga lényegében az összes szám összege elosztva a darabszámmal.

Az átlaggal együtt azonban az adatokban meglévő ***variáció*** is fontos. A variáció annak mérése, hogy az adatpontok mekkora tartományba mozognak. Az ***adatok változása*** egyik fontos mértéke a tartomány, amely egyszerűen a maximális és a minimális érték levonása. Azonban a tartomány teljesen figyelmen kívül hagyja az átlagot, ezért gyakran alkalmaznak egy másik mérőszámot, amely a variancia.

Az adatok második legfontosabb mérőszám, amely összefoglalja az összes pontkészletet a ***variancia***. A variancia annak mérése, hogy az ***adat***ok hogyan ***változ***nak az ***átlag***tól. Röviden összefoglalják a számok halmazát, bármilyen számkészletet. A varianciával együtt egy másik kifejezés, amellyel találkozhatunk, a szórás.

A ***szórás*** nem más, mint a ***variancia négyzetgyöke***, és az adatok változásának mértéke. A szórás segít kifejezni, hogy egy adott adatpont milyen messze van az átlagtól.

Azokat a pontokat, amelyek ***három***nál több ***szórás***t jelentenek az átlagtól, gyakran ***kiemelkedő***nek tekintünk. A kiugró értékek eltérési küszöbértéke gyakran a felhasználási eseten alapul.

**Egyváltozós numerikus elemzés**

Az adatoknál csak ***egy változó***t veszünk figyelembe. Az egyváltozós adatok elemzése tehát a legegyszerűbb elemzési forma, mivel az információk csak egy mennyiséggel foglalkoznak. Nem foglalkozik okokkal vagy összefüggésekkel, és az elemzés fő célja az adatok leírása és a bennük létező minták megtalálása. Az egyváltozós numerikus értékek például: átlag, medián, percentilis és egyebek. [24]



Látható, hogy a barátokkal való szórakozás és a szabadidő egyváltozós elemzési értékek nagyon hasonlóak. A szabadidő és a barátokkal töltött idő átlagosan 3, azaz közepes érték felé tendál. A barátokkal való szórakozásnak kicsit nagyobb a szórása mint a szabadidőnek, mivel a szabadidő rögzítettebb.

**Kétváltozós numerikus elemzés**

Az ilyen típusú elemzésnél ***két*** különböző ***változó***t figyelünk. Az elemzés az ***ok***okkal és ***összefüggés***ekkel foglalkozik. Két jellemző közötti kapcsolat kiderítésére használják. [24]



Visszatekintve az előző példára, a szabadidő és a barátokkal töltött idő korrelációját nézem meg. Megfigyelhető, hogy pozitív összefüggés van a két jellemző között. Több barátokkal való időtöltés, több rendelkezésre álló szabadidőt jelenthet.

Fontos kiemelni, hogy az ***összefüggés nem jelent okozatot***. Ezt a ***korrelációs politika*** mondja ki. A mi esetünkbe nem feltétlenül a szabadidő miatt fog több időt tölteni egy diák a barátaival. Inkább a diák személyiségéből adódik, extrovertált vagy introvertált.

**Többváltozós numerikus elemzés**

Ha három vagy ***több változó***t elemzünk, akkor többváltozós elemzésről beszélünk. Hasonló a kétváltozóshoz, de több függő változót tartalmaz. Az elemzés módja az elérendő céloktól függ. Néhány technika a regresszióanalízis, az útelemzés, a faktoranalízis és a többváltozós varianciaanalízis. [24]



A regressziós analízis szerint gyorsan kiderül, hogy a szabadidő és a barátokkal eltöltött idő egyáltalán nem befolyásolja az érdemjegyet.

## Hogyan osszuk szét az adatokat?

**Az adatok szétválasztásának szükségessége** [25]

Az adatelőkészítés után az adatok szétválasztásával kell foglalkozni. Felmerül a kérdés, hogy miért kell szétválasztani az adatokat?

A képzési és tesztelési fázisok elvégzéséhez van egy adatkészlet. Az adatokat szét fogjuk osztani egy olyan ***képzési*** ***halmaz***ra, amely tanítani fogja az algoritmusunkat. A másik rész pedig, mint ***tesztkészlet*** lesz felosztva, amelyet az algoritmusunk tesztelésére használunk. A tesztelési eredményekből olyan ***mutatók***at kapunk, amelyek segítenek eldönteni, hogy egy adott modell hogyan teljesítene a mintán kívüli adatokon. Ezen mutatók alapján eldönthetjük, melyik modell lesz alkalmas a választott problémánkra. Azonban az egymillió forintos kérdés a gépi tanulásban az, hogy hogyan fogjuk felosztani az adatkészletünket?

**Előítéletes adat**

Két szempont van az adatok felosztásakor. Először is szeretnénk **randomizálni** az adatkészletünket, hogy a hasonló értékek ne oszlanak fel ugyanazon halmazra. Nem akarjuk megosztani az adatkészletünket úgy, hogy a férfiak vizsgáinak a pontszáma a képzési készletben legyen, míg az nők pontszámai a tesztkészletben. Ellenkező esetben ***előítélet***et (bias) vezethetünk be. [13]

**Rosszul illeszkedő adat** [6]

Másodszor van egy kompromisszum. Minél nagyobb a képzési készletünk, annál nagyobb a bizalom a modell tanulás iránt. Minél nagyobb a tesztkészletünk, annál nagyobb a bizalom a modelltesztelés iránt. Tehát az, hogy hogyan kell kettéválasztanunk, egy másik probléma. Miért nem képezük ki az egész adathalmazzal a gépi tanulási algoritmust?

Az ugyanazon az adatkészleten végzett képzés és tesztelés az úgynevezett ***túlillesztés*** (overfitting) eredményét fogja eredményezni. Ekkor egy modell inkább a ***tanító adatait*** kezdi megjegyezni, mintsem általánosítani.

Úgy is értelmezhetjük, mint egy olyan hallgató, aki ***megjegyzi*** az oktatóanyagokat, de nem képes általánosítani. Azokat a vizsga kérdéseket, amelyeket még nem látott nem tudja megválaszolni. Egyszerűen bemagolja a tananyagot a vizsgára, ami nem jó. De a gépi tanulási modellt és a hallgatót új, eddig nem látott adatokkal kell tesztelni, hogy biztonságosan megállapíthassuk, hogy megfelelően tanultak-e.

Mivel elkezdtük megvitatni a túlillesztést, be kell vezetnünk az ***alul illesztés***t a teljesség érdekében. Az alul illesztést akkor következik be, amikor a gépi tanulási algoritmusok nem tanultak eleget a bemeneti adatokból, ezért nem tudják megfelelően megbecsülni a jövőbeli értékeket.

Elképzelhetjük úgy, mint egy hallgató, aki ***nem tanult megfelelően*** a korábbi vizsga kérdéseit.

A **megfelelő illesztés**re törekszünk. Amikor a gépi tanulási algoritmus éppen annyit tanul, hogy a bemeneti adatokból ***megfelelően általánosít***son. Mégsem jegyzi meg a modellünket. Ekkor megjegyezhetjük, hogy gépi tanulás szépen követi a bemeneti adatok trendjét anélkül, hogy pontosan át kellene menni az összes adatpéldányon

A hasonlat itt egy olyan hallgató lesz, aki épp annyira tanulmányozza a vizsga kérdéseit, hogy sikeresen megértse a vizsga stílusát, stratégiáját és irányzatát, anélkül, hogy energiáját csak a tiszta memorizálásra fordítaná.



**Képzés és tesztkészletre felosztás**

Két módszer létezik az adathalmazunk elkülönítésére.

* Az első technikát ***képzés és teszt felosztás***nak nevezzük, ahol az adatokat két különböző képzés és tesztkészletre osztjuk.
* Míg a második technikát ***K-Fold keresztellenőrzés***nek nevezik, ahol a hiányokat csoportokra és foltokra osztjuk, és minden csoportot külön tanítunk.

Az első adatszegregációs technika, amelyet megvizsgálunk, a képzés és teszt felosztás. A képzési halmaz, amely egy nagyobb részt vesz az adatkészletünkből. A tesztkészletet, amely kisebb arányt vesz fel az összes adatból. A képzési teszt általában 70-80% -ot vesz igénybe az adatkészletünkből, míg a tesztkészlet általában 20-30% -ot vesz igénybe (***Pareto-elv***). [11]

**Diák adatok felosztása**

Vegyünk egy vizuális példát a képzés és teszt felosztás szakaszhoz és nézzük meg annak fő korlátját. Tegyük fel, hogy van a diák által tanulásra töltött ideje és az eredménye. Az általános tendencia az, hogy a tanulásra ráfordított idő növekedésével jobb eredmény érünk el. ***Kétféle felosztás***sal képeztem ki az adatkészletemet. Más-más tanulási és teszt kombinációval. Nézzük meg, hogyan különbözik az algoritmus teljesítménye! A következő képen látható a tanító adatok eloszlása.



A gépi tanulási algoritmusra egy egyszerű ***lineáris regresszió***s algoritmust választottam. A lineáris regressziós vonal megközelítőleg az adatsorunk közepén halad el, és megmondja, hogyan fog kinézni a jövőbeni előrejelzése. Az alábbi tesztelési adatkészlettel teszteljük a betanított gépi tanulási algoritmust.



A hibát a betanított gépi tanulási algoritmusunk által becsült tényleges érték és a tesztelési adatok közötti távolság határozza meg. Számos technika létezik egy betanított algoritmus hibájának kiszámításához. A ***hiba*** kiszámításának egyik technikája a hibák ***négyzet***ének ***összeg***e. A hibákat négyzetre osztjuk, így eltávolítjuk a negatív hibák hatását, mivel csak a hiba nagysága érdekel, nem pedig az iránya. Most pedig próbáljuk meg más módon felosztani az adatkészletünket.



Teszteljük az adatsort a második kombinációban. Jól észrevehető, hogy más lesz az értéke a hibaösszegnek.

Nagy különbség a hibaösszegek között nem ígéretes a modellünk szempontjából. Ez azt jelenti, hogy ***becslésünk pontossága csak szerencse kérdése***. Az, hogy hogyan osztjuk fel adatkészletünket tanító és tesztkészletre jelentős hatással van a modell pontosságunkra. Azt állítják, hogy ha a tanító és teszt felosztás nagy szórással rendelkezik, akkor nagyon érzékeny lesz a tanító adatainkra. Jobb megközelítés, ha minden adatsort felhasználunk mind tesztelésre, mind tanításra, majd kiszámoljuk az egyes csoportok átlagos pontosságát. Ez az úgynevezett K-Fold keresztellenőrzés.

|  |  |  |
| --- | --- | --- |
| Layer strúktúra: 20-5-1 loss=mse optimizer=adam epoch=20 | | |
| random\_state=1 | random\_state=7 | K-Fold |
| 0.8526168331028696 | 4.118124532287352 | -0.417124858945095 |
| -0.26620601712951025 | 7.671645684068228 | 0.5809101832495003 |
| -0.278504863682527 | 0.751627762065235 | -0.019804980479560252 |
| -0.25255267086571054 | 0.4909871237048553 | 0.4297972603282248 |
| 1.6396168902943211 | 11.628206612645016 | 0.25104169419165123 |

**K-Fold keresztellenőrzés**

A K-szeres keresztellenőrzés olyan képzési és tesztelési technika, amelynek célja az adatkészlet használatának optimalizálása a jobb és stabilabb tanítás pontosság elérése érdekében. A következő lépésekből áll [9]:

1. Adatkészletünket K csoportokra osztjuk. Az egyik csoportot tesztkészletként, a többi csoportot pedig edzéskészletként.
2. Képezzük és kiszámoljuk a modell pontosságát a választott képzési és tesztkészletek alapján.
3. Tesztkészletként a következő csoportot, tanítókészletként a többi csoportot választjuk, és megismételjük.
4. Miután kiszámoltuk az összes csoport pontosságát, a tanítórészekből kiszámítjuk az átlagos pontosságot.

**Diák adatok K-Fold keresztellenőrzéssel**

Lássuk ezt működés közben. Vegyünk példát egy négyszeres keresztellenőrzési képzéssel. K-értéket 4-nek választottam az egyszerűség kedvéért, de kiválaszthatnánk bármilyen értéket.



A modell általános pontossága az 1. pontosság, a 2. pontosság, a 3. pontosság és a 4. pontosság átlaga.

Az adatszegregáció alkalmazásakor kevés szempontot érdemes megemlíteni. Az adatszegregációs paraméterek kiválasztásakor két szabály van érvényben. A tanító és teszt felosztás használata esetén ajánlott az úgynevezett Pareto elv követése. Amely szerint az adatok ***20%*** a ***teszt***elésre és ***80%*** a ***tanít***ásra kell osztani. Szokták még 30% tesztelésre és 70% a tanításra is szétosztani. Míg a ***k-szeres*** keresztellenőrzés használatakor a ***10***-es k értéket ajánlják kísérletezéssel.

Van néhány hátul ütője, ha a K-szeres keresztellenőrzést választjuk. Igaz, hogy a K-szeres keresztellenőrzés valószínűleg képzettebb modellt eredményez a tanító és teszt osztott módszerhez képest. Ez azonban k-szer ***lassabb***an megy végbe, mivel a tanítást k-szer hajtjuk végre. A valóban jelentős időbeli különbség a nagyméretű adatoknál jelentkezik.

# Adattisztítási lépések

## Az adatokkal való problémák

**Adat előkészítési problémák**

Általában, ha adatelemzésről beszélünk, akkor elmondhatjuk, hogy az adatok előkészítése rendkívül hosszú ideig tart. Továbbá rendkívül fontos lépés. Ebben a fejezetben részletezni fogom, hogy milyen fontos ellátni jó adatokkal a gépi tanulási modellünket. Hiszen a híres mondás még mindig állja a helyét: „Szemét be szemét ki”.

Ha a gépi tanulási modellbe betöltött ***adatok rossz*** minőségűek, maga a ***modell*** is ***rossz*** minőségű lesz. Ezeket az adatokat a modell tanításához, valamint az előrejelzéshez is felhasználják. Semmilyen módon nem lehet jó gépi tanulási modellt felépíteni, függetlenül attól, hogy milyen algoritmust használunk, ha az adatai nincsenek megfelelően előkészítve. [26]

Rengeteg probléma merül fel, amikor a valós adatokkal dolgozunk. Lehet, hogy éppen ***nincs elegendő adat*** a modell tanításhoz. Lehetséges, hogy ***túl sok adat*** áll rendelkezésre, amelyek közül nem mindegyik releváns. Lehet, hogy ***nem reprezentatív adat***ok alapján tanítjuk a modellt. Tehát rossz tulajdonságokat táplálunk a gépi tanulási modellünkbe. Ezenkívül a valós világ adatai nyers ***adatok***, ***nem*** biztos, hogy mindig ***tiszták***. Előfordulhat, hogy ***hiányoznak*** a fontos mezők és ***rekordok***. Megtörténhet, hogy ***duplikátumok*** vannak az adatokban, amelyekkel foglalkoznia kell. Végül lehetséges, hogy az adatokat hibák tarkítják, ***kiugró értékek*** vagy hiányzó adatok formájában. Tehát egyértelműen vannak problémák az adatokkal.

Hogyan kezeljük ezeket?

### Nem elegendő adat

Olyan modelleknél, amelyek nem elegendő adatokkal vannak kiképezve gyengén teljesítenek az előrejelzésben. Tehát ***nincs*** annyi ***minta***, amelyből a modell tanulhatna. Ha csak néhány rekord van a gépi tanulás modellhez, az kétféleképpen végződhet. [13]

* ***Túlillesztés***: Ilyenkor a modell túl sokszor olvas be a kevés adatból és egyszerűen megjegyzi az adatokban meglévő mintákat. A modell ***memorizál*** nem pedig általánosít.
* ***Alul illesztés***: Lehetséges, hogy kevés adat esetén a modell alul illeszt az adatokon. Itt a modell túlságosan leegyszerűsödhet, ami azt jelenti, hogy ***nem*** igazán ***értette meg*** az adatokban létező mintákat.

A nem elegendő adat problémája a valós projektek során gyakori küzdelem. Megeshet, hogy a releváns adatok egyszerűen nem állnak rendelkezésre. Ha mégis előállnak az adatok, akkor gyűjtésének tényleges folyamata időigényes és rendkívül nehéz. A modellek tanításához szükséges adatok hiányával küzdés során nincs nagyszerű megoldás. Egyszerűen ***több adatforrást kell találni***. Várni kell tovább, amíg az adatok előálltak vagy saját kézzel kell begyűjteni azokat.

Van néhány megközelítés, amellyel megoldhatjuk a kis adatmennyiség problémáját. A technika, amely részletezésre kerül, nem minden esetben alkalmazható széles körben. Mit tegyünk, ha nincs elég adatunk?

* ***Modell leegyszerűsítés***: Választhatunk egy egyszerűbb modell felépítését. Az egyszerűbb modellek kevesebb adat mellett jobban működnek.
* ***Átviteli tanulás (transfer learning)***: Ha neurálishálózatokkal vagy mély tanulási technikákkal dolgozunk, akkor használhatjuk az átviteli tanulást. Egy előre felépített modellt használunk, amelyet aztán módosítunk a meglévő kis adatkészleten.
* ***Adatbővítés***: Megpróbálhatja növelni az adatmennyiséget, amellyel dolgozik, az adatbővítési technikák használatával. Ezt a képadatokkal meglehetősen gyakran használják. Egyszerűen módosítja a meglévő képeket, hogy új képeket kapjon.
* ***Adatszintézis***: És még egy utolsó lehetőség lehet: ha megértettük az adatok típusát, statisztikai tulajdonságát, akkor ezeket felhasználhatjuk szintetikus (mesterséges) adatok előállításához.

**Modell leegyszerűsítés** [6]

Minden gépi tanulási algoritmusnak megvan a saját ***paraméterkészlet***e. Figyelembe kell venni, hogy milyen modellt használunk és mennyi adat elérhető. Ha ***kevesebb adat*** van, akkor ***egyszerűbb modell***t érdemes választani. Az egyszerűbb modellek kevésbé hajlamosak az adatok túlillesztésére és az adatok mintázatának memorizálására. A túlillesztésre hajlamos modellek jól teljesítenek az tanító adatkészleten, de a való világban gyengén.

Egy másik lehetőség az, hogy együttes tanítás (***ensemble training***) alkalmazunk a kis adatkészleten. Az együttes tanítás nem támaszkodnak egyetlen gépi tanulási modellre, ***több egyéni modell***t tanítanak és a modell végső előrejelzése az egyes modellek ***összesített előrejelzése***. Mivel az együttes tanítás sok egyéni előrejelzés eredményeit összesítik, gyakran jobb teljesítményt nyújtanak, mint az egyéni gépi tanulási modellek. Például logisztikai regresszióra, Random Forest, Naiv Bayes osztályozóra stb. Összevonjuk az egyes modellek eredményeit, hogy megkapjuk a az együttes végeredményt.



**Átviteli tanulás**

Ha neurális hálózatokkal dolgozunk és nincs túl sok adata a modell betanításához, akkor az átviteli tanulás egy lehetőség. Az ***transzfertanulás*** azonban csak ***gyakori*** használati ***esetek***re alkalmazható. Az átviteli tanulás magában foglalja egy képzett neurális ***hálózat újra felhasználás***át, amely megoldja a hasonló problémát.

Például képosztályozásnál szükség van egy modellre, amely képes képosztályozásra. Újra felhasználjuk az architektúrát, valamint a modell paramétereit és egyszerűen elvégezünk egy kis tanítást. Átveszük a modell részét képező ismereteket, majd betanítjuk a modellt az új adatkészletére. Az adatkészlet nem feltétlenül hatalmas, de a modell magában foglalja azt az ismeretet, amelyet az eredeti adatkészletből nyert, amely általában nagyobb. Majd a tanítás után a modell illeszkedik az új adatokra is.



**Adatbővítés**

Térjünk át a következő technikára, amelyet alkalmazhatunk, ha nincs elegendő adatunk: az adatok bővítése. Ilyenkor a meglévő mintákat veszünk és valamilyen módon megváltoztatjuk. A ***módosítás során új minták***at jönnek létre. Az adatbővítési technikák lehetővé teszik a tanító adatkészlet számának növelését. Általában képadatokhoz használják. Új képek előállításához az eredeti képeket kell manipulálni. Például zajosíthatjuk, méretezhetjük, forgathatjuk. Ezekkel további mintákat is lehet készíteni, amelyekkel taníthatjuk a hálózatunkat.

**Adatszintézis** [11]

Az utolsó lehetőség a szintetikus, mesterséges adatok előállítása. A szintetikus adatok saját problémakörrel járnak. Alapvetően mesterségesen előállított mintákat ***utánozzák a való világ adatait***. Ennek érdekében meg kell értenie az adatok jellemzőit, amelyekre jó modellek felépítéséhez van szükség. Túlmintázhatjuk a meglévő adatmintákat, hogy új adatokat kapjunk. Lehetnek néhány buktatók is. ***Bevezethetünk*** az adatkészletbe a való világban nem létező ***előítéletet***, ami előítéletes modellt eredményez.

### Túl sok adat

A szervezetek tudják, hogy fontosak az adatok, viszont gyakran nem tudják, milyen adatok lesznek fontosak. Végül túl sok adatot gyűjtenek be. Furcsának tűnhet, hogy a túl sok adat is problémát jelent. Az adatok, amelyekkel dolgozunk, kétféleképpen lehetnek túlzottak.

**Dimenzió átka** [6]

Az első a dimenzió átka. A sok dimenzió az jelenti, hogy a példányaink az adatkészletben **túl sok jellemző**je van. Más szóval az adatrekordoknak túl sok oszloppal rendelkezik. Ha túl sok dimenzióval rendelkezik az adatunk, akkor lényegtelen jellemzők vannak. Nem fontos jellemzők nem igazán segítenek a modellek tanításában.

A dimenzió átka hatalmas téma, amelyet az adattudósok részletesen tanulmányoznak. Két konkrét probléma merül fel, ha túl sok dimenzió áll rendelkezésre.

* Az első az, hogy valamilyen módon el kell döntenie, hogy mely adatok relevánsak valójában. Ez magába foglalhatja a jellemzők statisztikai technikákkal történő kiválasztását vagy a dimenziócsökkentést. Alapvetően ***oszlop***ok ***megszűntetés***éről lenne szó.
* Lehetséges, hogy az adatok túlságosan szemcsések ahhoz, hogy az ML modellek felvegyék a mintákat. Ilyenkor a nagyon alacsony szintű adatokat hasznos jellemzőkbe kell összesítenie. Tehát ***oszlop***okat ***összevonni*** értelmes módon.

**Elavult adatok** [26]

Lehetséges, hogy elavult történelmi adatokkal kell dolgozni. Túl sok minta vagy rekord van, amelyek közül sok lényegtelen. A történeti adatokkal való munka egyfajta kétélű kard. Mennyire fontosak a történelmi adatok? Ha túl sok olyan történelmi adata van, amely nem igazán jelentős, akkor találkozhatunk az úgynevezett ***koncepció áramlással***. Az idő múlásával a bemeneti adatok (X) és a címkék (Y) között változik a kapcsolat. A gépi tanulási modellek nem tudnak lépést tartani az új bemenetekkel, mert túl sok történelmi adat van. Következésképpen a modell teljesítménye hanyatlik. A koncepció áramlás lényegében azt jelenti, hogy gépi tanulási modell folyamatosan ***figyeli*** ***az elavult***, már nem jelentős ***adatrekordokat***. Az elavult tudás felhasználva nem képes jó előrejelzésekre az új mintákra. Tehát, az elavult történelmi adatokkal körültekintően kell használni. A történelmi adatok fontosak, de valamiféle ítéletet kell használnia, hogy kiderítse, valójában milyen rekordok relevánsok.

**Megoldási technikák** [6]

Elavult történelmi adatokkal elég nehéz foglalkozni. A dimenzió átka könnyebben megoldható probléma.

* ***Jellemző kiválasztás*** (feature selection): Statisztikai technikák segítségével ***kiválaszt***juk azokat a jellemzőket, amelyek relevánsak számunkra.
* ***Jellemző tervezés*** (feature engineering): A jellemző tervezéssel ***összevon***hatjuk az alacsony szintű és részletezett jellemzőket olyan hasznos jellemzőkbe, amelyek kevésbé szemcsések. Kombinálhatjuk a funkciókat is, hogy javítsuk a prediktív erejüket.
* ***Dimenzió csökkentés***: Elvégezhetünk dimenziócsökkentést. Itt csökkenthetjük adatainak bonyolultságát anélkül, hogy elveszítenénk az információkat. Ennek egyik módja az, ha adatokat ***új tengely mentén*** kerül átirányításra. Ezáltal elérjük az adatokban meglévő maximális varianciát.

Amikor ilyen jellegű fejlesztést végezünk az adatok összetettségének csökkentése érdekében, akkor találkozhatunk a ***koncepció hierarchia*** kifejezéssel. Ez egy olyan leképezés, amely nagyon ***alacsony szintű jellemzők***et kombináljuk ***általánosabb***an használható ***jellemzőkbe***. A koncepció hierarchia magában foglalja az adatok gyűjtését is, hogy az információk kevésbé legyenek túl részletezett formátumban.

### Nem reprezentatív adatok

A nem reprezentatív adatoknak számos megnyilvánulása van. Az egyik az, hogy csak ***rossz tulajdonság***okat táplálunk be a modellbe, de vannak más megnyilvánulásai is. Lehetséges, hogy az összegyűjtött adatok hibákat tartalmaznak. Bizonyos szempontból pontatlan és a hibák olyanok, hogy jelentős hatással lehetnek a modellre. Feltétlenül fontos az adatok tisztítása és feldolgozása azért, hogy az adatok jó állapotba kerüljenek.

Előfordulhat, hogy az adatok nem reprezentatívak, mert előítélettel rendelkeznek. Tegyük fel, hogy öt érzékelőből gyűjtünk adatokat, öt különböző országban, és van egy olyan érzékelő, amely nem működik mindig. Az adatok előítéletesek, mivel nincsenek arányos adatai az egyik érzékelőtől. Ha előítéletes adatokkal dolgozunk, akkor az előítéletes gépi tanulási modellt eredményez. Ezek a modellek a gyakorlatban gyengén teljesítenek, mivel nem tartják szem előtt az összképet. Enyhíthetünk a túl- és az almintázás segítségével. Tehát, ha kevesebb adata van az egyik érzékelőtől, akkor felülmintázhatjuk azokat az adatokat. Így reprezentatív mintát érünk el. A túl- vagy az alul mintavétel saját elfogultságot eredményezhet, ezért erre ügyelnie kell. [6]

### Hiányzó, kiugró értékek

További problémák, amelyekkel találkozhatunk, az adatok hiánya és a kiugró értékek jelenléte. Az adattisztítási eljárások jelentősen mérsékelhetik mind a hiányzó adatok, mind a kiugró értékek hatását. Egy későbbi fejezetben részletezem azokat a konkrét technikákat, amelyekkel kezelhetjük a hiányzó adatokat és a kiugró értékeket.

### Másolatok

Ha adatokat gyűjtünk, előfordulhat, hogy vannak duplikátumok. Ha az adatokat ***másodpéldányként*** lehet ***megjelölni***, a problémát nagyon könnyű megoldani. Egyszerűen csak másolatot kell alkalmaznia az adatokra, mielőtt betáplálnánk őket egy modellbe, de a világ nem ilyen egyszerű. A másolatokat nehéz lehet azonosítani bizonyos alkalmazásokban, konkrétan a valós idejű streaming alkalmazásoknál. Mivel az idő folyamatosan változik nem tudjuk érzékelni a másolatokat.

## Hiányzó adatok kezelése

Adatok gyűjtése és azokkal való munka során előfordulhat, hogy hiányzó adatok vannak a mezők hiányzó értékei formájában. Illetve olyan kiugró értékeket tartalmazhat, amelyeknek valójában nincs sok értelme. Kétféle megközelítést követhetünk, mikor a hiányzó adatokról beszélünk. [20]

* ***Törlés***: Ezek közül az első a törlés, ahol megszabadul az adatoktól, amelyek hiányzó mezőket tartalmaznak,
* ***Kitöltés***: A másik a kitöltés, ahol a hiányzó értékeket valamilyen technikával kikövetkeztetjük, majd beírjuk a hiányzó helyre.



Először beszéljünk a törlésről. A törlést listás törlésként is nevezik. Itt egy ***teljes rekord*** kerül ***törlés***re, amely megfelel az adatkészlet egy sorának. Ha hiányzik egyetlen egy érték egy jellemzőből (oszlop), akkor az egész sor törölni kell. Ez egy egyszerű, problémamentes technika a hiányzó értékektől való megszabaduláshoz. Ugyanakkor ***előítélet***hez vezethet, mert megszabadulunk egy teljes rekordoktól, még akkor is, ha csak egy irreleváns mezőnek hiányzik az értéke. A listán belüli törlés a gyakorlatban a legelterjedtebb módszer, mert ***egyszerű***, viszont ez több problémához is vezethet.

***Nagymértékben csökkenhet*** a ***minta méret***e. Ha nincs sok rekord, akkor olyan helyzetbe kerülhet, hogy nincs elegendő adat a gépi tanulási modell betanításához. Például egy jellemző értékei mind hiányoznak, mert az tegyük fel egy másik érzékelő jegyzi fel azokat. Miután kidobunk minden rekordot a bizonyos érzékelőről az jelentős torzítást eredményez.

Tehát világos, hogy a teljes rekordok eldobása, amelyekből hiányzik néhány mező, nem nagyszerű megoldás, ezért a ***kitöltés***re térünk át. Ilyenkor a hiányzó oszlopértékeket egészítjük ki, ahelyett, hogy a rekordokat törölnénk. A hiányzó értékek a már rendelkezésre álló ***adatokból*** lehet ***következtet***ni. Számos módszer közül választhatunk, amelyek a nagyon egyszerűtől a nagyon összetettig terjednek. [20]

* ***Oszlop átlag***: A lehető legegyszerűbb módszer az oszlopátlag használata. A hiányzó érték lényegében közelít az oszlopban szereplő átlagértékkel. Mivel az átlag írja le legjobban az adott adatkészletet.
* ***Oszlop medián***: További nagyon hasonló lehetőségek az adott oszlop mediánértékének használata.
* ***Interpoláció***: A hiányzó értékek beszámításának másik módja az interpoláció más közeli értékekből. Ez a technika akkor hasznos, ha a rekordokat valamiféle eredendő sorrendben rendezik.

**Egyváltozós kitöltés**

Az egyváltozós kitöltés csak ugyanazon tulajdonság vagy ugyanazon oszlop ismert értékeire támaszkodik.

**Többváltozós kitöltés**

A többváltozós kitöltés viszont minden ismert adatot felhasznál, amire az adatokból hiányzó értékekre kell következtetnie. Például érdemes az adatok más oszlopaiból regressziós modelleket készíteni, hogy előre jelezze az adott oszlop hiányzó értékeit. Ez egy példa a többváltozós kitöltés. Ezt iteratív módon megismétli az összes oszlopban, amelyek azonos értékeket tartalmaznak.

**Kritérium alapú kitöltés**

Vannak más technikák, amelyek a hiányzó értékek kitöltésére is alkalmazhatók, például a kritérium alapú kitöltés. Az összes bejegyzést fontos ***kritérium***ok ***alapján*** rendezük. Minden hiányzó értéknél felhasználhatjuk közvetlenül a kritérium szerinti előző értéket. Ezt az ***utoljára átvitt megfigyelés***nek nevezzük. Tehát a rekordok rendezése után kitöltjük a hiányzó értékeket az előző értékkel. Ez különösen hasznos az idősoros adatoknál, ahol az időbeli előrehaladásnak van értelme. [20]

**Átlag helyettesítés**

Egy általános technika, amelyet gyakran használnak egyváltozós kitöltésre, hogy minden hiányzó érték helyett az összes rendelkezésre álló érték átlagát helyettesíti. Az átlagos helyettesítés gyengíti az összefüggéseket az adatok oszlopai között. Amikor lényegében azt mondod, hogy ez egy átlagos adatpont, akkor nincs semmi különös, ***gyengül a korreláció***. Ez problematikus lehet, ha kétváltozós elemzést végzünk két változó kapcsolatának meghatározására. [20]

**Hiányzó adat előrejelzés**

Ha intelligensen módon meg akarjuk jósolni az adatok hiányzó értékeit, érdemes egy gépi tanulás modell segítségéhez fordulni. A modell ***előre jelzi*** a hiányzó oszlopokat ***más oszlop***értékek ***alapján***. Ennek a technikának az alkalmazása megmutatja az adatokban meglévő összefüggéseket. Lényegében látni fogjuk, hogy ez az oszlop függ a többi oszloptól. Így a regressziónak és az átlagos helyettesítésnek gyengítő hatásai vannak. Tisztában kell lennie a technikák alkalmazásának árnyalataival, és meg kell határoznia az adott felhasználási esetet.

## Kiugró értékek kezelése

Az adatainak feltárása és vizualizálása során előfordulhat, hogy az adatkészletben vannak kiugró értékek. Ez egy olyan adatpont, amely jelentősen eltér az ugyanazon adatkészlet többi adatpontjától. Előfordulhat, hogy a teljes rekord valamilyen módon kiugró érték, vagy vannak olyan mezők, amelyek kiugró értékekkel rendelkeznek.

Két lépést tudunk elkülöníteni, mikor kiugró adatokkal dolgozunk. Az első lépés az adatokban létező ***kiugró értékek azonosítása***. A második lépés ezeknek a ***kiugró értékek***nek a ***kezelése***.

Csakúgy, mint vannak gépi tanulási algoritmusok, vannak olyan speciális algoritmusok is, amelyeket a kiugró értékek észlelésére építettek. A legalapvetőbb azonosítása a kiugró értékeknek, hogy megmérjük az adott adatpont távolságát az adatok átlagától. Másik módszer, hogy vesszük az adatpont és a modell által illesztett vonal távolságát. Miután azonosítottuk a kiugró értékeket, három tág technika segítségével megbirkózhatunk a kiugró értékekkel. Eldobhatjuk a kiugró adatokat, korlátozhatja a kiugró értékeket vagy beállíthatja a szélső értékeket az átlagos értékre. [20]



**Észlelés**

Kezdjük azzal, hogy megbeszéljük, hogyan tudjuk azonosítani a kiugró értékeket a két megvitatott technika segítségével.

**Távolság az átlagtól**

Az adatok bármely jellemzőjének átlaga alapvetően a központi irányvonal mérőszáma. Ez az a pont, amely körül a fennmaradó pontok csoportosulnak. Egy adatpont, amelynek értéke messze van az átlagtól, azt kiugrónak tekinthető. [27]

**Távolság az illesztett vonaltól**

Az adataiban szereplő kiugró értékek azonosításának másik módja az, ha megmérjük azok távolságát a ***modell által illesztett vonal***tól. Tegyük fel, hogy van egy kétdimenziós adathalmazunk. Ebben az esetben a modellünk megpróbál egy sort illeszteni valamilyen regressziós algoritmus segítségével. A kiugró értékek lényegében olyan adatpontok, amelyek távol vannak az illesztett vonaltól vagy görbétől.

**Kezelés** [27]

Miután azonosítottuk a kiugró értékeket, ki kell találni, hogyan akarunk velük megbirkózni. Igazából nincs konkrét megoldás, számos feltétel szabadhatja meg a kiugró adatok kezelését. Meg kell vizsgálnia és meg kell értenie az adatokban található kiugró értékeket. Ezek a kiugró értékek helytelen megfigyelések vagy az adatokban előforduló hibák miatt vannak? Ekkor érdemes lehet teljesen ***megszabadulni***a ettől ***a rekordtól***, ha a rekord összes attribútuma hibás.

Ha egy sorra vagy egy rekordra vonatkozik, csak egy attribútum van tévesen rögzítve, akkor érdemes ezt a kiugró értéket ***átlagnak beállítani***, és nem kidobni a rekord egészét.

Feltéve, hogy modellt nem torzítják a kiugró értékek jelenléte, akkor hagyjuk úgy, ahogy van.

A kiugró értékek fontos információkat közvetítenek, amelyeket a modelljének esetleg fel kell ismernie. Ehhez szükség lehet arra, hogy először standardizálni az adatokat. A ***standardizálás*** az összes adatpontot kifejezze az átlagtól való eltéréssel. A standardizálás magában foglalja az összes értékből az átlag levonását, így az eredményül kapott ***érték skálázott***. A felső kiugró értékeket pozitív három standard eltérésként határozhatjuk meg az átlagtól. Az alsó kiugró értékeket negatív három standard eltérés lehet.

## Numerikus és kategorikus adatok

Az adatokat két nagy kategóriába lehet sorolni, a kategorikus és a numerikus adatok. Mi a különbség a kettő között?

**Kategorikus adatok** [20]

A kategóriák a megengedett értékek véges halmaza. A diszkrét értékekből vagy kategóriákból álló adatok további két alkategóriára oszthatók, a ***nevesített*** és a ***sorrendi adatok***ra. A sorrendi adatok rendezhetők a nevesített adatok, pedig nem.

A kategorikus adatokat numerikusan kódolni kell, mielőtt az modellekben felhasználható lenne. Két konkrét technika a címkekódolás és az egycsatornás kódolás.

A kategorikus adatok, amelyek kategóriákból állnak például: férfi, nő, az év hónapja, a hét napja.

A kategorikus értékek osztályozásra használható a K-Nearest Neighbor.

**Numerikus adatok** [20]

A numerikus adatok tetszőleges értéket vehetnek fel egy végtelen tartományból. Amikor a numerikus ***adatok különböző skálán*** vannak, akkor nem igazán működnek jól a gépi tanulási modellek. Különféle technikák segítségével méretezhetjük a numerikus adatokat.

A numerikus értékek mindig nagyságrend szerint ***rendezhetők***, sajátos rendezéssel vagy rangsorolással rendelkeznek. Ezért a numerikus értékeket ***folyamatos*** adatoknak is nevezik. Például a súly (kg), a hőmérséklet (Fahrenheit), a lakásárak.

Numerikus adatok a folyamatos értékek előrejelzésére használható, mint a regressziós modellek.

Minden más típusú adatot, a szöveges adatokat, a kép adatokat, át kell alakítani ezen adattípusok egyikévé.

## Adattisztítás megvalósítása

### Hiányzó értékek kezelése

Az alábbi bekezdésbe egy olyan adatsorral fogunk foglalkozni, amelyre adattisztítási technikákat kell alkalmazni. Az ***adatsor*** kezdeti formába ***nem áll készen*** a gépi tanulásra. Megtisztítjuk ezt az adatsort, majd feltárjuk, mielőtt illesztenénk egy gépi tanulási modellt.

Először be kell olvasni be az adatainkat tartalmazó csv fájlt. Adataim a **2019\_PublicDataFile\_Students.csv** fájlban találhatók az adatkészletek mappában. Az adatkészlet eredeti forrás .gov oldalán található. Több mezőben ***hiányzó értékek*** találhatóak.

Ha megnézzük a betöltött adatkészletet, akkor láthatjuk, hogy ***11 jellemző***ből és ***1829 rekord***ból áll. Meg kell állapítani meg a hiányzó értékeket, majd összegezni azokat. Az egyik jellemző összes rekordja üres, ezért azt kidobjuk.



Látható, hogy több mint több mint 1000 üres mező van. Különböző technikák segítségével kezelhetjük ezeket a hiányzó értékeket. Ha elegendő adat áll rendelkezésre, akkor az ***eldobás*** egy teljesen érvényes lehetőség. Figyelembe véve az oszlopok hiányzó értékeinek számát a „Trust Score” jellemzőnek van a legkevesebb üres mezője. Ráadásul ez a jellemző függ legjobban az összes többi oszloptól. Továbbá ez egy célváltozó, egy jellemző, ami a végeredményt reprezentálja. Tehát itt egyszerűen eldobom az üres mezőket tartalmazó rekordokat.

Egy másik érvényes stratégia, hogy minden hiányzó mezőt helyére az átlagot helyettesítem be. Ami a „Strong Family-Community Ties Score” oszlopot illeti, az összes hiányzó értéket az adatok jelenlegi átlagával fogom pótolni. Mivel nincs nagyon sok hiányzó mező, így az ***átlag***gal való ***kitöltés*** kézenfekvő. Hasonló módon jártam el a „Collaborative Teachers Score” és az „Effective School Leadership Score” jellemzőknél.

A „Rigorous Instruction Score” oszlop esetén a ***leggyakrabb***an szereplő ***érték***et helyettesítettem be.

Kifinomultabb megközelítés az ***IterativeImputer*** osztály használata, amely minden tulajdonságot hiányzó értékekkel modellez a ***több***i ***jellemző*** ***függvényében***, és ezt a becslést használja a kitöltéshez. Round-robin algoritmus szerint iterál egy megadott ismétlés számig. [28]

Az üres értékek eldobása és kitöltése után 1694 rekord van, szemben a kezdeti 1829 rekorddal. Tehát ***135 rekord került ki*** az adatkészletből, ami nem annyira számottevő. Ennek ellenére nullára csökkentettük az üres értékek számát.



### Címkekódolás és egycsatornás kódolás

Hogyan lehet konvertálni a kategorikus adatokat numerikus formába a címkekódoló és az egycsatornás kódoló segítségével? Egy másik diák adathalmaz ***kategorikus adat***ok átalakítása következik. A konkrét adatkészlet eredeti forrása az alábbi URL-címen található: <https://www.kaggle.com/janiobachmann/math-students>.

Ha megnézzük ezt közelebbről, látható, hogy számos kategorikus jellemzővel rendelkezik. A nem, az internet hozzáférés, a család mérete. Az adatkészlet ***395 rekord***ból és ***33 jellemző***ből áll.

**Címkekódolás** [29]

Vannak olyan kategorikus adatok, amelyek két diszkrét értéket vehetnek fel. Ilyen például: nem, családi támogatás, romantikus kapcsolat.

Ezeket egyszerű címkekódolással átalakítom numerikus adattá. A ***LabelEncoder*** osztály segítségével könnyedén transzformálhatók az adatok. Újra felhasználhatóság érdekében egy függvénybe szerveztem ki a címkekódolást. A címkekódoló ***0***-val és ***1***-gyel helyettesíti a ***kategóriák***at. [28]



**Egycsatornás kódolás** [9]

Egy másik kategorikus jellemző, a ’gyám’ (guardian). Ha megnézünk egy mintát, látható, hogy a gyám lehet apa, anya vagy egyéb. Ha az adatok kategóriáiban nincs implicit sorrend, akkor érdemes lehet egycsatornás kódolással használni (***one-hot encoder***). Ilyenkor különböző oszlopokba kerülnek a kategóriák, a mi esetünkbe: guardian\_father, guardian\_mother, guardian\_other. Ha a diák gyámja az apja házas, akkor a guardian\_father oszlop alatt 1-es érték és az összes többi oszlop értéke 0 lesz. Ezt követően az eredeti ’gyám’ oszlopot eldobhatjuk az adatkészletből. Majd beillesztjük a kódolt változatott.

Egyszerűbb módja az egycsatornás kódolásnak, ha a panda keretrendszerbe létrehozott „***get\_dummies***” függvényt használjuk. Csak át kell adnia az eredeti adatkeretet, és meg kell adnia azt az oszlopot, amelyet egyszerre szeretne kódolni. Ha nem szeretnék az összes kategorikus értéket egycsatornás módon kódolni, akkor elég, ha csak az adatkeretet adjuk át. Feltérképezi, majd hozzáadja a kódolt oszlopokat az adatkészlethez. A függvény gondoskodik az eredeti kategorikus oszlop eltávolításáról is. [30]



# Diák teljesítmény előrejelzése

Az egyetemek ma, hasonlóan az üzleti szervezetekhez, igen dinamikus és kompetitív környezetben működnek. Az egyetemek gyakran kénytelen ***gyors*** és fontos ***döntések***et meghozni, ennek érdekében jó minőségű információkra van szükségük. Fejlett információs technológiákat kell bevezetni a rendelkezésre álló adatok tudássá való hatékony átalakításához és a döntéshozatal támogatása érdekében.

Megfelelő döntésekhez pontos gépi tanulási algoritmusra van szükségünk. Honnan tudjuk melyik a legjobban illeszkedő modell az adott problémára? Vegyünk szemügyre néhány gépi tanulási algoritmus teljesítményét.

## Osztályozó modellek összehasonlítása

Az International Journal of Computer Science tanulmányon keresztül szeretném bemutatni a tanulási algoritmusok teljesítményét. A kutatási célokra használt adatkészlet az egyetemre három egymást követő évben felvett hallgatók adatait tartalmazza. Számos jól ismert osztályozási algoritmust, köztük egy szabálytanulót, egy döntési fa osztályozót, egy neurális hálózatot és egy legközelebbi szomszéd osztályozót alkalmaznak az adatkészleten. Adathalmaz 10067 hallgató tartalmaz 14 paraméterrel jellemezve. Az előre jelzett változónak két különálló értéke van, amelyek megfelelnek annak a két osztályba sorolják be a tanulókat: Gyenge és Erős.

A legnagyobb pontosságot a:

* neurális hálózat modellnél érik el (73,59%),
* amit követ a döntési fa modell (72,74%),
* majd a k-NN modell (70,49%).

A neurális hálózat modell az egyetlen modell, amely az „Erős” osztályt nagyobb pontossággal jósolja (igaz pozitív arány = 77%), mint a „Gyenge” osztályt (igaz pozitív arány = 70%). Tehát ezt a modellt lehetne a legsikeresebben használni az erős hallgatók megjóslására. Ennek a modellnek a hátrányai a bonyolultsága, valamint nehéz megértése és értelmezése. A másik három algoritmussal generált modellek nagyobb pontossággal jósolják a „Gyenge” osztályt, mint az „Erős” osztály. Felhasználhatók a veszélyeztetett tanulók korai azonosítására, amelyek további támogatásra szorulhatnak. A Döntésfa osztályozási modell a „Gyenge” osztályt (igaz pozitív arány = 75%) nagyobb pontossággal jósolja meg. Ennek a modellnek az az előnye, hogy könnyen értelmezhető, mert érthető szabályok halmazát állítja elő, és jól működik mind a nominális, mind a numerikus változókkal. [31]



2. ábra Az osztályozási modellek értékelésének eredményei [31]

## Online kurzusok sikeresége

Az oktatás már nem egyszeri esemény, hanem egész életen át tartó folyamat. Míg a klasszikus oktatási modell nem felel meg ezeknek az igényeknek, az onlinekurzusok új lehetőségeket adnak számunkra. Az online tanfolyamok világában a hallgatói létszám jelentős növekedése lehetetlenné teszi, hogy egyéni értékelés legyen.

Ebben az esetben segíthet egy automatizált rendszer, amely a ***hallgató***k valós idejű ***teljesítmény***ét ***jósol***ja meg. Értékes eszköz lenne jó döntések meghozatalához arról, hogy mikor kell élő oktatási beavatkozásokat végrehajtani, az elkötelezettség növelésének és ***motiváció biztosítás***ának céljából.

**Jellemző kinyerésének problémája**

Az egyik legnehezebb probléma, hogy kinyerjük az értékes jellemzőket az adatunkból.

* A hagyományos megközelítés elsősorban általánosított ***lineáris modellek***re támaszkodik. Ezek modellek a nyers adatokból kinyert különféle ***jellemzőkre támaszkodnak***.
* Az új megközelítés a ***neurális hálózatok*** feltárását foglalja magában. Az új modellek eddig primitív teljesítményt mutattak. Ez elsősorban azért van, mert a modellek továbbra is a ***jellemzők*** tervezésére ***támaszkodnak*** a bemeneti dimenziók csökkentése érdekében.

Ennek problémának a kiküszöbölésére a GritNet neurális háló architektúra nyújthat segítséget. [32]

**GritNet architektúra**

A tanulói teljesítmény előrejelzésének feladata szekvenciális esemény előrejelzés problémaként fejezhető ki. A múltbeli események sorozatából becsüljük meg a jövőbeli események valószínűségét. A GritNet célja a hallgatók eredményének becslése.

A diákok nyers adatainak betáplálása érdekében, az időbélyeggel ellátott rekordokat rögzített hosszúságú bemeneti vektorba kell alakítani. Egyszerű egycsatornás kódolással (one-hot encoding) transzformálható a tanuló időbélyeggel kapcsolatos tevékenységei (mentor interakció, kurzus teszt, tanvideó nézés). A változó tanulási sebesség fontos időfüggő információ, ami tükrözi a hallgató előrehaladását és a tananyag tartalmának nehézségeit.

A GritNet modell 4 főbb rétegre bonható fel [32]:

1. Beágyazott réteg: Mivel az egycsatornás kódolás nem helyez hasonló elemeket közelebb egymáshoz a vektortérben szükség van egy beágyazott rétegre. Megtanulja a beágyazott mátrixot és átalakítja a bemeneti vektor egy alacsony dimenziós reprezentációra.
2. BLSTM (Bidirectional LSTM) réteg: Két irányú LSTM-ből áll, az egyik a bemenetet előre, a másik pedig hátrafelé veszi. A kimeneti vektorok az előre és hátra irányúló kimenetek összefűzésével jönnek létre. A BLSTM-ek hatékonyan növelik a hálózat számára elérhető információk mennyiségét, javítva az algoritmus számára elérhető kontextust. Mivel képes a múlt és a jövő információinak megőrzése.
3. GMP réteg: A GMP rétegekkel a GritNet megtanul összpontosítani az leglényegesebb eseményekre, a többit figyelmen kívül hagyva. Ez a GMP-művelet döntőnek tűnik a jóslási teljesítmény növelésében, különösen a kiegyensúlyozatlan adatok esetén.
4. Fully-connected réteg: A GMP réteg kimenetét végül egy teljesen összekapcsolt rétegbe és egy softmax (pl. sigmoid) rétegbe vezetik be, hogy kiszámítsuk a diák jövőbeli tevékenységeinek valószínűségét.



3. ábra GritNet architektúra [32]

**Tanítás**

BLSTM előre és hátra irányuló LSTM rétegekkel, irányonként 128 cellát tartalmaz. A beágyazott réteg legjobb paraméterek számát az adathalmaz alapján keresték meg: egyik adatkészletnél 1024-től 3584-ig és 1024-től 5120-ig a másik készletnél. BLSTM kimenetén 10 és 20% közötti drop-out arányt alkalmaztak, hogy megakadályozzák a túlillesztést tanítás közben. [32]

**Előrejelzés pontossága**

A referencia modell (logisztikus regresszió) nyolc hét hallgatói adatot igényel, hogy elérje ugyanazt a teljesítményt, mint amit a GritNet csak három hét diákadattal képes elérni. A GritNet kulcsfontosságú előnye, hogy gyorsan adaptálható, de pontos előrejelzést add hallgatói eredmények becsléséhez. Gyorsítva a hallgatói visszacsatolási ciklust. [32]

## Gyakorlati feladatok sikerességének előrejelzése

Az oktatási rendszerekben a hallgatóknak szükségük van gyakorlati feladatokra, hogy a tudásukat leteszteljék. A hallgató önfejlesztésére proaktív szolgáltatásokat (pl. gyakorló feladat ajánlás) kell kínálni. Ennek érdekében meg kell jósolni, hogy a tanuló képes-e válaszolni a kérdésre.

**Problémák**

Sajnos számos technikai és tartományi kihívás áll fenn.

* Először is, a gyakorlatok sokféle kifejezése létezik, amely egységes módszert igényel, hogy megértsük a jellemzőiket.
* Másodszor, a hallgatók jövőbeni teljesítménye mélyen támaszkodik a történelmi gyakorlataikra. A diákok számára összpontosított információk nyomon követése nagyon nehéz.
* Végül a diákok teljesítmény előrejelzés feladata a „cold-start” problémájától szenved. Új hallgatóknál nem tudunk támaszkodni történelmi adatokra. Korlátozott mennyiségű információ rossz előrejelzés eredményekhez vezethet.

**EERRN keretrendszer**

Kihívások megoldására Anhui egyetem tanulói előálltak egy új javaslattal: Exercise-Enhanced Recurrent Neural Network (ERRN).

Az EERNN egy általános keret, ahol különböző stratégiák alapján megjósolhatjuk a hallgatók teljesítményét. Mind az EERNNM, mind az EERNNA modell ugyanazt a folyamatot alkalmazzák a hallgatói rekordok modellezésére, mégis különböző előrejelzési stratégiákat követnek.

Először egy BLSTM automatikusan jellemzi az egyes gyakorlati feladatokat. A megtanult gyakorlati specifikációkat szakértő nélkül kinyerhetők. Továbbá egy új LSTM követi nyomon a hallgatók állapotát. Az állapot a szekvenciális gyakorlati folyamat és a gyakorlat reprezentációjából áll össze. [33]

Az előrejelzés elkészítéséhez két keretrendszert vizsgáltak meg. Az első felépítés Markov attribútummal (EERNNM) a második a Figyelem mechanizmussal (EERNNA).

1. EERNNM: A hallgatók jövőbeli teljesítmények előrejelzése csak az aktuális állapotától függ. Tehát nincs szükség múltbeli állapotra (memoryless property) [34]
2. EERNNA: Fókuszál a hallgatók állapotaira, követi a múltbeli hasonló gyakorlatokat. [35]



4. ábra EERNN modellek [33]

**Modell tanulás**

Az oktatási pszichológia területén a kognitív diagnózis olyan technika, amellyel megjósolhatjuk a hallgatói teljesítményt azáltal, hogy felfedezzük a hallgatói állapotokat gyakorlási nyilvántartásból.

Miután megszereztük a gyakorlatok reprezentációját a szövegéből, következik a diák reprezentációja. Cél a tanuló gyakorlási folyamatának modellezése, amelyeket hallgatói állapotának neveznek. Az EERNN két alapfeltevésre támaszkodik:

* A hallgatói állapotokat mind a gyakorlatok, mind a kapott pontszámok befolyásolják.
* A hallgatók tanulás és felejtés a hosszú távú szekvenciális gyakorlási folyamat során történik.

Miután modelleztük az egyes hallgatók gyakorlási folyamatát, következik a teljesítmény előrejelzés. A pszichológiai eredmények azt állítják, hogy a hallgatói gyakorlatok a tanuló állapotától és a gyakorló feladat jellegétől függenek.

**Célfüggvény**

A mindkét javasolt modellben frissítendő teljes paraméterek főleg három részből állnak. A gyakorlat reprezentációs paramétereiből, a diákok ábrázolásának paramétereiből és az előrejelzési kimenet paramétereiből.

Az EERNN célfüggvénye a tanuló gyakorlási folyamatának megfigyelt sorrendjének negatív logaritmus valószínűsége. Szükség van az előre jelezett pontszámra () egy adott gyakorlati feladaton (). Továbbá az aktuális pontszámra (). Így a teljes vesztesége egy adott hallgató a következő [33]:



**Kísérleti eredmények** [33]

Az eredmények azt mutatják, hogy az EERNN keretrendszer jól kihasználja a gyakorlati rekordokat és a gyakorlati szövegeket.

Másodszor, a figyelem mechanizmussal (EERNNA) rendelkező modellek felülmúlják a Markov tulajdonságúakat (EERNNM), ami azt bizonyítja, hogy az előrejelzéshez hasonló gyakorlatok alapján hatékonyan fókuszált hallgatói beágyazásokat érdemes követni.

Az eredmények jelzik, hogy az EERNN keretrendszer jól tudja kezelni a hidegindítási problémát, amikor meg kell jósolja a hallgatók teljesítményét új gyakorlatokon.

## Regresszió neurális hálózattal

Regressziós probléma esetén a cél egy folyamatos érték, például ár vagy valószínűség kimenetének megjóslása. Ezt szembe állíthatjuk egy osztályozási problémával, ahol a cél egy osztály kiválasztása az osztályok listájából.

Egy jupyter notebook-on a megtisztított diák adatkészletet (student\_mat\_processed01.csv) használva egy modellt építek, ami diákok harmadik periodús (G3) pontját jelzi előre. A probléma megoldáshoz a tf.keras API-t használom.



Majd fel kell osztani az adatkészletet tanítókészletre és tesztkészletre. A tesztkészletet modelljeink végső kiértékelésére tudjuk használni. Ha megnézzük az attribútum statisztikát látható, hogy nagyon különböző tartományba esnek. Mint például:

|  |  |  |
| --- | --- | --- |
|  | **mean** | **std** |
| **health** | 3.550633 | 1.373433 |
| **absences** | 5.655063 | 8.045387 |
| **G1** | 10.768987 | 3.277696 |

**Normalizáció**

A különböző statisztikák arra adnak következtetést, hogy szükség van normalizációs lépésekre. Jó gyakorlat a különböző skálákat és tartományokat használó jellemzők normalizálása. Ennek egyik oka az, hogy a jellemzőket megszorozzuk a modell súlyával. Tehát a kimenetek és a gradiensek skáláját befolyásolja a bemenetek skálája. Bár egy modell a jellemzők normalizálása nélkül is működhet, viszont a normalizálás sokkal stabilabbá teszi az tanítást.

Az ***preprocessing.Normalization*** réteg tiszta és egyszerű módja annak, hogy az előfeldolgozást beépítsük a modellbe.

1. Az első lépés a réteg létrehozása
2. Ezután adaptálni az adatokhoz



**Egyváltozós lineáris regresszió**

A mély neurális hálózat modell felépítése előtt vizsgáljuk meg az adatkészletet lineáris regresszióval. Kezdésként egyváltozós lineáris regresszióval, jósoljuk meg a 3. periódus pontszámát a tanulási idő alapján. A modell készítés a Keras-szal tipikusan az architektúra definiálásával kezdődik. Esetemben a ***keras.Sequential*** modellt használom, ami lépések sorozatát reprezentálja:

1. Normalizálni a bemenetet a tanulási időt (studytime).
2. Lineáris transzformáció alkalmazása (), hogy egy kimenetet generáljunk a ***layers.Dense*** használatával.

A bemenetek számát beállíthatja vagy az *input\_shape* argumentummal, vagy automatikusan, amikor a modellt először futtatják.



A modell felépítése után a tanítás konfigurációt kell beállítani a Model.compile() metódussal. A complie() legfontosabb argumentumai a veszteség és az optimalizáló, mivel ezek határozzák meg, hogy mi lesz optimalizálva (átlagos abszolút hiba) és hogyan (az Adam optimalizáló). A modell konfigurálása után a Model.fit() használatával tudjuk végrehajtani a tanítást.



Vizualizáljuk a modell tanítását az előzmények szerint. Látható, hogy 0-10 közötti iterációban tanult a legtöbbet a modell. A cél az, hogy minél kisebb legyen a validációs hiba. Mindenesetre figyelnünk kell a tanítási és a validációs hiba között különbségre is.

* Ha a tanítási hiba sokkal kisebb, mint a validációs hiba, akkor beszélhetünk túlillesztésről. A tanító készletre jó eredményt add a modell, viszont nem látott értékeken gyengén teljesít.
* Fordítva, ha a validációs hiba sokkal kisebb, mint a tanítása, akkor alul illesztéssel nézzünk szembe. Jobban teljesít a validációs adaton, mint a tanító adatkészleten.



**Többváltozós lineáris regresszió**

Szinte azonos beállítással előrejelzéseket lehet tenni több bemenet alapján. Ez a modell továbbra is ugyanazt csinálja (), kivéve, hogy az egy mátrix és a egy vektor.

Az összes bemenet felhasználásával alacsonyabb tanítási és validációs hibát eredményez, mint az egyváltozós modell:



Általában kompromisszumot kell kötni a gyorsaság és a pontosság tekintetében. Míg az egyváltozós regresszió nagyságrenddel gyorsabb, addig a többváltozós pontosabb előrejelzést eredményez.

**Mély neurális hálózat**

Ez a szakasz egy és több bemenetű mély neurális hálózat modelleket valósít meg. A kód alapvetően megegyezik, azzal a különbséggel, hogy a modell kibővül néhány "rejtett" nemlineáris réteggel. A "rejtett" név itt csak azt jelenti, hogy nem kapcsolódik közvetlenül a bemenetekhez vagy kimenetekhez.

Ezek a modellek a következő rétegeket tartalmaznak:

* A normalizáló réteg.
* Két rejtett, nemlineáris, ***Dense*** réteg a ***relu*** (Rectified Linear Unit) aktivációs függvény használatával.
* Lineáris kimeneti réteg.

Mindkettő ugyanazt a képzési eljárást fogja használni, így a fordítási metódust az alábbi build\_and\_compile\_model függvény tartalmazza.



A mély neurális hálózat (DNN) modell tanítása egy bemenet (studytime) szerint.



Ha az előrejelzéseket a tanulási idő függvényében ábrázoljuk, akkor láthatjuk, hogy ez a modell hogyan használja ki a rejtett rétegek által biztosított nemlinearitást.



A neurális hálózat rájön, hogy a tanulási idő növelésével jobb eredményeket tudunk elérni. Viszont a függvény láthatóan túl meredek. A valóságban van egy fordulópont (tipping point), amitől kezdve nem fogunk exponenciális mértékű eredményt elérni. Jobban illeszkedni egy logaritmus függvény rá. Mindenesetre egy remek összefüggést nyert ki a hálózat a 3. periódus pontszám és a tanulási idő között.

**Full Model**

Megismételve ez előbbi folyamatot az összes bemenet felhasználásával, ez kissé javítja a validációs adatkészleten történő teljesítményét. Kissé több konfigurálható paraméterrel rendelkezik a neurális hálózat, mint egy egyszerű lineáris regressziós modell. Az alábbi táblázatban egy 8-4-1 neurálishálózat architektúra paraméter számosságát szemlélteti:

|  |  |  |
| --- | --- | --- |
| Réteg típus (Type) | Kimeneti alak (Shape) | Paraméter szám |
| normalization | (None, 45) | 91 |
| Dense | (None, 8) | 368 |
| Dense | (None, 4) | 36 |
| Dense | (None, 1) | 5 |
| Összes paraméter: 500 | | |
| Tanítható paraméter: 409 | | |
| Nem tanítható paraméter: 91 | | |

Könnyedén ki lehet számolni a rétegekhez tartozó paraméter számot. Két dolgot kell összeadni: a súlyokat (weights) és az előítéleteket (bias). Mivel egy teljesen összekötött hálóról van szó (fully connected), így az előző réteg neuron számát össze kell szorozni a meglévővel. Majd hozzáadjuk az aktuális réteg előítéleteit. Az utolsó réteg esetében: (4 súly \* 1 neuron) + 1 bias = 5.

Az összes bemenet esetén látható egy kis túlillesztés. Továbbá a tanítás hosszabb ideai tart, mint a lineáris regresszió esetén. Közel háromszar annyi idő szükséges ugyanakkor tanítása hiba elérése a DNN megközelítéssel.



**Eredmények összevetése**

Következő táblázatban kiemeltem néhány modell teljesítményét. Különböző hyper paraméterekkel való kísérletezéseket a bal oszlopba említem meg. A cellákban a átlagos abszolút hiba található.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DNN hyper paraméterek | egyváltozós regresszió | többváltozós regresszió | egyváltozós DNN | többváltozós DNN |
| 64-32-1, relu | 3.549848 | 3.436738 | 3.457496 | 3.585140 |
| 64-32-16-8-4-1, relu | 3.549172 | 3.474072 | 3.471775 | 3.790980 |
| 32-16-1, linear | 3.525714 | 3.481254 | 3.537117 | 3.390226 |
| 32-16-8-4-2-1, linear | 3.515388 | 3.426862 | 3.505179 | 3.437827 |
| 8-4-1, linear | 3.537366 | 3.388921 | 3.544936 | 3.368469 |
| 8-4-1, relu | 3.535824 | 3.547977 | 3.550696 | 3.389491 |

Mind a négy megközelítés kis eltéréssel hasonló hiba értéket produkált. Tehát a probléma kör és az adatok megengedik számunkra, hogy egy sokkal egyszerűbb megoldást használjunk pl. egyváltozós lineáris regressziót.

Továbbá a mély neurális hálózat esetében észrevehető, hogy a tanítás lassabb. Mivel sokkal több paraméterekkel rendelkezik, ezért több időbe telik azokat finomra hangolni. A kompromisszum lehet tenni és használhatunk egy egyszerűbb hálózatot. Az eredmények azt mutatják, hogy nem csak a tanítási időt csökkentjük le az egyszerűbb hálózattal, hanem növelni tudjuk a teljesítményt is.

Több esetben is látható, hogy az egyszerűbb neurális hálózat jobb eredményt ér el, mint a komplexebb párja. Az aktivációs függvény kicserélése kisebb teljesítmény növekedést eredményezett. A ReLu aktivációs függvény kicserélése lineárisra csökkentette az átlagos hibát körülbelül 5%-al.

Összegezve azt tapasztaltam, hogy az egyszerű problémához elegendő egyszerűbb modellhez. Adatbányászati szempontból, ha rájövünk, hogy a tanulási idő a legmérvadóbb a diák pontszámok előrejelzésében, akkor bátran használhatunk egyszerű lineáris regressziót. Az kevesebb paraméterrel rendelkező neurális hálózat gyorsabban tanul, ami nagy előny az erőforrás-felhasználás tekintetében.

Fontolóra kell venni milyen megoldást használunk. A neurális hálózatok nagyon elterjedtek és számos probléma körre megoldást nyújtanak. Ezzel szemben fekete dobozok az ember számára, mivel nehéz megérteni a belső világának működését (SGD, dropout, regulazitation, activation). A megmagyarázhatóságot előtérbe kell helyezni (explainable AI). Könnyedén problémákba ütközhetünk, ha egy olyan modellt építünk, amit nem értünk teljességébe.

***„We need the ability to not only have high-performance models, but also to understands when we cannot trust those models.”*** – Alexander Amini

## Hogyan gyorsítsuk fel a neurális hálózatot?

Az egyik legfigyelemreméltóbb megfigyelés az aggyal kapcsolatban az, hogy bárhová nézünk is, a neuronok aktivitása ritka. Tehát csak kis százalékuk tűzel (firing) az idő bármely pontján. A ritkaság különbözhet egy százalék és több százalék között, de mindig ritka.

Az emberi agynak több mint 86 milliárd neuronja van. A neuron közötti szinapszisok száma pedig több mint 100 billió. Első hallásra ez óriási számnak tűnik, de ha közelebbről megvizsgáljuk, akkor látható, hogy neuronok közötti kapcsolatok száma ritka. Továbbá egy adott időben az aktív neuronok száma közel 10%. Összesen a bevitt kalória 20% használja az agyunk, ami egyenlő közel 20 watt fogyasztással. Lenyűgöző teljesítményre képes az emberi agy összehasonlítva egy szuperszámítógéphez, ami 4 040 000 watt-ot fogyaszt. [36]

**Teljesen összekapcsolt réteg problémája**

Meg kell próbálni az emberi agy felépítésének példáját venni, hogy sokkal kevesebb számítási erőforrást igényeljen a neurális hálózat. Ez lehetővé teszi a jövőbeli projektek számára, hogy soha nem látott mértékig méretezzék. Mi a baj az eddigi hálózat struktúrával?

A mély neurális hálózatok teljesen összekapcsolt rétegeket (FCL) használnak. Ezek sűrű mátrixok segítségével alakítják át a bemeneti vektorokat egy kimeneti vektor térré. Az FCL-t általában egy ilyen diagram ábrázolja: [37]



A probléma az, hogy az egyes FCL-ekben a kapcsolatok száma kvadratikusan növekszik a bemenetek és a kimenetek csomópontjainak számával együtt. A 16 bemenettel és 32 kimenettel rendelkező FCL 16 x 32 = 512 csatlakozással rendelkezik. Egy 32 bemenettel és 64 kimenettel rendelkező FCL 32 x 64 = 2048 csatlakozással rendelkezik. Ez gyakorlati felső határt szab az FCL-ben lévő bemenetek és kimenetek számának.

A konvolúciós rétegek megkerülik a kvadratikus FCL problémát és áttörésekhez vezetnek a képfeldolgozási modellekben. Konvolúciós hálózat esetén kihasználjuk azt a tényt, hogy a szomszédos pixelek fontos információkat osztanak meg.

De mi van, ha adataink nem ilyen módon vannak strukturálva? Például mi van akkor, ha a genetikai adatokat több tíz vagy százezer DNS markerrel modellezzük bemeneti jellemzőket? Ebben az esetben nincs struktúra, amit kihasználhatnánk.

**Ritkán összekapcsolt rétegek**

Feltétlenül szükséges, hogy egy réteg minden csomópontja a következő réteg minden csomópontjához kapcsolódjon? A válasz „nem”.

Könnyen el tudunk képzelni egy réteget, amely nem köt össze minden csomópontot kimerítően. A kutatás során ezt a fajta réteget gyakran ritkán kapcsolódó rétegnek (SCL) nevezik. [37]



5. ábra

Ezt könnyű elképzelni. Valójában a természetben található számos hálózat sokkal jobban viselkedik így. Például visszatérve az emberi ideghálózatok eredeti inspirációjához: az idegsejtek (a csomópontokhoz hasonlóan) csak néhány más idegsejthez kapcsolódnak.

**Neurális hálózat metszés kezdete**

A neurális hálózatok ritkaságának története a metszéssel kezdődik, amely egyfajta tömörítés révén csökkentheti a hálózat méretét. A tömörített modellek az IoT (Internet of Things) térnyerésével még fontosabbá váltak.

2015-ben publikált cikk egy háromlépcsős módszert javasol a neurális hálózatok metszésére: [38]

1. Teljesen összekapcsolt hálózat tanítása
2. A kevésbé fontos kapcsolatokat elmetszése. Minden kapcsolat, ami egy adott küszöb érték alá esik azt eltávolítjuk.
3. Újra tanítjuk a hálózatot a fennmaradó kapcsolatok súlyának finomhangolásához.

Meglepő módon azt mutatták, hogy ez a módszer nemcsak összehasonlítható a rendszeres mélytanulási modellekkel, de még nagyobb pontosságot is elérhet.



6. ábra: Neuronok szinapszis metszés előtt és után

7. ábra: 3 lépéses tanítási folyamat

A metszési technika azt mutatja, hogy a ritka modellek felülmúlják a sűrű modelleket, de az optimális neuron kapcsolatok megtalálásához azt javasolják, hogy először egy sűrű modellt kell tanítani. Következtetésképpen ez hatékonyság növekedéshez vezet, viszont nem a tanítás fázisba, mivel duplán tanítjuk a hálózatot.

**Metszés Keras-ban**

A nagyságon alapuló súlymetszés technikát fogom alkalmazni a neurális hálózat tanítása során. Ebben az esetben fokozatosan nullázza a modell súlyait a ritkaság elérése érdekében. A ritka modelleket könnyebb tömöríteni és a gyorsabb, mivel átugorhatjuk a nulla értékeket. Akár hatszoros javulást láthattunk a modell tömörítésben, minimális teljesítmény vesztéssel. [39]

Az összehasonlításhoz először szükség lesz egy alap modellre (baseline model), amit később összevetünk a metszett modellekkel (pruned model). Az eddigiekhez hasonlóan egy ***keras.Sequential*** modellt definiálok, ami a következő rétegekből állnak:

* Bemeneti réteg 45 jellemzővel.
* Három darab teljesen összekötött (FC) rejtett réteg. Sorrendbe 32, 16, 8 neuronszámmal rendelkeznek. Az aktivációs függvény ReLu.
* Kimeneti réteg az előző réteggel teljesen összekötött réteg, ami egy neuronnal rendelkezik.

Tanítás során az átlagos abszolút hibát optimalizálom az Adam optimalizáló segítségével 0.001 tanítási ráta használatával.

Az összes paraméterek száma a teljesen összekötött hálózatnál 2145, amiből az összes tanítható. A modell lementésre került tanítás követően. Átlagos abszolút hiba: **3.384** gzippelt mérete: 9866 bytes.

**Modell finomhangolása metszéssel**

A metszést az egész modellre fogom alkalmazni, ami gyakran teljesítmény csökkenéssel jár. Lehetőség van szelektív metszésre, azaz csak adott rétegeket ritkítani. Továbbá létre lehet hozni egyedi réteget és azon belül kiválasztható, hogy milyen paramétereket (prunable) ritkítsunk.

Jobb pontosság érdekében a következőket vehetjük figyelembe: [39]

* Általában jobb metszéssel finom hangolni, mint nulláról tanulni.
* Érdemes metszeni a későbbi rétegeket az első rétegek helyett.
* Kerüljük a kritikus rétegek metszését (pl. az LSTM-nél az Attention réteget).
* Az előítélet (bias) metszése általában károsítja a modell pontosságát.

Esetemben a modellt 50% ritkasággal (50% nulla súllyal) kezdem, és 80% ritkasággal fogom befejezni. Tensorflow kínál egy csomagot a modell optimalizáláshoz. Egy függvény segítségével, pedig konfigurálható a metszési stratégia.



A függvény egy tf.keras modellt vagy egy egyedi modellt metszési funkcióval körülöleli (wrapper), amely megrövidíti a rétegek súlyait az tanítás során. Az első bemeneti argumentum egy keras réteg, keras rétegek vagy keras modell lehet. A teljesen összekötött modell mellé meg kell adni a metszési paramétereket.

Metszési stratégiaként a keras.PolynomialDecay választottam. Ekkor a metszés mértéke kezdetben gyorsan növekszik a kezdetben, de a végső ritkaság felé lassul. A kezdeti ritkaság 50% a végső pedig 80%. Metszési stratégia használatával energia hatékonyság érhető el, ezzel támogatva az erőforrás-korlátozott környezeteket. [40]



Ezt követően újra le kell fordítani (compile) a modellt, majd következik a megszokott modell tanítás (fit). Tanítás közben szükség van függvény hívásokra, hogy a súlyokat módosítsuk. A finomhangoláshoz a következő hívásra van szükség: ***tfmot.sparsity.keras.UpdatePruningStep()***

Első metszés igazán jól sikeredet. Két epoch után az átlagos abszolút hiba: **3.277**. Tehát jobban teljesít a ritka neurális hálózat, mint a teljesen összekötött párja. Mekkora méret tömörítést sikerült elérni?

A rikta modellek lementéséhez szükség van egy lépésre, ami előkészíti a modell lementését (***tfmot.sparsity.keras.strip\_pruning***).



A gzippelt méret: 6795 bytes. A méret tovább csökkenthető Tensorflow Lite csomagban lévő konverterek használatával (***tf.lite.TFLiteConverter***). A gzippelt TFLite modell mérete: 6005 bytes. Tehát nem csak teljesítmény növekedést tudunk elérni a modell metszéssel, hanem közel **40%**-os (100%-60,86%) ***tömörítés***t is. Figyelembe kell venni, hogy a kisebb modelleknél kevésbé észrevehető tömörítést érhető el. Ellenbe nagyobb modelleknél, mint például a InceptionV3 képfelismerőnél (paraméterek: 23,851,784 méret: 92MB) sokkal számottevőbb tömörítés érhető el.

**Futtatási eredmények**

Első futtatás túlságosan bíztató volt, ezért megismételtem párszor a kísérletet. Háromféle esetben vizsgáltam meg a metszés hatékonyságát. Először 2, 4 és 8 darab tanítási iterációban (epoch) használtam hálózat metszést.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Epoch szám | Baseline modell | Pruned modell |
| 1. | 2 | 3.3839049339294434 | 3.277818441390991 |
| 2. | 2 | 3.503624200820923 | 5.358527660369873 |
| 3. | 2 | 3.503624200820923 | 3.6340973377227783 |
| 4. | 2 | 3.384648084640503 | 4.181692600250244 |
| 5. | 4 | 3.4953389167785645 | 3.643183946609497 |
| 6. | 4 | 3.695019245147705 | 3.6739559173583984 |
| 7. | 4 | 3.66666841506958 | 3.665585994720459 |
| 8. | 8 | 3.6481285095214844 | 3.6767160892486572 |
| 9. | 8 | 3.70527982711792 | 3.7090964317321777 |
| 10. | 8 | 3.8007121086120605 | 3.7642366886138916 |

Látható, hogy minél több tanítási iterációt veszünk igénybe, annál közelebbiek az értékek. Két iteráció esetén egy szembe tűnő teljesítmény romlás történt. Újra tanítást követően nagy mértékben javult a metszett hálózat pontossága. Összeségében nagyon ígéretes a metszett neurális hálózatok működése. Tíz futási eredményből négy esetben a metszett hálózat jobban teljesít, mint a teljesen összekötött párja. Az eredmények alapján kijelenthetjük, hogy a kevesebb néha több.

**Ritka evolúciós tanítás (SET)**

Hogyan tudjuk megmondani, mely kapcsolatokat használjuk a neuronok között? A neuronok összekapcsolásának megtanulásának módja kiemelkedő probléma. A neuronok összekapcsolásának intelligens módjának megtalálása az Adaptive Sparse Connectivity publikáció témája. A Sparse Evolutionary Training (SET) eljárásnak nevezett algoritmus a következő. [41]



Alapvetően véletlenszerűen inicializáljuk az SCL-eket a hálózatunkban. A véletlenszerű inicializáláshoz Erdős-Rényi gráf topológia van használva. A véletlenszerűséget egy valószínűségi változóval () lehet befolyásolni.

Majd a tanítás hátra terjesztéssel (backpropagation) és más szabványos mély tanulási optimalizálási technikák segítségével történik. Minden korszak (epoch) végén a legkisebb súlyú kapcsolatokat („leggyengébb” kapcsolatok) eltávolításával és véletlenszerűen inicializált új kapcsolatok helyettesítésével végződik.

A SET meglepően robusztus és stabil. Bátorító módon a szerzők az FCL modellekhez nagyon hasonló eredményeket tudnak felmutatni (néha meghaladják a teljesítményüket) a sokkal kevesebb paramétert tartalmazó SET modellekkel.

A SET hálózatok nemcsak a célt tanulják meg, hanem kódolják a bemeneti jellemző fontosságát az egyes rétegekben kialakult kapcsolatokban is. Ha megnézzük az MNIST probléma bemeneti kapcsolatai eloszlásának alakulását, láthatjuk, hogy a modell implicit módon megismeri a prediktív információk eloszlását a bemeneti adatokban az általa megtartott kapcsolatokon keresztül. Egy olyan tartományban, ahol még nem ismerjük ezt a disztribúciót, ez az információ segíthet abban, hogy érdekes struktúrákat és összefüggéseket „fedezzünk fel” a strukturálatlan bemeneti adatokban. [41]



**GPU támogatás**

Tehát, ha ritka hálózatokkal minden rendben van, akkor miért nem azokkal próbálkozunk? Van egy hátrány, amit figyelembe kell venni. A metszéssel létrehozott ritka hálózatoknak nincs nyilvánvaló hatékonyságnövelésük, ha GPU-on tanítjuk őket. Sajnos a könyvtárakban a támogatás meglehetősen „ritka”. [42]

Jó hír, hogy a közelmúltbeli bejelentés szerint, az OpenAI a block-sparse eszközkészletet fogják megvalósítani PyTorch-ban. Magasan optimalizált GPU-kerneleket szeretnének kiadni a ritka neurális hálózati architektúrákhoz. Tehát az OpenAI bejelentése a block-sparse eszközéről nagyon jó hír, mindazoknak, akik ritka GPU operációkat akarnak használni anélkül, hogy fel kellene áldozniuk az tanítási sebességet. [43]



**Ritkaság előnyei**

A gépi tanulás ritka hálózatainak inspirációja az agyból származik, és számos más tulajdonság, amely az emberek tanulásához kapcsolódik. Az egyik az, hogy robusztus. A ritka reprezentációk természetesen robusztusabbak a zajra, amit a publikáció is bemutattak. A ritka kapcsolati halmaz és ritka aktiválási funkció, a ritka ideghálózatoknál megelőzik a sűrű hálózatok teljesítményét zaj hozzáadásakor. [44]

A másik nagy előnye, hogy sokkal kisebb a modell mérete. Ezáltal az erőforrás igénye is kisebb a ritka modellek esetén. Továbbá a ritka modellek jobban illeszkedhetnek a **folyamatos** (egész életen át tartó) **tanulás**hoz, azaz ugyanazon hálózat használata, még akkor is, ha a feladatok és az alapul szolgáló statisztikák változnak.

# Oktatást segítő eszközök használata

## Tananyag összegzés

Amikor tanulunk egy adott tárgyra sokszor túl sok információ áll rendelkezésünkre. Az órai jegyzetek, a kivetített dia szöveg, az ajánlott irodalmak. Rengeteg forrás van és mind értékes információkkal rendelkezik.

Sajnos a hallgatónak nem mindig jut ideje elolvasni ezeket a forrásokat. Milyen jó lenne, ha egy kattintással össze tudnánk foglalni egy hosszú szöveget. Az összefoglalóknak az eredeti hatalmas szövegből történő kivonásának módszerét nevezzük szövegösszefoglalásnak. Fontos, hogy az összefoglalás során ne veszítsünk el létfontosságú információt.

A további bekezdésekben különféle módszereket fogok tárgyalni az automatikus szövegösszefoglalásra.

**Szövegösszegző típusok**

A szövegösszefoglaló módszerek két fő kategóriába sorolhatók: Extraktív és Absztraktív

* ***Extraktív szöveg összefoglalás***: Ez a hagyományos módszer, amelyet először fejlesztettek ki. A fő cél a szöveg jelentős mondatainak azonosítása és hozzáadása az összefoglalóhoz.
* ***Absztraktív szöveg összefoglalás***: Ez egy fejlettebb módszer. A megközelítés a fontos szakaszok azonosítása, a kontextus értelmezése és újszerű reprodukciója. Ez biztosítja, hogy az alapvető információkat a lehető legrövidebb időn belül továbbítsák.



8. ábra [45]

**Extraktív szövegösszefoglaló**

Számos algoritmus létezik a szöveg összefoglalására. Mennyivel könnyebb lenne, ha lenne egy könyvtár, amely lehetővé teszi az összefoglalást több algoritmuson keresztül. Szerencsére már van ilyen összegző könyvtár. [46]

A *sumy* könyvtár számos algoritmust kínál a szövegösszegzés megvalósításához.

A következő bekezdésben az alábbi algoritmusok fogom bemutatni:

* LexRank
* Luhn
* KL-Sum

**LexRank**

Hogyan is működik valójában a LexRank? Az a mondat, amely hasonló a többi mondatok szövegéhez, az nagy valószínűséggel fontos. A LexRank megközelítése az, hogy egy adott mondatot más hasonló mondatok javasolnak, és ennélfogva magasabb besorolású. A magasabb besorolású mondatok fognak bekerülni az összesített szövegbe. [46]

Először is szükség van egy parser-re. Mivel egy szöveg fájlba tárolom a tartalmat, így a *PlaintextParser* osztályt használom. A parser mellé importálnia kell egy Tokenizer-t, ami a nyers szöveget tokenekké alakítja. Majd a LexRank összegző paraméteren keresztül meg tudjuk adni, hogy hány darab mondatig összegezünk.



A nyers szöveg 14 mondatból és összesen 1504 karaktert tartalmaz. A végeredmény 2 mondat, ami 299 karakter hosszú. Tehát a 20%-ra sikerült lerövidíteni az eredeti szöveget. Sajnos magyar nyelvű szövegösszegzést nem tudtam elérni, mivel nem találtam hozzá megfelelő tokenizert.

**Luhn**

A Luhn-összefoglaló algoritmus megközelítése a TF-IDF-en (Term Frequency-Inverse Document Frequency) alapul. Ez akkor hasznos, ha a nagyon ritkán előforduló szavak, valamint a nagyon gyakran előforduló szavak (zárszavak) sem jelentõsek. [46]

Az implementáció teljesen hasonló, mint a LexRank esetében, annyi különbséggel, hogy más osztályt használunk. Az összegzett szöveg a következő:

*„Your computer's operating system (OS) manages all of the software and hardware on the computer.*

*Most of the time, there are several different computer programs running at the same time, and they all need to access your computer's central processing unit (CPU), memory, and storage.”*

Látható, hogy egy mondat megegyezik az előzővel. Azzal magyarázható, hogy a megegyező mondatba, sok zárszó (’the’, ’a’, ’and’) van. Emiatt az első algoritmus fontosnak tekintheti. A Luhn algoritmus kizárja ezeket a szavakat és ennek ellenére is látható az összegzőbe a mondat. Ez azt jelenti, hogy fontos a megegyező mondat, mivel két algoritmus is belevette az összegzésbe.

**KL-Sum**

A szavak eloszlásának hasonlósága alapján választja ki a mondatokat. Célja a KL-divergencia (Kullback–Leibler) kritériumok, másnéven a relatív entrópia csökkentése. Kapzsi optimalizálási megközelítést alkalmaz, és folyamatosan kiegészíti a mondatokat, amíg a KL-divergencia csökken. [47]



Az összes algoritmus összegzése közül a KL-Sum algoritmus adja a legkompaktabb összegzést. Az eredeti szövegből 149 karakter számú szöveget készített. Számomra a KL-Sum összegzés tetszik legjobban, mivel egyszerű és tömör megfogalmazást add.

**Absztraktív szövegösszefoglaló**

Az absztraktív összefoglalás az új korszerű módszer, amely új mondatokat generál, amelyek a legjobban reprezentálhatják az egész szöveget. Ez jobb, mint az extrakciós módszerek, amikor a mondatokat csak az eredeti szövegből választják ki az összefoglaláshoz.

Hogyan lehet egyszerűen megvalósítani az absztrakciós összesítést? Egyszerű és hatékony módja a Huggingface transzformátor-könyvtárának. [48]



A HuggingFace támogatja a legmodernebb modelleket olyan feladatok végrehajtásához, mint az összesítés, osztályozás, fordítás, szöveggenerálás stb. Néhány általános modell a GPT-2, GPT-3, BERT, OpenAI, GPT, T5.

**T5 kódoló**

A T5 egy kódoló-dekóder modell. Először importálnia kell a tokenizert és a hozzá tartozó modellt. Az előképzett „t5-base” modellt a *from\_pretrained* metódussal lehet példányosítani. A következő a legfontosabb lépés, amelyet nem szabad elfelejteni. A nyers szöveg elejéhez hozzá kell adnia az „summarize:” karakterláncot. A T5 transzformátorok különböző feladatokat hajt végre azáltal, hogy milyen előtagot adunk a bemeneti szöveghez.





Látható, hogy a modell egy tenzort adott vissza id-k sorozatával. Az extrakciós módszerekkel ellentétben az összegzés kimenet nem része az eredeti szövegnek.

*„a* ***time-shared*** *operating system allows users to share a computer* ***at a time****.”*

**GPT-2**

Az elmúlt években egyre nagyobb az érdeklődés a nyílt végű szöveg generálás iránt. Szerencsére a nagyméretű transzformátor alapú nyelvi modellek megjelentek, amelyeket több millió weboldal képzett ki, például az OpenAI híres GPT-2 modellje.

A modell **automatikus regresszív nyelvi generálás**t használ. Azon alapul, hogy a szavak sorrendjének valószínűségi eloszlásából készíthetünk feltételes következő lehetséges szó eloszlást. A legnagyobb valószínűségű szavak kerülnek az ajánlásba. [49]

W0 a kezdeti szövegkörnyezet. A szekvencia *T* hosszát általában menet közben határozzák meg, és megegyezik a t = T időponttal, az EOS (endofsequence) tokent generálják.

A jelenleg legelterjedtebb dekódolási módszerek:

* Kapzsi keresés
* Sugárnyaláb keresés
* Top-K mintavétel
* Top-p mintavétel

A szöveg összegzéshez felhasználóm a GPT-2 által kínált szöveggenerálást. Az esetemben két szöveget fogok használni: operációs rendszerrel és többfeladatos rendszerrel kapcsolatos szöveg.

Mindkét esetben ~1500 karakter hosszú szöveghez akarunk összegzést generálni. A GPT-2 modellek közül a gpt2-large betanított modell-t használom. A szavak ajánlásánál csak a 90%-nál nagyobb valószínűséggel előforduló szavakat vesszük figyelembe. Beállítható, hogy mennyire legyen determinisztikus a szavak kimenetele, ezt a Temperature paraméterrel lehet meghatározni. A Boltzmann eloszlást vezérli. Minél nagyobb a szám annál véletlenszerűbb a kimenet. Minél kisebb annál inkább determinisztikus. [50]



A megközelítéssel sikerült az eredeti szövegből 410 karakter hosszú összegzést létrehozni. A szöveg teljesen új fajta szavakat és terminológiákat használ.

*„In summary, operating systems are programs that run on your computer, and they manage the hardware that makes up your computer, like the CPU, memory, storage, and* ***network interfaces****. The OS handles most of the tasks in your computer, but it doesn't always take care of all the necessary functions,* ***like checking your email or playing music****.”*

Új szavakkal egészíti ki az előző szöveget. Továbbá példákat is használ, hogy megmagyarázzon szavakat. Lenyűgöző, amire képesek ezek a modellek egy újkorszak nyílt meg a nyelvi feldolgozás történelmében.

**Józan ész**

Figyelemre méltó fejlődések vannak az NLP feladatok terén, beleértve a kérdések megválaszolását és az olvasás megértését. Azonban viszonylag kevés munkát végeztek a kontextus ábrázolásnál található józan ész vizsgálatával.

Érdekes módon a modellek zavart mutatnak ezekben a tesztesetekben, ami arra utal, hogy a józan észt inkább a felszínen nem pedig mélyen tanulják meg.

Például a GPT2-base alul teljesít a GPT-hez képest a józan ész szempontjából. Az a tény, hogy a RoBERTa-bázis paramétereinek mérete megegyezik a GPT2-vel, ugyanakkor jobban teljesít a tesztesetekben, ami arra utal, hogy a kétirányú modellek nagyobb reprezentatív erővel bírnak a józan ész képességeiben. [51]

Lássunk egy élő példát, hogyan teljesít egy józan ész teszten a GPT-2. Első esetben megemlítem, hogy Sam szeretne zongorázni, de eltört a széke.

*„Sam did want to play on the piano. He pulled up a chair to the piano, but the chair was broken, so he had to* ***stand up to play it.****”*

Úgy tűnik teljesen jól értelmezi a szituációt. Viszont mi történik, ha kiveszük, hogy Sam szeretne zongorázni.

*„Sam pulled up a chair to the piano, but the chair was broken, so he had to* ***sit on the floor next to the piano.****”*

GPT2 szerint, csak le kell ülni a zongora mellé a földre. Az ember probléma nélkül rájön, hogy Sam azért hozott egy széket a zongora mellé, mert zongorázni szeretne.

Online GPT-2 transzformátor: <https://transformer.huggingface.co/doc/gpt2-large>

A józan ész az embernél megadatott. A világ „megértéséhez” szükséges történeteket, ötleteket és összetett feladatokat könnyedén értelmezzük minden erőfeszítés nélkül. Ennek ellenére a mesterséges intelligenciának óriási erőfeszítést kell tennie (több millió számítás másodpercenként), hogy megértse a szavakban és a mondatokban lévő mögöttes tartalmat.

*„Peter bought a new clean gown. Peter went to work, but his working clothes was dirty. He had no time to clean it. So he* ***decided to put on the new clean one****.”*

## Kérdezz felelek

A kérdések megválaszolása fontos NLP feladat és hosszú távú mérföldkő a mesterséges intelligencia rendszerek számára. A QA rendszerek lehetővé teszik a felhasználó számára, hogy természetes nyelven tegyen fel kérdést, és gyorsan és tömören megkapja a választ a kérdésére. Manapság a QA rendszereket a keresőmotorokban és a telefonos beszélgetési interfészekben használják, és nagyon jól tudnak válaszolni az egyszerű tényszerű kérdésekre.

Az a képesség, hogy elolvashatunk egy szöveget, majd megválaszolhatjuk a vele kapcsolatos kérdéseket, szövegértésnek nevezzük. A szövegértés kihívást jelent a gépek számára, mind a természetes nyelv megértését, mind a világ ismeretét megköveteli.

**Első lépés az olvasásmegértés kihívásában**

Hogyan léphetünk előre az olvasásmegértés kihívást jelentő feladatában? Először is nagy mennyiségű és jó minőségű adatokra van szükség. Szerencsére a Stanford Question Answering Dataset, más néven SQuAD. 100 000 kérdés-válasz párnál rendelkezik. Továbbá sokféle témakört fed le a különböző területeken, a zenei hírességektől az elvont fogalmakig. [52]



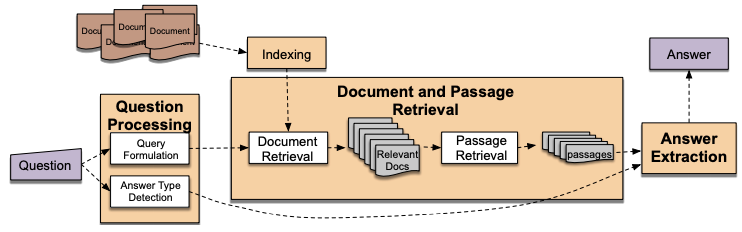
A szakasz (passage) egy cikk része, hossza változó lehet. Az SQuAD minden egyes részéhez kísérő szövegértési kérdések tartoznak. Ezek a kérdések a részlet tartalmán alapulnak, és az átolvasással megválaszolhatók. A jó szakaszok és kérdés bemenetek, minél változatosabb annál jobb:

* Szinonima szavak: a kérdéses szavak gyakran a szinonimái a szakaszba lévőknek (lexikális variáció).
* Külső tudás: ebben az esetben meg kell érteni a szavak mögötti értelmet. Az ilyen kérdésekre nehéz választ adni, mert túlmutatnak a szakaszon.
* Szintaktikus variáció: ekkor a kérdések és a szakasz mondatok struktúrája meglehetősen eltérnek.
* Több mondatos értelmezés: Az ilyen jellegű kérdésekhez a válaszban több mondatot kell használni a szakaszban. Az idő nagy részében ez magában foglalja a tárgyilagos felbontását annak az entitásnak az azonosítására, amelyre a névmás utal.

**Kérdés válaszoló rendszer felépítése**

A kérdések megválaszolásának számos paradigmája van, de két fő paradigmát alkalmaztak: tudásalapú és információ visszakeresésen alapuló kérdésválaszolást. Amikor az emberek válaszolnak egy kérdésre, először saját tudásukkal próbálnak válaszolni a kérdésre. Ha nem tudnak válaszolni a kérdésre, az interneten vagy könyvekben keresik a választ. A kérdéseket megválaszoló rendszerek hasonlóak az emberekhez. Az előbbi tudásalapú rendszernek, az utóbbi pedig információ-visszakeresésen alapuló rendszernek felel meg. [53]

Az információ-visszakeresésen alapuló rendszer négy részre osztható: kérdésfeldolgozás, dokumentum kinyerés, szakasz kinyerés és válasz kibontás. [54]

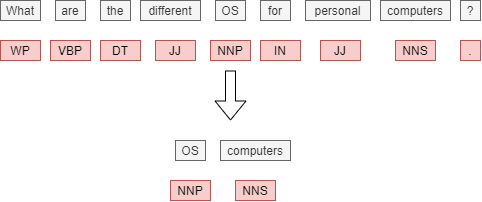


**Kérdésfeldolgozás**

A kérdésfeldolgozás átalakítja a kérdést keresési lekérdezéssé. Általában a végszavakat (stopwords) és meghatározott beszédrészeket tartalmazó szavakat eltávolításra kerülnek. Ezenkívül a mély tanulási technológia legújabb fejlődésével lehetővé vált egy olyan vektor létrehozása, amely jól kifejezheti a mondat jelentését, ezért alkalmazzák a kérdés vektorokká alakításának és lekérdezésként való felhasználásának módszerét is.

Ezúttal az **nltk** könyvtár használatával generálunk egy keresési lekérdezést. Először a kérdés elemzés történik, majd címkékkel látjuk (part-of-speech tag) el a kérdés szavait. Ahhoz, hogy hozzá tudjunk férni a címkékhez (POS-tag), le kell töltenünk az nltk-hoz tartozó adatokat: **nltk.download()**.

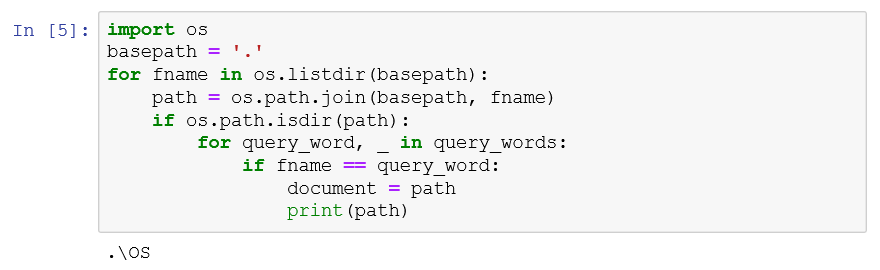
Ezután csak a számunkra fontos szavakat tartjuk meg a szavak címkéi alapján. Az egyszerűség kedvéért csak a főneveket („NN”) tartottam meg, de bármilyen más logikát lehet alkalmazni. Például: „Milyen különböző OS-k vannak a személyi számítógépekhez?” generálja az „OS számítógép” lekérdezést.



**Dokumentum kinyerés**

A dokumentum-visszakeresés során a releváns dokumentumokat a létrehozott lekérdezés segítségével fogjuk lekérni. A későbbi feldolgozás során, mivel a válasz kivonásra kerül ezekből a dokumentumokból, a lehető legnagyobb mértékben meg kell keresni azokat a dokumentumokat, amelyek tartalmazhatják a választ. A kérdésfeldolgozáshoz hasonlóan a mély tanulási technológia lehetővé teszi számunkra, hogy a dokumentumokat vektorokká konvertáljuk. [55]

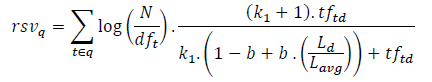
Áttekinthetőség érdekében a dokumentumokat a mappák fogják reprezentálni. A benne lévő szövegfájlok, pedig a szakaszokat (passage) fogják szimbolizálni. A kinyert lekérdező szavak által történik a dokumentum keresés. Feltárja az összes olyan mappát, amely tartalmazza a lekérdező szavakat.



**Szakasz kinyerés**

Az szakasz kinyerés során a dokumentumokat kisebb egységekre osztják, például mondatokra és bekezdésekre. Majd kiválasztják azokat a részleteket, amelyek valószínűleg tartalmazzák a választ. Ha a dokumentum rövid, akkor nem szükséges, de ha hosszú, akkor a szakaszválasztás hatékony, mert nem tudjuk, hogy a dokumentum mely részének felelt meg a lekérdezés. Általában a későbbi válasz kibontás folyamat hosszú ideig tart, így a szakaszok kiválasztása fontos lépés, hogy felgyorsítsuk a következő folyamatot.

Ezúttal a kérdéshez hasonló részeket választunk ki. A hasonlóság kiszámításához a ATIRE BM25 algoritmus fogom használni. Az ATIRE BM25 variáns fordított dokumentum gyakoriság (IDF – inverse document frequency) logritmus függvénybe ágyazza. A negatív IDF-pontszámok megakadályozása érdekében az algoritmus egy egész-részt is hozzá ad az IDF epsilon () értékhez. A kinyerési állapot értékének kiszámításához a következő egyenlet számólható ki: [56]



Egy adott lekérdezése: , a visszakeresési állapot értéke: , az egyes pontszámok összege: .

Mivel a BM25 nagy teljesítményű, gyakran használják alapként a szakaszok kinyeréséhez. A használata végetlenül egyszerű:

1. Megfelelő könyvtár importálása: from rank\_bm25 import BM25Okapi
2. Inicializálás a szakaszokkal: bm25 = BM25Okapi(tokenized\_corpus)
3. Érték lekérdezés: doc\_scores = bm25.get\_scores(query\_words)

**Válasz kibontás**

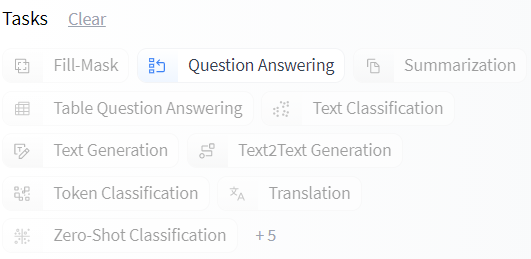
A válasz kinyeréshez a Google betanított modelljét fogjuk használni, a BERT-et. Előszőr is szükség van a már telepített ***transformation*** könyvtárra és a TensorFlow-ra vagy PyTorch-ra. A kérdés megválaszolásának folyamata három fő lépésből áll:

1. Modell és tokenizer inicializálása
2. Lekérdezés tokenizálása
3. Csővezeték és előrejelzés

Ezek a lényegek - a valóságban valószínűleg számos más lépés is megtörténik, például az adatok előfeldolgozása vagy a kontextus visszakeresése. De a legegyszerűbb formájában csak ezen három lépés miatt kell aggódnunk.

**Modell és tokenizer inicializálása**

Először keressünk egy modellt, amelyet használni szeretnénk. Ha átlépünk a huggingface.co/models oldalra, akkor ki tudjuk választani a Question-Answering filtert. Továbbá filterek a könyvtárak, adathalmazok, nyelvek és licenszek.



Majd inicializáljuk a modellünket és a tokenizer-t a: *distilbert-base-uncased-distilled-squad* használatával.

**Lekérdezés tokenizálása**

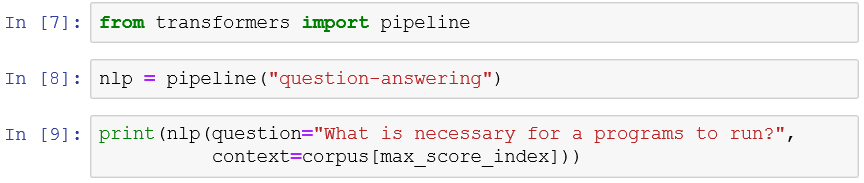
Következő lépés, hogy betöltsünk egy szöveget és token azonosítókká alakítsuk, hogy a Bert által is olvasható legyen. A Bert tokenizer felelős azért, hogy az ember által olvasható szöveget Bert-barát adatokká, token-azonosítókká alakítsa. Néhány ilyen token, például a [CLS] és a [PAD] - ezek speciális tokenek, amelyeket különböző dolgok jelölésére használnak Bert számára. Bert által használt speciális tokenek az alábbi táblázatba láthatók. [57]

|  |  |  |
| --- | --- | --- |
| Token | Jelentés | Token ID |
| **[PAD]** | Padding token lehetővé teszi számunkra az azonos hosszúságú szekvenciák (512 tokenek a Bert számára) fenntartását akkor is, ha különböző méretű mondatokat adunk be | 0 |
| **[UNK]** | Akkor használják, ha egy szó ismeretlen Bert számára | 100 |
| **[CLS]** | Minden sorozat elején megjelenik | 101 |
| **[SEP]** | Elválasztót jelöl - a kontextus-kérdés közötti pont jelölésére szolgál, és a szekvenciák végén jelenik meg | 102 |
| **[MASK]** | Tokenek maszkolásakor használatos, például maszkos nyelv modellezéssel (MLM) végzett képzésben | 103 |

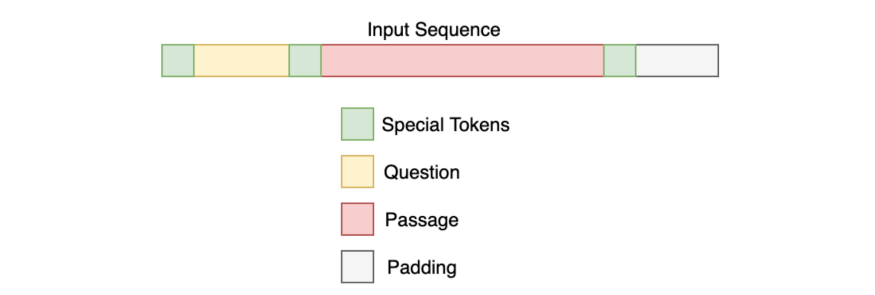
A kezdeti tokenizálás után ezt a karakterlánc-listát (tokenek) konvertáljuk egész számok listájává (token-azonosítók). Ez egy belső szótár használatával történik, amely minden Bert által megértett tokent tartalmaz.

**Csővezeték és előrejelzés**

Áttérhetünk a kérdések feltevésére!



Most ez megtévesztően egyszerű - de valóban elegendő a Q&A modell kódolására és a kérdések feltevésére. Azonban bontsuk le egy kicsit ezt a csővezeték funkciót, hogy megértsük, mi történik valójában. Először is átadjuk a kérdést és a kontextust. Amikor ezeket egy Q&A modellbe illesztjük, egy speciális formátum várható - amely a következőképpen néz ki:



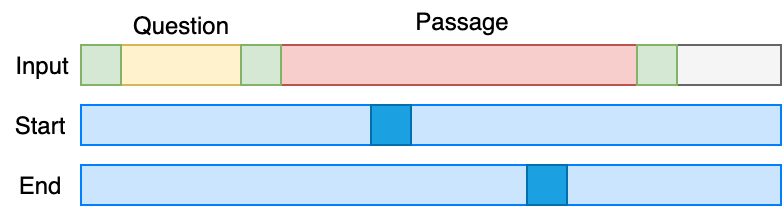
9. ábra A Bert bemenetének tartalma [58]

Amit aztán konvertálunk token ID formátumba:

[101, 2054, 2024, 1996, 2367, 9808, 2005, 3167, 7588, 1029, 102, 2019, 4082, 2291, 2003, 1996, 2087, 2590, 4007, …, …, …, 3001, 2024, 2881, 2000, 2022, 3733, 2000, 2224, 1010, 1998, 2087, 1997, 1996, 3937, 6481, 2024, 1996, 2168, 1012, 102]

Az ID-k tovább adódnak a Bert-nek. Ezen a ponton Bert nyelvi lépéseket végez, hogy azonosítsa a válasz helyét a megadott kontextusban. Miután Bert elhatározta, visszaadja nekünk a válasz tartományát - vagyis a kezdő token indexet és a vég token indexet. Ami esetünkben így fog kinézni:

{'score': 0.04071327671408653, 'start': 557, 'end': 607, 'answer': 'central processing unit (CPU), memory, and storage'}



10. ábra A válasz pozíció kiválasztása [58]

Ezután a csővezetékünk kihúzza a vonatkozó token azonosítókat Bert kezdő-végi token indexének előrejelzése szerint. Ezeket a token azonosítókat ezután visszavezetjük a tokenizer-be, hogy az ember által olvasható szöveggé dekódoljuk.

**Hogyan találja meg BERT a válaszokat?**

Először is nézzük, meg hogyan írják le a Bert NLP keretrendszert: „A BERT a **B**idirectional **E**ncoder **R**epresentations from **T**ransformers jelenti. Úgy tervezték, hogy előkészítse a mély kétirányú ábrázolásokat a felirat nélküli szövegből a bal és a jobb oldali kontextus együttes kondicionálásával. Ennek eredményeként az előre kiképzett BERT modell csak egy további kimeneti réteggel finomhangolható, hogy a legmodernebb modelleket hozza létre az NLP feladatok széles köréhez.” [59]

Ez túl komplexen hangzik kiindulópontként. Viszont remekül összefoglalja, hogy mit csinál a BERT. A legfontosabb következtetés, hogy a BERT a Transformer architektúrán alapul. Másodszor, a BERT előre tanított egy nagy, felirat nélküli szövegekre (Wikipédia: 2500 millió szó, Book Corpus: 800 millió szó).

Harmadszor, a BERT egy „mélyen kétirányú” modell. Kétirányú azt jelenti, hogy a BERT a képzési szakaszban egy token kontextusának bal és jobb oldaláról egyaránt megtanul információkat. A modell kétirányúsága fontos a nyelv jelentésének valódi megértéséhez. Lássunk egy példát ennek szemléltetésére.

„***Elmentünk a városi*** bankba”

„El kell mennem a bankba, ***hogy befektessek***”

Ha megpróbáljuk megjósolni a „bank” szó jellegét úgy, hogy csak a bal vagy a jobb kontextust vesszük figyelembe, akkor a megadott két példa legalább egyikében hibát fogunk elkövetni. Ennek egyik módja a bal és a jobb kontextus figyelembevétele, mielőtt előrejelzést adnánk. A BERT pontosan ezt teszi.

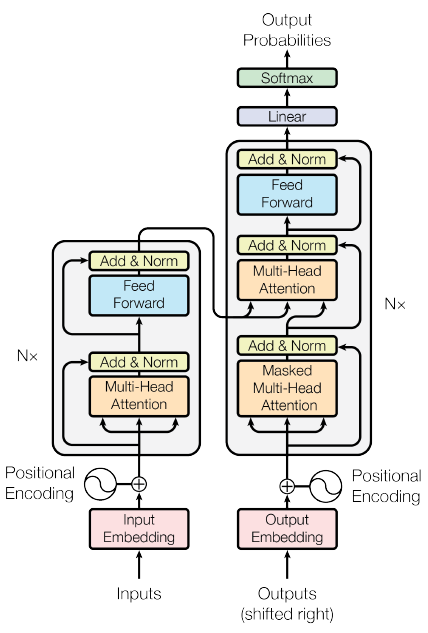
Végül a BERT lenyűgöző aspektusa a finomhangolás. Finomra hangolhatjuk úgy, hogy csak néhány további kimeneti réteget adunk hozzá, hogy a legmodernebb modelleket hozzuk létre a különféle NLP feladatokhoz.

**BERT architektúra**

A BERT architektúra a transzformátor tetejére épít. Jelenleg két változat áll rendelkezésre:

* BERT-base: 12 réteg (transzformátor blokkok), 12 attention head és 110 millió paraméter
* BERT-large: 24 réteg (transzformátor blokkok), 16 attention head és 340 millió paraméter

Hogyan épül fel egy transzformátor modell architektúra?



Dekódoló

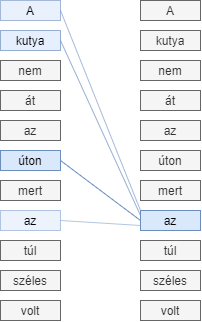
Kódoló

11. ábra A Transformer - modell architektúra [60]

A Kódoló blokknak van egy rétege a Multi-Head attention, amelyet egy másik réteg követ a Feed Forward neurális hálózat. A dekódernek viszont van egy extra Masked Multi-Head Attention rétegje. A kódoló és a dekóder verem müködése a következő lépésekből áll:

* A bemeneti szekvencia szóbeágyazásait továbbítjuk az első kódolónak.
* Ezeket aztán átalakítják és továbbterjesztik a következő kódolóba.
* A kódoló-verem utolsó kódolójának kimenetét a dekóder-verem összes dekóderéhez továbbítjuk.

Fontos megjegyezni itt - az önfigyelő (self-attention) és a feed-forward rétegek mellett a dekódereknek van még egy rétegük az Encoder-Decoder Attention. Ez segít a dekódernek a bemeneti szekvencia megfelelő részeire koncentrálni. Az önfigyelem pedig, egy figyelmi mechanizmus, amely egyetlen szekvencia különböző pozícióit kapcsolja össze a szekvencia reprezentációjának kiszámítása céljából. [61]



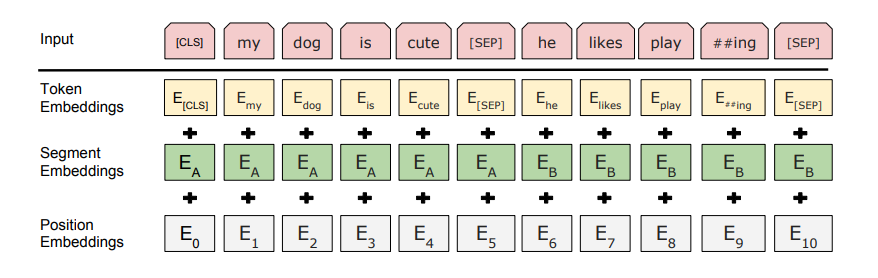
A fenti ábrán látható, hogy az „az” jelentése több dologra is hivatkozhat. Az önfigyelő felismeri, hogy az „az” ebben a kontextusba az „út” szót jelenti.

**Szöveg előfeldolgozása**

A BERT mögött álló fejlesztők egy meghatározott szabályrendszert adtak hozzá a modell beviteli szövegének képviseletéhez. Ezek közül sok olyan kreatív tervezési választás van, amely még jobbá teszi a modellt. [62]

Kezdetnek minden bemeneti beágyazás 3 beágyazás kombinációja:

* ***Pozíció beágyazások***: A BERT megtanul és használja a helyzeti beágyazásokat a szavak helyzetének kifejezésére egy mondatban. Ezeket azért adjuk hozzá, hogy túllépjük a transzformátor korlátozottságát, amely az RNN-től eltérően nem képes „szekvencia” vagy „sorrend” információkat elkapni.
* ***Szegmensbeágyazások***: A BERT mondatpárokat is bevihet a feladatokhoz (kérdés-válasz). Ezáltal megtanul egy egyedi beágyazást az első és a második mondathoz, hogy segítsen a modellnek megkülönböztetni őket. A leni példában az EA-ként megjelölt összes token az A mondathoz tartozik (és hasonlóan az EB-hez is).
* ***Token beágyazások***: Ezek a beágyazások megtanulják az adott tokent a WordPiece token szókincséből.



12. ábra BERT bemeneti reprezentáció [59]

**Különböző BERT modellek összehasonlítása**

Összesen négy modellt fogok összehasonlítani, amik a következők:

1. bert-large-uncased-whole-word-masking-finetuned-squad
2. distilbert-base-uncased-distilled-squad
3. bert-base-uncased-squadv1-x1.16-f88.1-d8-unstruct-v1
4. bert-base-uncased-squadv1-x2.32-f86.6-d15-hybrid-v1

Az első modellnél (bert-large) a nagy architektúra (24 transzformátor blokk, 16 attention-head) van alapul véve. Uncased, ami azt jelenti, hogy nincs megkülönböztetés kis és nagy kezdőbetűk között. Más BERT modellektől eltérően ezt a modellt egy új technikával tanították ki: whole world masking. Ebben az esetben a szónak megfelelő összes token egyszerre maszkolva van. Továbbá ezt a modellt finom hangolták a SQuAD adatkészleten. [59]

A második modell (distil-bert) a DistilBERT, ami kisebb és gyorsabb, mint a BERT, amelyet ugyanazon adatkészleten önellenőrzéssel előkészítettek és a BERT-base modelljét tanárként használva. Ez azt jelenti, hogy csak a nyers szövegeken tanították elő, emberi címkézés nélkül. Majd finomhangolása az SQuAD v1.1 adathalmaz segítségével. [63]

A következő modell (bert-prun-v1) az ***nn\_pruning*** python könyvtár segítségével hozták létre. Tehát neurális hálózat metszést hajtottak végre. A lineáris rétegek az eredeti súlyok 8.0%-át tartalmazzák. Az összes eredeti súlyok 28,0%-át tartalmazza (a beágyazások a modell jelentős részét teszik ki, és ez a módszer nem metszi őket). A lineáris mátrixok egyszerű átméretezésével 1.16x gyorsabb futást érhető el, mint a BERT-base modellnél. Pontosságát tekintve F1 értéke 88.11, szemben a BERT-base 88.5-tel, az F1-es csökkenés 0.39. [64]

Az utolsó modell (bert-prun-v2) hasonló csak kicsit nagyobb metszést hajtottak végre. A lineáris rétegek az eredeti súlyok 15.0% -át tartalmazzák. Az összes eredeti súlyok 34,0% -át tartalmazza. Az átméretezésével 2.32x olyan gyorsan futott, mint a BERT-base modell. [65]

A teljesítményüket az alábbi táblázatban vetettem össze az alábbi kérdéssel: (1) „What are the different OS for personal computers?”, (2) „What does operating system coordinates?”, (3) „What is necessary for a programs to run?”.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | bert-large | distil-bert | bert-prun-v1 | bert-prun-v2 |
| 1. kérdés | 0.716 | 0.982 | 0.957 | 0.964 |
| 1. kérdés | 0.191 | 0.689 | 0.678 | 0.391 |
| 1. kérdés | 0.056 | 0.048 | 0.109 | 0.205 |

Látható, hogy az esetemben nagy architektúrával rendelkező modellt könnyedén fölül teljesítik a modellek. Két esetben a jobban metszett modell jobban teljesít a kevésbé metszett párjával szemben. A természetes nyelvfeldolgozásnál is előtérbe kerülnek a metszett modellek. Ezáltal erőforrást tudunk megtakarítani csekély teljesítmény csökkenéssel vagy akár nagyobb teljesítmény növekedéssel (táblázat 3. sorra).

# Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | T. M. M. Arthur Samuel, „Machine\_learning: Wikipedia,” 16 December 2020. [Online]. Available: https://en.wikipedia.org. |
| [2] | D. Cielen és A. Meysman, Introducing Data Science, New York: Manning Publications, 2016. |
| [3] | P. Blanes, „Search: Flaticon,” 2019. [Online]. Available: https://www.flaticon.com. |
| [4] | A. Adesina, „Data is the new oil: Medium,” 13 Nov 2018. [Online]. Available: https://medium.com/. |
| [5] | M. Heller, „What is machine learning? Intelligence derived from data: Infoworld,” 15 May 2019. [Online]. Available: https://www.infoworld.com/. |
| [6] | A. Geron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, USA: O'Reilly Media, 2019. |
| [7] | F. Chollet, „Why choose Keras?: keras.io,” Google, 22 March 2020. [Online]. Available: https://keras.io/. |
| [8] | S. Weidman, Deep Learning from Scratch, USA: O’Reilly Media, 2019. |
| [9] | A. C. Müller és S. Guido, Introduction to Machine Learning with Python, USA: O’Reilly Media, 2017. |
| [10] | D. Silver, „Teaching: david silver,” University College London, 31 May 2015. [Online]. Available: https://www.davidsilver.uk. |
| [11] | E. Ameisen, Building Machine Learning Powered Applications, USA: O'Reilly Media, 2020. |
| [12] | P.-N. Tan, M. Steinbach és V. Kumar, Introduction to Data Mining, Edinburgh Gate: Pearson Education, 2014. |
| [13] | J. Grus, Data Science from Scratch, USA: O'Reilly Media, 2015. |
| [14] | S. H. Mohit Gupta, „Introduction Data Machine Learning: Geeks for geeks,” 17 May 2020. [Online]. Available: https://www.geeksforgeeks.org/. |
| [15] | T. Naeem, „Blog: Astera website,” 4 November 2020. [Online]. Available: https://www.astera.com/type/blog/structured-semi-structured-and-unstructured-data/. |
| [16] | W. Badr, „Top Sources For Machine Learning Datasets: Towards Data Science,” 13 Jan 2019. [Online]. Available: https://towardsdatascience.com. |
| [17] | H. Timothy, „Blog: Great learning web site,” 11 May 2020. [Online]. Available: https://www.mygreatlearning.com/blog/sources-for-analytics-and-machine-learning-datasets/. |
| [18] | N. Noy, „Blog: Google,” 5 Sep 2018. [Online]. Available: https://www.blog.google/products/search/making-it-easier-discover-datasets/. |
| [19] | C. N. Knaflic, Storytelling with data, New Jersey: John Wiley & Sons, Inc., 2015. |
| [20] | W. McKinney, Python for Data Analysis, USA: O’Reilly Media, 2017. |
| [21] | A. Zheng és A. Casari, Feature Engineering for Machine Learning, USA: O’Reilly Media, 2018. |
| [22] | W. Badr, „Why feature correlation matters a lot: Towards datascience,” 18 Jan 2019. [Online]. Available: https://towardsdatascience.com/. |
| [23] | J. Brownlee, „Data, Learning and Modeling: Machine learning mastery,” 6 January 2017. [Online]. Available: https://machinelearningmastery.com. |
| [24] | A. Thapliyal, „Univariate bivariate and multivariate data and its analysis: Geeks for geeks,” 14 August 2018. [Online]. Available: https://www.geeksforgeeks.org/. |
| [25] | H. I. Rhys, Machine Learning with R, the tidyverse, and mlr, New York: Manning Publications, 2020. |
| [26] | Q. E. McCallum, Bad Data Handbook, USA: O’Reilly Media, 2013. |
| [27] | J. Brownlee, „How to Remove Outliers for Machine Learning: Machine learning mastery,” 25 April 2018. [Online]. Available: https://machinelearningmastery.com/. |
| [28] | „Scikit-learn,” 20 November 2020. [Online]. Available: https://scikit-learn.org/. |
| [29] | D. Yadav, „Categorical Encoding using Label Encoding and One Hot Encoder,” 9 Dec 2019. [Online]. Available: https://towardsdatascience.com/. |
| [30] | „Pandas,” 2020. [Online]. Available: https://pandas.pydata.org/. |
| [31] | D. Kabakchieva, „Student performance prediction by using data mining classification algorithms,” *International journal of computer science and management research,* %1. kötet1, pp. 686-690, 2012. |
| [32] | B.-H. a. V. E. a. G. V. Kim, „GritNet: Student performance prediction with deep learning,” *arXiv preprint arXiv:1804.07405,* 2018. |
| [33] | Y. a. L. Q. a. L. Q. a. H. Su, „Exercise-enhanced sequential modeling for student performance prediction,” *Proceedings of the AAAI Conference on Artificial Intelligence,* %1. kötet32, 2018. |
| [34] | J. Rocca, „Introduction to Markov chains: Towards datascience,” 24 Feb 2019. [Online]. Available: https://towardsdatascience.com/brief-introduction-to-markov-chains-2c8cab9c98ab. |
| [35] | H. Lamba, „Intuitive Understanding of Attention Mechanism in Deep Learning : Towards datascience,” 20 March 2019. [Online]. Available: https://towardsdatascience.com/intuitive-understanding-of-attention-mechanism-in-deep-learning-6c9482aecf4f. |
| [36] | C. King, „Neocortical sparsity, active dendrites, and their relevance in robust sequence learning models,” *CSE 599B: “AI and The Brain”,* 2020. |
| [37] | M. Klear, „Towards Data Science: The Sparse Future of Deep Learning,” 25 November 2018. [Online]. Available: https://towardsdatascience.com/the-sparse-future-of-deep-learning-bce05e8e094a. |
| [38] | S. a. P. J. a. T. J. a. D. W. J. Han, „Learning both weights and connections for efficient neural networks,” *arXiv preprint arXiv:1506.02626,* 2015. |
| [39] | P. B. B. L. Alan Chiao, „Model Optimization - Pruning: Tensorflow,” 11 September 2020. [Online]. Available: https://www.tensorflow.org. |
| [40] | M. a. G. S. Zhu, „To prune, or not to prune: exploring the efficacy of pruning for model compression,” *arXiv preprint arXiv:1710.01878,* 2017. |
| [41] | D. C. a. M. E. a. S. P. a. N. P. H. a. G. M. a. L. A. Mocanu, „Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science,” *Nature communications,* %1. kötet9, pp. 1--12, 2018. |
| [42] | F. Lagunas, „Is the future of neural networks sparse an introduction 1/n: Medium,” 4 February 2020. [Online]. Available: https://medium.com/. |
| [43] | S. Gray, A. Radford és D. Kingma, „Block-Sparse GPU Kernels: OpenAI,” 6 December 2017. [Online]. Available: https://openai.com/. |
| [44] | L. Souza, „The Case for Sparsity in Neural Networks, Part 1: Pruning: Numenta,” 30 August 2019. [Online]. Available: https://numenta.com/. |
| [45] | A. Pai, „Comprehensive Guide to Text Summarization : Analytics Vidhya,” 10 June 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/. |
| [46] | Shrivarsheni, „Text summarization approaches NLP example: Machine learning plus,” 24 October 2020. [Online]. Available: https://www.machinelearningplus.com/nlp/text-summarization-approaches-nlp-example/. |
| [47] | T. Ganegedara, „Intuitive guide to understanding KL divergence: Towards Data Science,” 1 May 2018. [Online]. Available: https://towardsdatascience.com. |
| [48] | C. Delangue és J. Chaumond, „Transformers: Hugging Face,” The Hugging Face Team, 2020. [Online]. Available: https://huggingface.co/transformers/index.html. |
| [49] | P. v. Platen, „How to generate text: Hugging face,” Hugging face, 18 March 2020. [Online]. Available: https://huggingface.co/blog/how-to-generate. |
| [50] | R. K. Singh, „Generating text summaries GPT2: Paper space,” August 2020. [Online]. Available: https://blog.paperspace.com/generating-text-summaries-gpt-2/. |
| [51] | X. Zhou, Y. Zhang, L. Cui és D. Huang, „Evaluating Commonsense in Pre-trained Language Models,” *Proceedings of the AAAI Conference on Artificial Intelligence,* %1. kötet34, pp. 9733--9740, 2020. |
| [52] | P. Rajpurkar, „The Stanford Question Answering Dataset: Github.io,” 3 April 2017. [Online]. Available: https://rajpurkar.github.io/mlx/qa-and-squad/. |
| [53] | H. Nakayama, „Building and application of question answering system from scratch: Towards datascience,” 9 Jul 2020. [Online]. Available: https://towardsdatascience.com/building-an-application-of-question-answering-system-from-scratch-2dfc53f760aa. |
| [54] | D. Jurafsky és J. H. Martin, Speech and Language Processing, California: Stanford University, 2020. |
| [55] | H. a. Z. H. a. C. W. B. Padigela, „Investigating the successes and failures of BERT for passage re-ranking,” *arXiv preprint arXiv:1905.01758,* 2019. |
| [56] | A. a. P. A. a. B. B. Trotman, „Improvements to BM25 and language models examined,” *Proceedings of the 2014 Australasian Document Computing Symposium,* pp. 58--65, 2014. |
| [57] | T. Mills, „AI.io: Medium web site,” 3 Jul 2019. [Online]. Available: https://medium.com/ai-io/why-big-data-and-machine-learning-are-important-in-our-society-b4e708d2c654. |