ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS
INFORMATIKAI KAR**

SZAKDOLGOZAT

OE-NIK Hallgató neve: **Gonda Gréta**

2023 Hallgató törzskönyvi száma: **T/006766/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertervezés és -fejlesztés Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Gonda Gréta**
Törzskönyvi száma: T/006766/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Szülés idejének előrejelzése gépi tanulással méhizom-aktivitás alapján
Birth time prediction based on uterus-activity using machine learning

Intézményi konzulens: Dr. Kertész Gábor
Külső konzulens:

Beadási határidő: 2022. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Szoftvertervezés és –fejlesztés
specializáció

A feladat

Vajúdás során a méh-izom aktivitása alkalmas előrejelezni, hogy mikorra várható a szülés pontos ideje. A dolgozat célja egy olyan gépi tanulási modell tervezése és implementálása, amely idősoros izom-aktivitási adatok alapján képes prediktálni a szülés bekövetkeztét.
A feladat részeként olyan szintetikus adatokat állítson elő, ami az egyes fájások idejét és hosszát tartalmazza. Prediktáláskor a vajúdáshoz tartozó, adott időpontig rendelkezésre álló adatokat használja és eredményként a gyermek megszületésének várható időpontját adja vissza.
A megvalósítás előtt vizsgálja meg az idősoros adatokon alapuló előrejelző módszereket, az irodalomkutatás eredményei alapján dolgozza ki a predikciós módszert és implementálja a megoldását!

A dolgozatnak tartalmaznia kell:

- a kapcsolódó szakirodalom részletes áttekintését, a kapcsolódó gépi tanulási módszerek bemutatását,
- hasonló problémák, hasonló módszerek bemutatását,
- az elérhető adatok bemutatását, analízisét,
- a megoldás részletes tervét, a módszertan leírását,
- a megoldás implementációjának bemutatását, az eredmények részletes kiértékelését,
- a további kutatási-fejlesztési lehetőségek bemutatását.

Ph.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

.....
Dr. Vámossy Zoltán
intézetigazgató

A szakdolgozat elévülésének határideje: **2024. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022.12.09.

Sanda Sretz
.....
hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve:	Neptun kód:	Tagozat:
Gonda Gréta	[redacted]	Nappali

Telefon: Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Szülés idejének előrejelzése gépi tanulással méhizom-aktivitás alapján

Szakdolgozat / Diplomamunka[2] címe angolul:

Birth time prediction based on uterus-activity using machine learning

Intézményi konzulens: Külső konzulens:

Dr. Kertész Gábor

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.02.21.	KONZULTÁCIÓ	
2.	2022.03.30.	KONZULTÁCIÓ	
3.	2022.04.28.	KONZULTÁCIÓ	
4.	2022.05.02.	KONZULTÁCIÓ	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2022.05.02.

.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Gonda Gréta

Neptun kód: [redacted]

Tagozat: Nappali

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Szülés idejének előrejelzése gépi tanulással méhizom-aktivitás alapján

Szakdolgozat / Diplomamunka[2] címe angolul:

Birth time prediction based on uterus-activity using machine learning

Intézményi konzulens: Dr. Kertész Gábor

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.09.14.	KONZULTÁCIÓ	
2.	2022.10.26.	KONZULTÁCIÓ	
3.	2022.11.14.	KONZULTÁCIÓ	
4.	2022.12.06.	KONZULTÁCIÓ	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 20.22.12.06.....

.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

Tartalomjegyzék

1. Bevezetés

A dolgozat célja gépi tanulás segítségével egy olyan előrejelző modellt létrehozni, amely alkalmas a szülés alatti méhösszehúzódások időbeli eloszlása alapján prediktálni az újszülött világra jövetelének pontos idejét.
A modern orvostudomány eszköztárában ma már egyre inkább elterjedt a gépi tanulás segítségével támogatott diagnosztika, ami nem csak, hogy megkönnyíti és meggyorsítja az orvosok munkáját, de rendkívül pontos és megbízható eredményeket képes adni. Nem csoda tehát a technológia rohamos térhódítása a gyógyászatban, hiszen integrálva ezen megoldásokat a véges egészségügyi erőforrások a lehető leghatékonyabban használhatók fel.
Mindezen fejlődéssel párhuzamosan egy teljesen új páciens kultúra is kibontakozóban van. Az interneten keresztül azonnal rendelkezésre álló, végtelen mennyiségű információnak köszönhetően a páciensek egyre tájékozottabbak, egyre aktívabb szerepet kívánnak kivenni saját diagnosztikájukból és kezelésükből, amihez a technológiai megoldások is már egyre széleskörűbben elérhetőek. Az egyre hordozhatóbb és olcsóbb eszközök lehetőséget biztosítanak a kórházakon kívüli adatgyűjtésre és elemzésre, ezzel együtt a személyre szabott gyógyászat megvalósítására. Az egyre inkább elterjedt kifejezés mhealth (mobile health) azaz mobil egészségügy magába foglal minden olyan mobil technológiát, amit az egészségügyi gyakorlatban használt.
Az orvosi diagnosztika már jóval a kórházba érkezés vagy akár a panaszok megjelenése előtt kezdetét veheti az okostelefonok által gyűjtött egészségügyi adatokat felhasználva, amik az orvossal való első találkozáskor fontos információkkal szolgálhatnak a megfelelő ellátás megkezdéséhez. Az összegyűjtött információk diagnosztikai értékéhez azonban elengedhetetlen, a megfelelő kiértékelő szoftverek megléte.
Már jóval az okostelefonok megjelenése előtt bevett szokás volt, a vajúdás kezdetétől az összehúzódások idejének és a hosszúságának a mérése és feljegyzése, azzal a céllal, hogy egyrészt információval szolgálhasson az orvosnak a szülés előrehaladásáról másrészt, hogy az orvos által meghatározott fájási sűrűségkor érkezhessen a nő a kórházba. Ma már a mérési folyamat egyszerűbbé tételéhez, megannyi mobil applikáció áll rendelkezésre, azonban ezek az összegyűjtött adatok értékelését a felhasználókra bízzák.
Célom létrehozni egy olyan prediktív modellt, ami az összegyűjtött adatok alapján képes becslést adni a szülés pontos idejére és a későbbiekben akár az adatgyűjtési folyamatot a kórházban is folytatva lehetőséget biztosítani a kórházi személyzet számára a lehető legfelkészültebb ellátás biztosítására.

2. Háttér

2.1 A szülés mechanizmusa

A vajúdás a magzat hüvelyen keresztüli világrahozatalát jelenti. A sikeres szülés három tényezőt foglal magában: az anya erőfeszítéseit és a méh összehúzódásait, a magzati jellemzőket és a kismedence anatómiáját. A klinikai gyakorlatban többféle módszert is alkalmazhatnak a vajúdás előrehaladásának figyelemmel kísérésére.
Rendszeres hüvelyi vizsgálatokat használnak a méhnyak tágulásának és a magzat helyzetének meghatározására. Magzati szívmonitorozást alkalmaznak a magzat egészségi állapotának felmérésére. A kardiotokográfiát (CTG) a kontrakciók gyakoriságának és erősségének nyomon követésére használják. Az egészségügyi szakemberek a vizsgálatok során szerzett információkat felhasználják a vajúdási fázis meghatározására és a vajúdás előrehaladásának figyelemmel kísérésére. [1]

2.1.1 *A szülés I. szakasza – Tágulási szakasz*

A tágulási szakaszon belül két fázist különítünk el, a látens és az aktív fázist. A látens fázis során a méhnyak rövidülés és korai tágulása (3 cm-ig) történik, az aktív tágulás alatt felgyorsul a méhnyak tágulása. Az első gyermeküket szülő nők méhszáj tágulása átlagosan 1,2 cm/óra, a többgyermekes anyáké 1,5 cm/óra, de ezektől jelentős eltérések lehetnek. Általánosan elmondható, hogy első gyermekeseknél a tágulási szakasz 9-11 óra, többgyermekeseknél 4,5-5,5 óra. [2]
Kórházban történő szülés esetében ezen szakasz alatt történik a kórházba utazás, az azonban, hogy a szülés megindulása után mennyivel is kell elindulni a kórházba több tényezőtől is függhet. Általános orvosi ajánlás, hogy a magzatvíz elfolyásakor érdemes felkeresni a kórházat, azonban ha a magzatvíz még nem folyt el, de a fájások már elkezdődtek, akkor érdemes várni amíg azok rendszeresek és erősek lesznek, valamint nagyjából egymástól 5 perc távolságra 60 másodperc hosszan kezdenek jelentkezni. [3]

2.1.2 *A szülés II. szakasza – Kitolási szakasz*

A kitolási szakasz a magzat megszületéséig tart, hossza elsőszülőknél átlagosan 50-60 perc, többedik alkalommal szülőknél 25-30 perc. [2] Azoknál a nőknél, akiknek ez az első szülése, a vajúdás ezen szakasza általában kevesebb mint három órát, azoknál, akik már szültek korábban, kevesebb mint két órát vesz igénybe. [1]

2.1.3 *A szülés III. szakasza – Placentáris szakasz*

A placentáris szakasz a magzat megszületésekor kezdődik, és a placenta távozásáig tart. Ennek a szakasznak a hossza nagyjából 5-22 perc. [2]

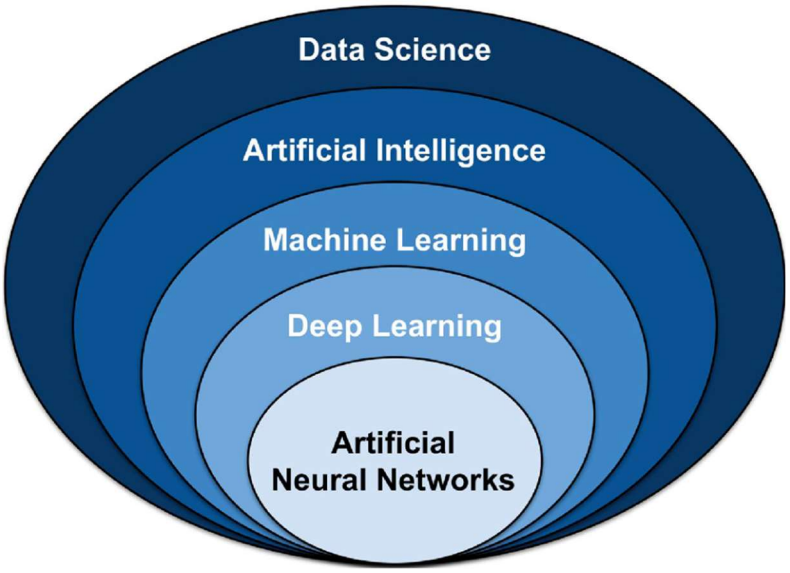
Az egyes szakaszok statisztikai jellemzőit az alábbi táblázat foglalja össze:

Szülési szakasz vagy fázis	Frekvencia (perc)	Kontrakció hossz (másodperc)	Pihenő szakasz hossza (perc)	Összidő elsőszülőknél	Összidő többgyermekeseknél
Látens fázis (I. szakasz)	5-10	25-40	5-30	9-11 óra	4,5-5,5 óra
Aktív fázis (I. szakasz)	3-5	40-60	3-5		
II. szakasz	2-3	60-90	0,5-2	50-60 perc	25-30 perc

2.1. táblázat - Méhösszehúzódások a vajúdás különböző szakaszaiban [4] [5]

3. Alkalmazott módszertan

Fontos tisztázni az Artifical Intelligence - AI (Mesterséges Intelligencia), Machine Learning - ML (Gépi tanulás), Deep Learning - DL (Mélytanulás), Neural Networks (Neurális hálózatok) fogalmait, ugyanis ezek kontextusa gyakran hasonló. Az Artifical Intelligence egy olyan terület, amely az emberek által általában végzett intellektuális feladatok automatizálására összpontosít. A Machine Learning, Deep Learning és a Neural Networks pedig specifikus módszerek e cél elérésére, az utóbbiakkal szemben azonban az AI nem tartalmaz semmiféle „tanulást”. A Machine Learning, Deep Learning és a Neural Networks az AI területén belül helyezkednek el, továbbá a Deep Learning valójában a Machine Learning, a Neural Networks pedig a Deep Learning részterülete. [6]



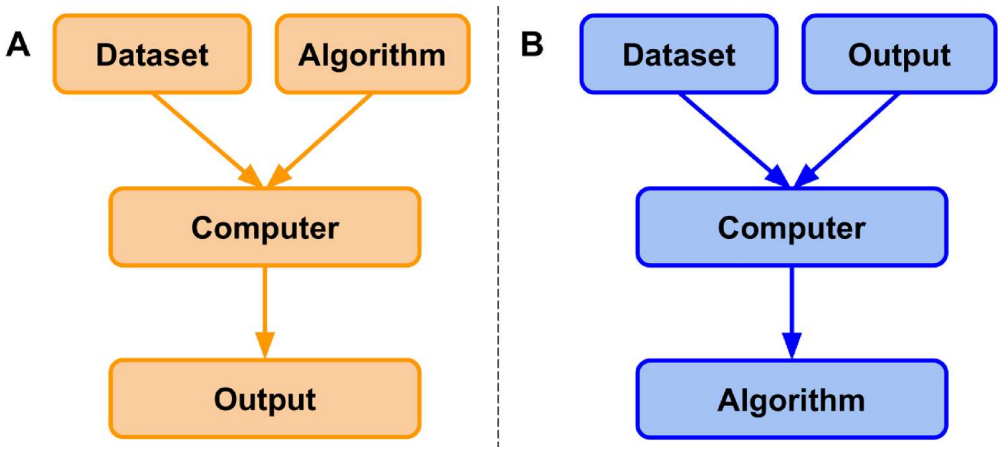
3.1. ábra - Ábra a Data Science-en belüli tudományterületek egymáshoz viszonyított helyzetéről. [6]

Az AI jól alkalmazható a világosan meghatározott logikai problémák megoldásában, azonban gyakran elbukik azoknál a feladatoknál, amelyek magasabb szintű mintafelismerést igényelnek. Ezeknél a bonyolultabb feladatoknál ugyanakkor az ML és a DL módszerek jól teljesítenek. A mély tanulás és a gépi tanulás az egyes algoritmusok tanulási mechanizmusában különböznek. A mély tanulás automatizálja a speciális jellemzők kiemelési folyamatának nagy részét, kiküszöbölve a szükséges emberi beavatkozások egy részét, és lehetővé téve nagyobb adathalmazok használatát. A klasszikus Machine Learning tanulása jobban függ az emberi beavatkozástól, ugyanis támaszkodik arra, hogy az emberi szakértők meghatározzák a jellemzőket a bevitt adatok közötti különbségek megértéséhez. Általánosságban elmondható, hogy strukturáltabb adatokra van szükségük a tanuláshoz.

„Deep” Machine Learning-nél is alkalmazhatóak felcímkézett adathalmazok, ez az úgynevezett „supervised learning” (felügyelt tanulás). Az algoritmus azonban nyers formában is befogadhat strukturálatlan adatokat, és automatikusan meghatározhatja azokat a tulajdonságokat, amelyek megkülönböztetik az adatok különböző kategóriáit. A gépi tanulással ellentétben ez nem igényel emberi beavatkozást az adatok feldolgozásához, így a gépi tanulást nagyobb méretű adathalmazokon is alkalmazhatjuk. A mély tanulás és az neurális hálók elsősorban az olyan területek gyors előrehaladásához járulnak hozzá, mint a számítógépes látás, a természetes nyelvek feldolgozása (Natural Language Processing, NLP) és a beszédfelismerés.
A neurális hálózatok neuron rétegekből állnak, amelyek egy bemeneti réteget, egy vagy több rejtett réteget és egy kimeneti réteget tartalmaznak. Minden csomópont vagy mesterséges neuron összekapcsolódik egy másikkal, és ehhez társul a súlya és a küszöbértéke. Ha bármelyik csomópont kimenete meghaladja a megadott küszöbértéket, akkor ez a csomópont aktiválódik, és adatokat küld a hálózat következő rétegének. Ellenkező esetben nem továbbítanak adatokat a hálózat következő rétegére. A mély tanulás „mélye” csupán az ideghálózat rétegeinek mélységére utal. Az a neurális hálózat, amely háromnál több rétegből áll - amely magában foglalja a bemeneteket és a kimeneteket - mély tanulási algoritmusnak vagy mély neurális hálózatnak tekinthető. Az a neurális hálózat, amelynek csak két vagy három rétege van, az egy alap neurális hálózat. [6] [7]

3.1 Machine Learning bemutatása

Az Machine Learning olyan terület, amely az AI tanulási aspektusára összpontosít olyan algoritmusok kifejlesztésével, amelyek a legjobban reprezentálják az adathalmazt. Ellentétben a klasszikus programozással (A. ábra), amelyben egy algoritmus expliciten kódolt az ismert jellemzők felhasználásával, az Machine Learning az adatok részhalmazait használja olyan algoritmusok létrehozásához, amelyek a jellemzőknek és súlyoknak új kombinációit használják, a probléma megoldására. (B ábra). [6]

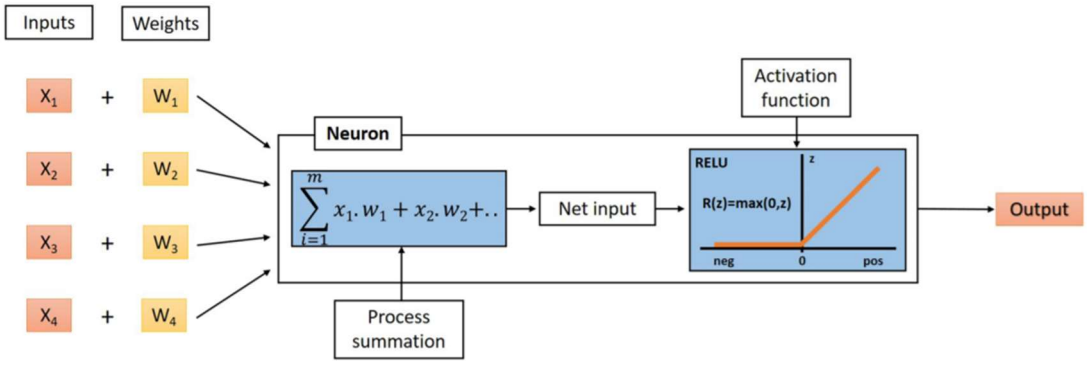


3.2. ábra - A klasszikus programozás összehasonlítása a Machine Learning-el [6]

Az ML-ben négy általánosan alkalmazott tanulási módszer létezik, amelyek mindegyike különböző feladatok megoldására hasznos: felügyelt, felügyelet nélküli, félig felügyelt és megerősítéses tanulás. Felügyelt tanulás akkor fordulhat elő, ha a neurális hálózat tanításához használt adatokat felcímkézik, vagyis a kimenet (pl. mortalitás egy bináris osztályozási probléma esetén) ismert. Ha a kimeneti címke ismeretlen, akkor felügyelet nélküli megközelítésre van szükség. Példaként „clustering" (csoportosítás) alkalmazható az adatok csoportjainak azonosítására, vagy dimenziócsökkentés az adatok kevésbé összetett ábrázolásának előállítására. A félig felügyelt tanulásként ismert harmadik kategória felcímkézett adathalmazokat használ a felügyelet nélküli tanulás eredményeinek javítása érdekében. A megerősítéses gépi tanulás pedig egy a felügyelt tanuláshoz hasonló módszer, azonban az algoritmus nem előre specifikált adatok segítségével tanul, hanem próbálkozás útján oly módon, hogy a sikeres eredmények sorozatai megerősítésre kerülnek a lehető legoptimálisabb szabályrendszer előállításának érdekében. [7] [8]

3.1.1 *A felügyelt gépi tanulás*

Felügyelt tanulás során a neurális hálózat az egyik rétegben lévő neuronokból származó bemeneti adatokat egymás után továbbítja a következő réteg neuronjaihoz egy olyan folyamat során, amely sokszor, gyakran többezerszer megismétlődik. Egy ciklust vagy iterációt „epoch"-nak (korszaknak) nevezünk. Minden lépés során a kimeneti információk továbbításra kerülnek a következő rétegbe. Minden neuron több más neuronból fogad súlyozott bemeneteket, ezek a bemeneti értékek összegződnek és átadásra kerülnek egy belső aktivációs függvénynek. Ha a kapott érték túllépi az aktiválási küszöböt, akkor az adott neuron kimenetet generál, amely egy súlyértékeket kap a következő rétegben lévő neuronokhoz való továbbítás előtt. Ha az érték nem lépi át a küszöböt, akkor a kimenet nulla lesz. [8] Az ilyen módon felépülő mesterséges neuron a neurális háló alapelemét képezi, más néven perceptron-nak is hívják és egy egyrétegű neurális hálónak tekinthető. [9]



3.3. ábra - Neuron működésének összefoglalása [8]

A neurális háló által szerzett ismeret a súlyok értékeiben kerül tárolásra, egy nagyobb méretű hálózat akár több millió súllyal is rendelkezhet. Miután az információ az előbb leírt módon átjutott minden rétegen, a modell kimenetet generál, ami aztán

összehasonlításra kerül a címkézéskor megadott elvárt értékkel. A kapott érték és az elvárt érték eltérése alapján létrejött hiba szerint frissítésre kerülnek a súly értékek. A megadott számú tanulási iterációk során a modell célja a hiba minimalizálása és a legkisebb hibaértéket generáló súlyértékek kombinációjának megteremtése. A súlyok ismétlődő frissítése a hiba nagysága alapján az, amit „tanulásnak” nevezünk. [8]
A felügyelt tanulás leggyakoribb alkalmazási területei:

- Regresszión vagy kategorikus osztályozáson alapuló prediktív elemzés
- Mintázat felismerés
- NLP (Natural Language Processing)
- Kép klasszifikáció
- Szekvencia analízis [10]

3.1.2 *A felügyelet nélküli gépi tanulás*

Nem felügyelt tanításkor, a minták nincsenek felcímkézve, ezáltal a hálózatnak nem előre megírt viselkedést kell megtanulnia, hanem a bemenetei és a kimenetei alapján, az elvárt válaszok ismerete nélkül kell egy szabályrendszert kialakítania. Ekkor azonban nincsen viszonyítási alap, amihez mérni lehetne a hálózat helyességét. A vizsgálat tárgya így tehát az lesz, hogy a minták mennyire hasonlítanak vagy mennyire különböznek egymáshoz képest, azaz kialakulnak-e olyan tartományok, ahol a minták sűrűsödnek, létrejönnek-e klaszterek.
A felügyelet nélküli tanulás két fő alkalmazási területe:

- Klaszterezés: Alapvetően a klaszterezés célja különböző csoportok megtalálása az adathalmaz elemein belül. Ehhez a klaszterező algoritmusok úgy találják meg a struktúrát az adatokban, hogy az egy klaszterbe (vagy csoportba) tartozó elemek jobban hasonlítsanak egymásra, mint a más klaszterekből származó elemekre.
- Dimenzió redukció: Az adathalmaz dimenzióját a bemeneti adatok száma adja. A dimenzió redukció csökkenti a bemeneti adatok számát, miközben a fontos információkat kiemeli és megtartja, ezzel megkönnyítve a fennmaradó adatok későbbi elemzését. [11]

3.1.3 *A félig felügyelt gépi tanulás*

Félig felügyelt tanulás során mind felcímkézett, mind címkézetlen adatokat felhasználnak a háló tanításához. Jól alkalmazható olyankor, amikor nagy mennyiségű adatot kell kategorizálni, kis mennyiségben rendelkezésre álló címkézett adat segítségével. [10]

3.1.4 *A megerősítéses gépi tanulás*

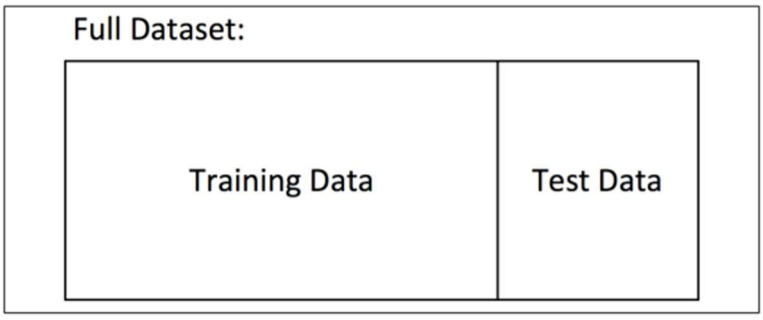
A megerősítéses tanulás lényege, hogy a környezettel való interakció révén történik tanulás, se címkéket se adatokat nem kap a rendszer. Ez a tanulási folyamat egy „actor”-t (cselekvőt), egy „environment”-t (környezetet) és egy „reward signal”-t (jutalmazó jelet) foglal magában. Az actor úgy dönt, hogy cselekszik az environment-ben, amiért ennek megfelelően megjutalmazzák. Azt, ami alapján az actor cselekedet választ „policy”

-nak (irányelvnek) nevezzük. Az actor célja a kapott jutalom növelése, ezért meg kell tanulnia az optimális irányelveket a környezettel való interakcióhoz. [12]
Az olyan automatizált rendszerek esetében érdemes ezt a módszert használni, ahol sok kisebb, emberi beavatkozástól mentes döntést kell hoznia egy rendszernek. Pl. Önvezető autó tervezése [13]

3.4. ábra - Megerősítéses tanulás vázlata [12]

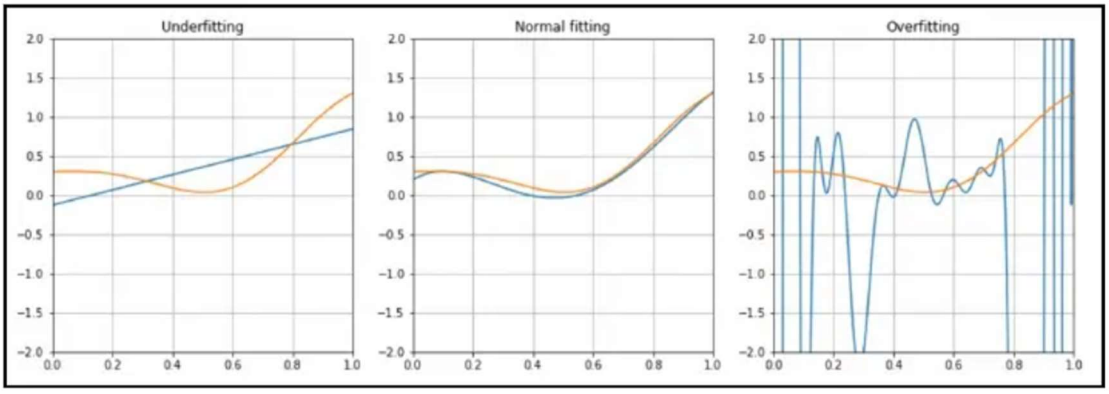
3.2 Eredmények kiértékelése

A gépi tanulás algoritmusainak kiértékeléséhez, a rendelkezésre álló adathalmaz általában két részre van bontva. Az egyik a tanító adatokat tartalmazza, amelyek a rendszer tanításához lesznek felhasználva, a másik a teszteléshez felhasznált adatokat, amikkel a rendszer a tanítás során nem fog találkozni. [6] A tanító és teszt adatok aránya általában 70-30, ahol a tanitó adatok az összes rendelkezésre álló minta 70%-át teszik ki, a teszt adatok pedig az összes minta 30%-át. [14]



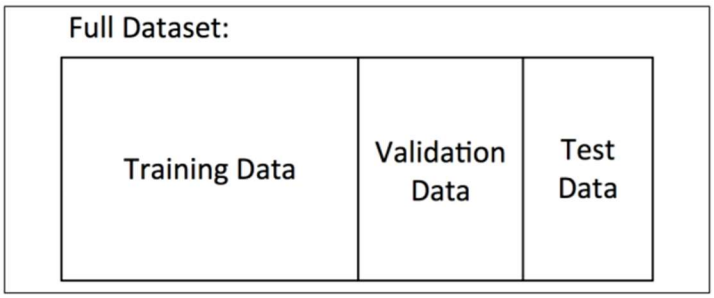
3.5. ábra - Rendelkezésre álló adatok felosztásának szemléltetése [12]

Ha a modell nagy százalékban jó eredményeket ad a tanító adatokon, de rosszul teljesít a teszt adatok esetében, akkor „overfitting”-ről, túltanulásról beszélhetünk. Ilyenkor az elvárt feladathoz képest a rendszer túl komplex és általános modell felállítása helyett, az rendszer memorizálja a mintákat. Ennek ellenkező esete az „underfitting”, az alultanulás, ekkor a modell mind a tanító, mind a teszt adatokon rosszul teljesít, nem áll fent a minták megjegyzése, de nem is sikerült modellt felállítani az adatokra.



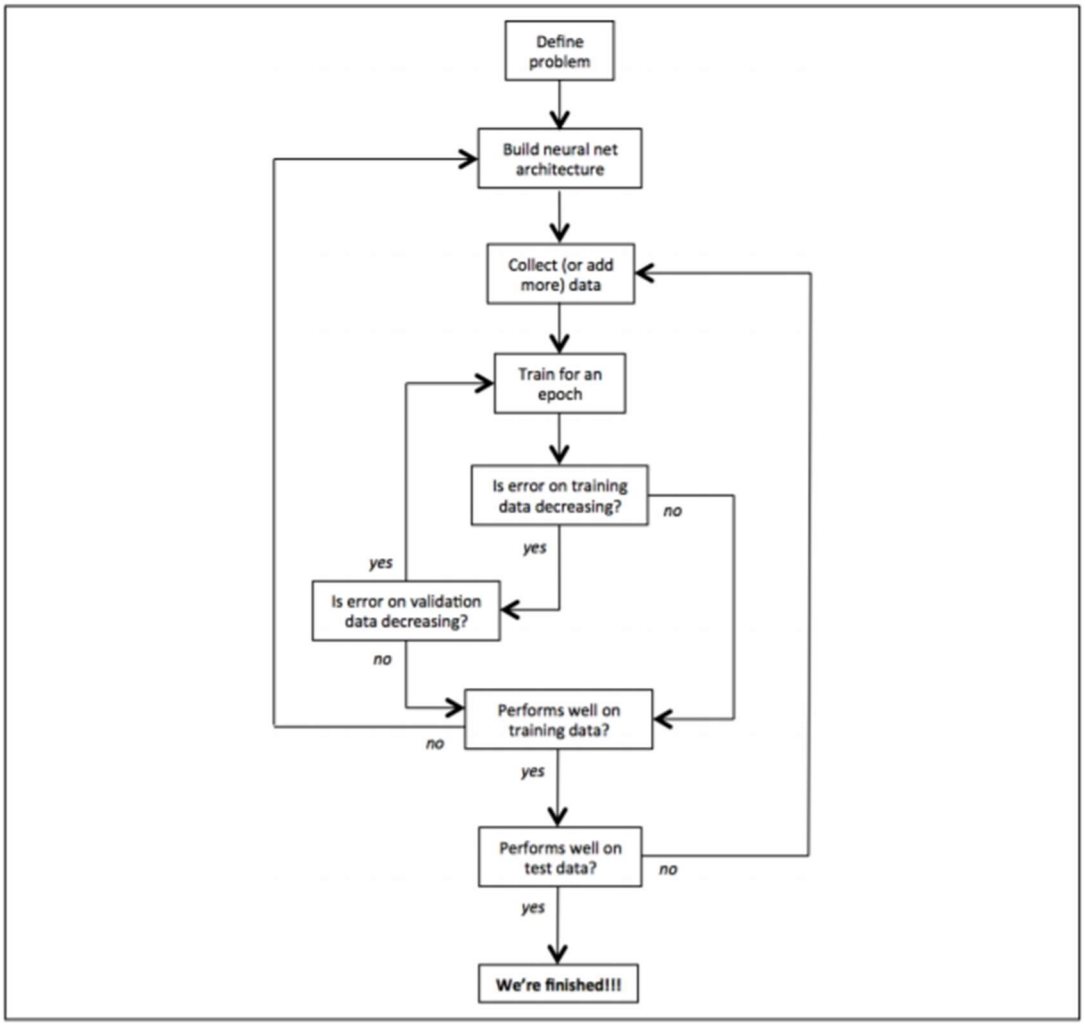
3.6. ábra - Példa Overfitting-re és Underfitting-re [10]

Overfitting megelőzhető a tanítás azelőtti leállításával, hogy az kialakulna, ekkor az epoch-ok számát kell csökkenteni. Underfitting esetében pedig a rendszer komplexitásának a növelése, illetve további tanító minták hozzáadása lehet a megoldás. Annak ellenőrzésére, hogy ne alakuljon ki overfitting, érdemes lehet a rendelkezésre álló adatok közül egy validációs adathalmazt is létrehozni, amely minden epoch után ellenőrzi, a modell pontosságát. Ha a pontosság a tanító mintákon nő miközben a validációs adatokon nem változik, akkor ez jelzi, hogy le kell állítani a tanítást, mert overfitting alakul ki. [12]



3.7. ábra - Adatok felosztása validációs minták alkalmazásával [12]

A machine learning modell tanításának és kiértékelésének a folyamatát az alábbi ábra foglalja össze:

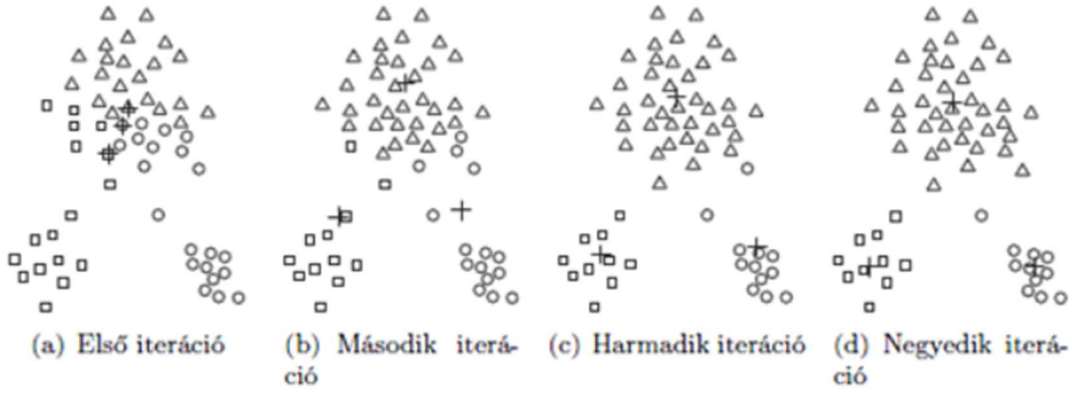


3.8. ábra - A machine learning modell tanításának és kiértékelésének részletes munkafolyamata [12]

3.3 Módszerek mintázatfelismeréshez és előrejelzéshez

3.3.1 *K-means clustering – K-közép klaszterelemzés*

A K-means clustering a nem felügyelt tanulás egyik algoritmusa. A „módszer egy középpontot (centroidot) választ ki prototípusnak, amely általában pontok egy csoportjának az átlaga, és jellemzően csak folytonos, n-dimenziós térben elhelyezkedő pontokra alkalmazható.” Az algoritmus K db kezdő középpont kiválasztásával kezd, ahol a K a felhasználó által kívánt klaszterek megadott száma. A kiinduló klaszterek létrehozásakor minden adat-pontot a hozzá legközelebb eső középponthoz rendelünk, majd a klaszter középpontját az újonnan a klaszterhez rendelt pontok alapján frissítjük. Ezen hozzárendelési ér középpont frissítési lépéseket addig folytatjuk, amíg egyetlen pont sem vált már klasztert, azaz a középpontok ugyanazok maradnak. [15] [16]



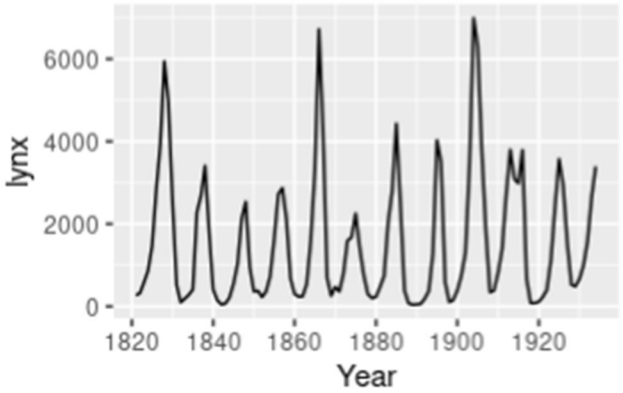
3.9. ábra - Példa három klaszter keresésére [15]

3.3.2 *Principal Component Analysis (PCA)– Főkomponens analízis*

A Principal component analysis-t arra használják, hogy a változók számának csökkentése által, az adatok könnyebben megvizsgálhatóak és vizualizálhatóak legyenek. Az adatok maximális eltérését rögzítik egy koordinátarendszerben, amelynek tengelyei a „fő-komponensek". Mindegyik komponens az eredeti változók lineáris kombinációja és merőlegesek egymásra. A komponensek közötti merőlegesség azt jelzi, hogy ezeknek az összetevőknek az összefüggése nulla. A PCA célja az, hogy:
(1) kivonja a legfontosabb információkat az adattáblából;
(2) tömörítse az adatkészlet méretét csak e fontos információk megőrzésével;
(3) egyszerűsíteni az adatkészlet leírását;
(4) elemezi a megfigyelések szerkezetét és a változókat [17]

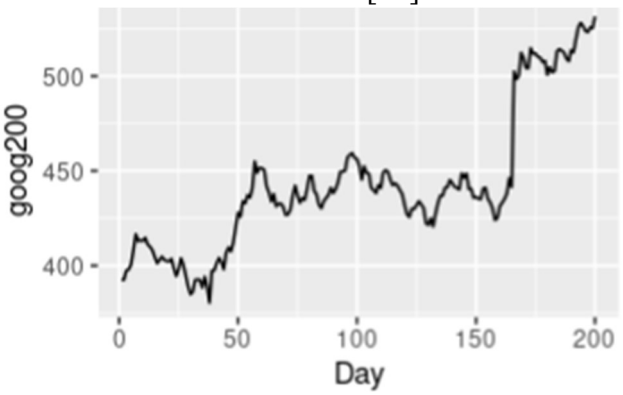
3.3.3 *Time Series Forecasting – Idősorok előrejelzése*

Idősor alatt megfigyelések egy sorozatát értjük. Amennyiben a megfigyelés egyetlen változó értékére irányul, akkor egyváltozós (univariate) idősorról beszélünk. Abban az esetben, ha több attribútum értékét is figyeljük az egymást követő időpontokban, akkor az idősor többváltozós (multivariate). [18]
Idősorok általában azonos időközökben vett adatokkal kerülnek előállításra, ezek az idősorokat nevezzük reguláris (szabályos) idősoroknak. Az adatok tartalmazhatnak explicit megadott időbélyegeket, vagy kiszűrhetőek az adatok létrejövetelének intervallumai alapján.
Az irreguláris idősorok ennek az ellentettjei. Az idősor adatai időbeli sorrendet követnek, de a mérések nem azonos időközönként történnek. [19]
Másik fontos jellemzője az idősoroknak, hogy stacionáriusak-e vagy nem stacionáriusak. Stacionárius idősor esetében az idősor tulajdonságai nem függenek a megfigyelés idejétől. Általánosságban elmondható, hogy az idősor mintázata hosszú távon nem lesz kiszámítható, az időiagramokon ábrázolt sorozat nagyjából vízszintes, állandó szórással.



3.10. ábra - Példa stacionárius idősor időiagramjára [20]

Ebből következik, hogy a trendekkel (iránnyal) vagy szezonalitással (periodikus komponenssel) rendelkező idősorok nem stacionáriusok, mivel a trend és a szezonalitás más időpontokban más hatással van az idősorra. [20]



3.11. ábra - Példa nem stacionárius idősor időiagramjára [20]

Az statisztikai előrejelző módszerek többsége azon alapul, hogy az idősorokat matematikai transzformációkkal megközelíthetőleg stacionáriussá lehet tenni. [21]
A nem stacionárius idősorok stacionáriussá tételének egyik módja, az egymást követő megfigyelések különbségeinek kiszámítása, azaz differenciálása. Az olyan transzformációk, mint például a logaritmus, segíthetnek az idősorok szórásának a stabilizálásában. A differenciálás pedig segíthet a középérték stabilizálásában azzal, hogy megszünteti az idősorban a szintbeli (a szint itt az értékek középértékét jelenti a sorozatban) változásokat, ezáltal eliminálva (vagy csökkentve) a trendet és a szezonalitást. [20] A stacionárius sorozatok pedig viszonylag könnyen felismerhetőek annak vizsgálatával, hogy a statisztikai tulajdonságok ugyanazok lesznek-e a jövőben, mint amik a múltban voltak. A stacionáriussá transzformált sorozat predikciói az előrejelzés után visszatranszformálhatóak, így megkapva az előrejelzésekét az eredeti sorozatra. [21]

3.4 Idősor előrejelzéshez használt módszerek

3.4.1 *Klasszikus idősor előrejelző módszerek*

3.4.1.1 *Integrált autoregresszív mozgóátlag modell - Autoregressive integrated moving average (ARIMA)*

Az autoregresszív integrált mozgóátlag (Autoregressive Integrated Moving Average - ARIMA) modell idősoros adatokat és statisztikai elemzést használ az adatok értelmezéséhez és a jövőbeli értékek előrejelzéséhez. Az ARIMA modell célja az adatok magyarázata az idősoros adatok múltbeli értékeinek felhasználásával, és az előrejelzés lineáris regresszió használatával.

Az ARIMA modell komponensei:

- Az „AR" az ARIMA-ban az autoregressziót jelent, ami azt jelzi, hogy a modell az aktuális adatok és azok múltbeli értékei közötti ok-okozati kapcsolatot használja.
- Az „I" az integrált kifejezést jelenti, ami azt jelenti, hogy az adatok stacionáriusok.
- Az „MA" a mozgóátlag modellt jelöli, ami azt jelzi, hogy a modell előrejelzése vagy eredménye lineárisan függ a múltbeli értékektől. Ez azt is jelenti, hogy az előrejelzés hibái a múltbeli hibák lineáris függvényei.

Az AR, I és MA komponensek mindegyike paraméterként szerepel a modellben. A paraméterekhez meghatározott egész értékek vannak hozzárendelve, amelyek az ARIMA modell típusát jelzik.

Az ARIMA paramétereinek általános jelölése [22]: ARIMA (p, d, q). A jelölésben használt paraméterek az alábbi változóknak felelnek meg:

- p = az autoregresszív tagok száma
- d = a differenciálások száma, ami a stacionáriussá tételhez szükséges [23]
- q = az előrejelzési hibák száma a modellben, avagy a mozgóátlag ablak mérete

A paraméterek egész értékeket vehetnek fel, és a modell működéséhez mindenképpen definiálni kell őket. 0 értéket is felvehetnek, ami azt jelenti, hogy nem használják őket a modellben.

3.4.1.2 *Holt's exponential smoothing vagy Double exponential smoothing – Holt féle exponenciális simítás*

Az exponenciális simítás egy idősor előrejelző módszer. Az exponenciális simítási módszerekkel előállított előrejelzések a múltbeli megfigyelések súlyozott átlagai. A súlyok exponenciálisan csökkennek, ahogy a megfigyelések öregszenek, tehát minél újabb a megfigyelés, annál nagyobb a társított súly. [23] Ennek egyik fajtája a Holt féle exponenciális simítás, amely olyan sorozatok előrejelzésére használható, ahol van trend, de nincsen szezonalitás.

A módszer leírásához alkalmazott paraméterek:

- alpha(a) – a szint simító együtthatója: Ez a paraméter szabályozza azt a sebességet, amellyel az előző időpontokban végzett megfigyelések hatása exponenciálisan csökken. Az alfa 0 és 1 közötti értékre van állítva. A nagy (1-hez közeli) értékek azt jelentik, hogy a modell főként a legfrissebb megfigyeléseket veszi figyelembe, míg a kisebb értékek azt jelentik, hogy az előrejelzés során a régebbi megfigyelések is nagyobb hatással vannak. [24]
- beta(b) - a trend simító együtthatója: Ez a paraméter szabályozza a trendváltozás hatásának csökkenését. Az előzőhöz hasonlóan ennek az értéke is 0 és 1 közötti értéket vehet fel, és a magasabb érték a frissebb információk magasabb súlyát jelentik.
- phi(p) – trend csillapító tényező: a trend változási sebességének csillapítására szolgáló paraméter.

A módszer kétféle módon változó trendet képes előrejelezni, ezek az additív és a multiplikatív attól függően, hogy a trend lineáris vagy exponenciális.

- additív trend: Holt féle exponenciális simítás lineáris trenddel.
- multiplikatív trend: Holt féle exponenciális simítás exponenciális trenddel.

A módszer egy előrejelző és két simító egyenletet használ.
Additive trend esetén:

- Az előrejelzés leírható az alábbi összefoglaló egyenlettel, ahol F_{t+k} a legutóbbi t időponttól k időegységre vett előrejelzés, L_t a legutóbbi becsült szint, T_t a legutóbbi trend, k pedig, hogy lépéssel próbálunk előrejelezni: $F_{t+k} = L_t + kT_t$
- A sorozat szintje és trendje a rendelkezésre álló adatok alapján kerül becslésre és új adatok elérhetősége esetén frissítésre kerül. A szint frissítésének egyenlete: $L_t = ay_t + (1\text{-}a)(L_{t-1} + T_{t-1})$.
- A trend frissítésének egyenlete: $T_t = b(L_t - L_{t-1}) + (1\text{-}b)T_{t-1}$.

multiplikatív trend esetén:

- Az előrejelzés egyenlete: $F_{t+k} = L_t * T^k_t$.
- A szintet frissítő egyenlet: $L_t = ay_t + (1\text{-}a)(L_{t-1} * T_{t-1})$.
- A trendet frissítő egyenlet: $T_t = b(L_t / L_{t-1}) + (1\text{-}b)T_{t-1}$. [24]

3.4.2 *Neurális hálózatokon alapuló idősor előrejelző módszerek*

3.4.2.1 *Rekurrens neurális hálózatok - Recurrent neural network (RNN)*

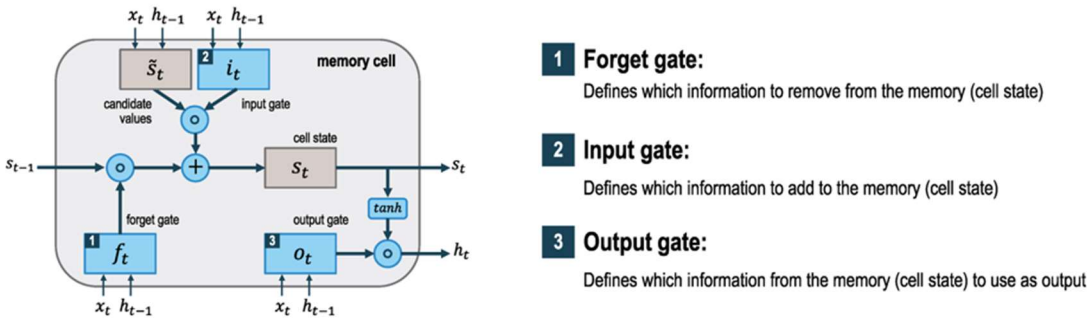
A rekurrens neurális hálózatok olyan szekvenciális adatokat tartalmazó neurális hálózatok, amelyek célja megfigyelések sorozatában előrejelezni a következő lépést, a sorozat előző lépései alapján. Az RNN-ek rejtett rétegeket tartalmaznak, amelyek lehetővé teszik az adatsor olvasásakor korábban már megszerzett információ tárolását.

Az egyszerű neurális hálózatok nem képesek a hosszú távú időbeli függőségek kezelésére. Ennek oka a vanishing gradient probléma, amire a LSTM azaz long short-term memory architektúra ad megoldást. Ennek implementációjakor a rejtett réteget

számítási egységek összetett blokkja váltja fel, amelyek kapukból állnak és csapdába ejtik a blokkban lévő hibát. [25]

LSTM – Long Short-Term Memory

Az long short-term memory hálózat a rekurrens neurális hálózat egy fajtája. Az LSTM hálózatokat kifejezetten a hosszú távú függőségek megtanulására tervezték, és képesek leküzdeni a RNN korábban felmerülő problémáit.



3.12. ábra - Egy LSTM memória blokk felépítése [26]

Az LSTM hálózatok egy bemeneti rétegből, egy vagy több memóriacellából és egy kimeneti rétegből állnak. A bemeneti rétegben lévő neuronok száma megegyezik a magyarázó változók (explanatory variables) számával. Az LSTM hálózatok fő jellemzője az úgynevezett memóriacellákból álló rejtett rétegben található.

Mindegyik memóriacellának három kapuja van, amelyek fenntartják és beállítják a cella állapotát(s_t):

- forget gate / felejtési kapu (f_t)
  - A felejtési kapu határozza meg, hogy milyen információ kerüljön eltávolításra a cella állapotából.
- input gate / bemeneti kapu (i_t)
  - A bemeneti kapu határozza meg, hogy milyen információ kerüljön hozzáadásra a cella állapotához.
- output gate / kimeneti kapu (o_t)
  - A kimeneti kapu határozza meg, hogy a cella állapotából milyen információt használnak fel.

A szekvenciális frissítés képletei a következők:

- input node / bemeneti csomópont:

$$g^{(t)} = \tanh(W_{gx}x^{(t)} + W_{gh}h^{(t-1)} + b_g)$$

- input gate / bemeneti kapu:

$$i^{(t)} = \sigma(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + b_i)$$

- forget gate / felejtési kapu:

$$f^{(t)} = \sigma(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + b_f)$$

- output gate / kimeneti kapu

$$o^{(t)} = \sigma(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + b_o)$$

- cell state / cella állapot:

$$s^{(t)} = g^{(t)} \odot i^{(t)} + s^{(t-1)} \odot o^{(t)}$$

- hidden gate / rejtett kapu:

$$h^{(t)} = \tanh(s^{(t)}) \odot o^{(t)}$$

- output layer / kimeneti réteg:

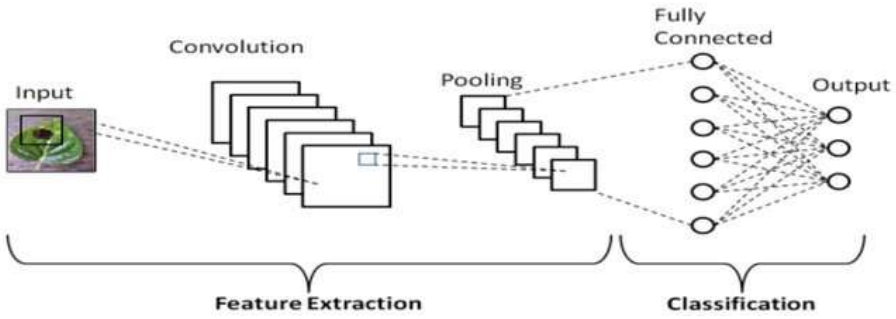
$$y^{(t)} = (W_{hy}h^{(t)} + b_y)$$

ahol a σ a szigmoid függvény, $\odot$ az elemenkénti szorzás, $x^{(t)}$ az bemeneti vektor t időegységhez, W-k a hálózati súlyok, b-k a torzítási paraméterek, y a megfigyelésekkel összehasonlítandó kimenet, h a rejtett állapot, az s-t pedig a memóriacellák cella állapotának nevezzük, ami egyedi az LSTM-re. [26]

3.4.2.2 *Konvolúciós neurális hálózatok - Convolutional neural networks (CNN)*

A konvolúciós neurális hálózatok vagy CNN-ek olyan speciális neurális hálózatok, amelyek többdimenziós topológiájú adatok feldolgozására szolgának. Ilyen adatoknak tekinthetjük az idősoros adatokat is, amelyek rendszeres időközönkénti mintavételek adataiból állnak össze és 1 dimenziós rácsnak tekinthetők. [27]

A CNN-ek háromféle rétegből állnak. Ezek a konvolúciós rétegek (convolutional layer), összevonó rétegek (pooling layer) és teljesen összekötött rétegek (fully-connected layer). Egy CNN architektúra ezeknek a rétegeknek az egymásra helyezésével jön létre. [28]



3.13. ábra - CNN általános felépítése [29]

A konvolúciós rétegekben két fontos technikát használnak: a helyi összeköttetést (local connectivity) és a súlymegosztást (weight sharing).
A helyi kapcsolódás azt jelenti, hogy minden konvolúciós csomópont a bemenetek egy kis részhalmazához csatlakozik (amely a bemenet egy helyi régiójának felel meg, és amelyet a csomópont befogadó mezőjének neveznek) az összes bemenet helyett, mint a szabványos előrecsatolt multi-layer perceptron neural network-ökben.
Továbbá a standard neurális hálózatokban alkalmazott súlyozott összegeket (weighted sums) konvolúciókkal helyettesítik. Pontosabban minden konvolúciós réteg egy előre meghatározott méretű súlymátrixszal konvolálja a bemenetet (ezt szűrőnek vagy csúszóablaknak nevezik), hogy így kiszámítsa az aktivációs térképet. Ez úgy történik, hogy súlymátrixot a bemenetre csúsztatják, és kiszámítják a bemeneti és súlymátrix közötti skalárszorzatot. Az ugyanabban a rétegben lévő konvolúciós neuronok azonos súllyal rendelkeznek, azaz ugyanazt a mintát észlelik, de a bemenet különböző részein. A súlymegosztás és a helyi összeköttetés használata csökkenti a megtanult és tárolt súlyok számát, ami gyorsabb tanulást eredményez. A konvolúciót és a súlymegosztást egy meghatározott kernelméretű szűrő segítségével valósítják meg, amely meghatározza a súlyokon osztozó csomópontok számát.

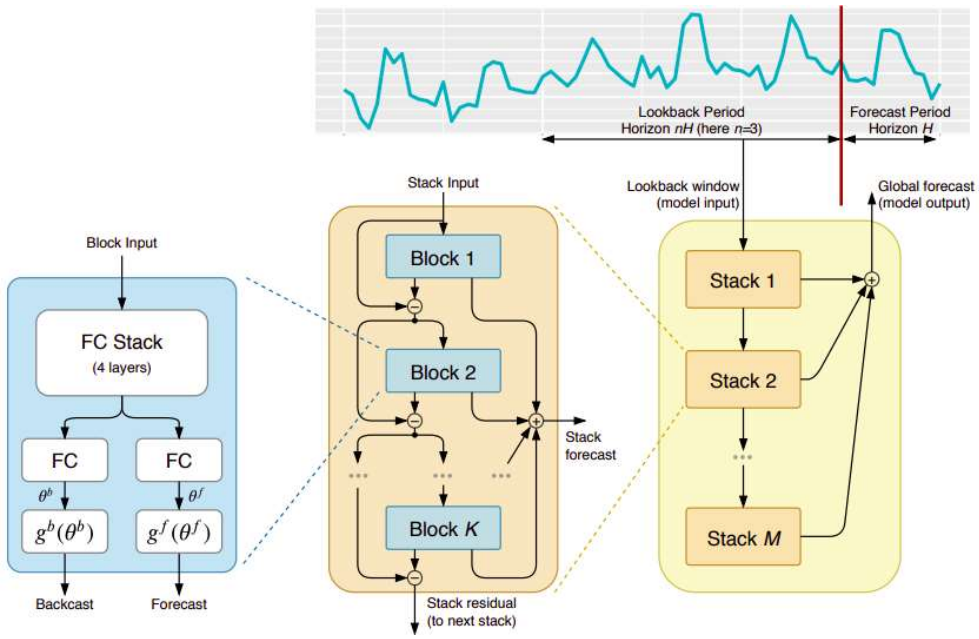
A konvolúciós réteget jellemzően egy max pooling réteg követi, amely a szomszédos neuronok kiválasztott készletének maximális értékét számítja ki a konvolúciós rétegből.

A konvolúciós és a max-pooling réteg kombinációja biztosítja, hogy a max-pooling réteg kimenete invariáns legyen a bemeneti adatok eltolódásaitól.
Végül egy teljesen összekötött kimeneti réteg kiszámítja a végső előrejelzést. [30] A teljesen összekapcsolt réteg bemenete a végső pooling réteg kimenete, amelyet vektorrá alakítanak, majd betáplálnak a teljesen összekapcsolt rétegbe. Miután áthaladt a teljesen összekapcsolt rétegeken, az utolsó réteg egy nem lineáris aktivációs függvényt használ (pl. softmax, ReLU), hogy megállapítsa annak valószínűségét, hogy a bemenet egy adott osztályba tartozik. [31]

3.4.2.3 *N-BEATS - Neural Basis Expansion Analysis for interpretable Time Series forecasting (Neurális bázis kiterjesztés elemzés értelmezhető idősor előrejelzéshez)*

Az N-BEATS egy mély tanulási architektúra, amely visszafelé és előre, visszamaradó linkeken (residual links), valamint egyváltozós idősoros előrejelzéshez használt, teljesen összekapcsolt rétegek mély halmán (deep stack) alapul.

Az N-Beats architektúra halmok (stack) halmából áll, ahol minden halom több alapvető blokkot tartalmaz.



3.14. ábra - N-BEATS architektúra [38]

Az alap blokknak villa szerű architektúrája van. Az minden blokk elfogadja a megfelelő x_l bemenetet, és két vektort ad ki, a $\widehat{x_l}$-et és a $\widehat{y_l}$-t.

Az alap blokk belül két részből áll. Az első rész egy teljesen összekapcsolt hálózat (fully connected network), amelyben a tágulási együtthatók (expansion coefficients) előre θ_l^f és hátra θ_l^b prediktorai kerülnek előállításra. A második rész a visszafelé irányuló g_l^b és az előremenő g_l^f bázisrétegekből áll, amelyek elfogadják a megfelelő előre θ_l^f és visszafelé θ_l^b kiterjesztési együtthatókat, belsőleg kivetítik azokat a bázisfüggvények halmazára, és előállítják a visszamutató (backcast) $\widehat{x_l}$-et és az előrejelző (forecast) kimeneteket $\widehat{y_l}$.

Az alap blokk első részének a működését az alábbi egyenletek írják le:

$$h_{l,1} = FC_{l,1}(x_l),\ h_{l,2} = FC_{l,2}(h_{l,1}),\ h_{l,3} = FC_{l,3}(h_{l,2}), h_{l,4} = FC_{l,4}(h_{l,3}).$$

$$\theta_l^b = LINEAR_l^b(h_{l,4}),\ \theta_l^f = LINEAR_l^f(h_{l,4}).$$

Itt a LINEAR réteg egy lineáris projekciós réteg, azaz $\theta_l^f = W_l^f(h_{l,4})$. Az FC réteg egy standard teljesen összekötött réteg RELU nemlinearitással, például:

$$h_{l,1} = RELU(W_{l,1}x_l + b_{l,1}).$$

A hálózat második része bázis rétegek segítségével a tágulási együtthatókat leképezi a kimenetre. Működését a következő egyenletek írják le:

$$\widehat{y_l} = \sum_{i=1}^{\dim(\theta_l^f)} \theta_{l,i}^f v_i^f, \quad \widehat{x_l} = \sum_{i=1}^{\dim(\theta_l^b)} \theta_{l,i}^{fb} v_i^{fb}.$$

Az architektúra által képzett két ág közül az egyik az egyes rétegek visszamutató előrejelzésén fut, a másik pedig az egyes rétegek előrejelzési ágán fut. A következő egyenletek írják le a működésüket:

$$x_l = x_{l-1} - \hat{x}_{l-1}, \quad \hat{y} = \sum_l \hat{y}_l.$$

A visszamaradt visszamutató ág (x_l) felfogható úgy, hogy azon a bemeneti jel szekvenciális elemzése fut. Az előző blokk eltávolítja az x_{l-1} jelnek azt a részét, amelyet jól tud közelíteni, így könnyebbé válik a későbbi blokkok előrejelzése. Minden blokk egy részleges előrejelzést ad ki, amelyet először a halom szintjén, majd a teljes hálózat szintjén összesítenek, hierarchikus bontást biztosítva. A végső előrejelzés ($\hat{y}$) az összes részleges előrejelzés összege. [32]

3.4.3 *Idősor előrejelzések hatékonyságának mérőszámai*

3.4.3.1 *Átlagos abszolút hiba - Mean Absolute Error (MAE)*

A MAE az előre jelzett és a valós értékek abszolút különbségének átlaga. Ahol y_i a várható érték és x_i a tényleges érték. Az n betű a teszt adathalmazban lévő értékek számát jelöli.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

A MAE megmutatja, hogy mekkora pontatlanságra számíthatunk az előrejelzéstől átlagosan. Minél alacsonyabb a MAE érték, annál jobb a modell; a nulla érték azt jelzi, hogy az előrejelzés hibamentes. Tehát modellek összehasonlításánál a legalacsonyabb MAE-vel rendelkező modellt tekintjük jobbnak. A hibastatisztika az előrejelzett értékek eredeti egységeiben van megadva.

Hátránya, hogy mivel azonban a MAE nem fedi fel a hiba arányos mértékét, nehéz lehet különbséget tenni a nagy és a kis hibák között. [33]

3.4.3.2 *Átlagos abszolút százalékos hiba - Mean Absolute Percentage Error (MAPE)*

Ez egy skálafüggetlen metrika, azaz alkalmas több, különböző léptékű idősoros modell eredményének összehasonlítására. [34] A MAPE az előre jelzett és a valós értékek átlagos abszolút különbségének aránya osztva a valós értékkel. A várt érték F_t, a valós érték A_t. Az n szám a teszt adathalmaz értékeinek számára vonatkozik.

$$MAPE = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_T}{A_t} \right|$$

Olyan adatokkal működik jól, amikben nincsenek nullák és extrém értékek. A MAPE extrém értéket vesz fel, ha a nevezőben lévő érték rendkívül kicsi vagy nagy. Egy modell akkor jobb egy másiknál, ha a MAPE értéke alacsonyabb. [33]

4. Hasonló megoldások

Az alábbiakban olyan kutatási eredmények áttekintése történik, amik sikeresen alkalmaztak machine learning algoritmusokat, várandósság alatti vagy szüléshez kapcsolódó diagnosztikában.

4.1 Machine learning rendszer alkalmazása a koraszülés jeleinek automatikus felismerésére neurális háló és elektromyographia alkalmazásával

Habár a szülés idejének előrejelzésére nem gyakori machine learning felhasználás, egy szignifikáns terület, ahol a szülészeti gyakorlatban felhasználnak gépi tanulást, az a koraszülöttség előrejelzése. Egy példa erre a „A Machine Learning System for Automatic Detection of Preterm Activity Using Artificial Neural Networks and Uterine Electromyography Data" című cikk a Liverpool-I John Moors Egyetemről. Ebben a tanulmányban egy 300 elemű adathalmazt alkalmaznak, melyet méh electromyográfia vizsgálatok során gyűjtöttek össze. A cél a 37 hét előtti koraszülés megbízható előrejelzése, valamint a feltérképezése azon különböző faktoroknak, amik koraszülést eredményeznek.
A főbb kihívások az adathalmaz kiegyensúlyozatlan természetéből fakadtak. Megoldásként többrétegű perceptron neurális hálózat lett használva Levenberg-Marquart algoritmussal tanítva. Ez a megközelítés a legkisebb négyzetek módszerét használja regressziós analízisre az adatok optimális közelítéséhez. Habár ezen terület kiemelkedő 94,56%-os szenzitivitást és 87,83%-os specificitást eredményezett, azonban a módszer informatikai erőforrások szempontjából erősen költséges. Mindettől függetlenül a tanulmány jó példaként szolgál arról, hogy hogyan informálhatja az orvosi döntéshozást a machine learning felhasználása szülészeti adatokon. [35]

4.2 Elektrohysterogram-mal mért méhösszehúzódások kiértékelése konvolúciós neurális hálózattal

Az „Evaluation of convolutional neural network for recognizing uterine contractions with electrohysterogram" egy kutatás a Bejing-i Műszaki Egyetemről, amely méh kontrakciós adatok elemzésével foglalkozik konvolúciós neurális hálózat felhasználásával. Ez a cikk a jelen vizsgált esettől eltérően nem kizárólag szülés alatti összehúzódásokat vizsgál, azzal a céllal, hogy megbízhatóan különbséget tegyen az effektív és az ineffektív összehúzódások között. Kis méretű adathalmazzal dolgozik, amely 45 terhes izlandi nő mérési eredményeit összegzi, amelyet 16 elektródás electrohysterogram-jával gyűjtöttek össze. Habár az alacsony mintavétel miatt a tanulmány hatásköre limitált, a konvolúciós neurális hálózat felhasználásával 87%-os szenzitivitást és 98%-os specificitást értek el. [36]

5. Technológiák Machine Learning-hez

Mivel az általam használt technológiákat támogató eszközök, mint alább is látható, mind Python-t támogató rendszerek, ezért a machine learning algoritmus fejlesztéséhez ez a nyelv kerül alkalmazásra.

TensorFlow:
A TensorFlow egy a Google által fejlesztett open-source machine learning framework. Támogat klasszifikációs, regressziós és deep learning algoritmusokat. Az elsődleges fejlesztői nyelve a Python, támogatja a JavaScript-ben, Java-ban, Go-n, C++-ban és C#-ban való fejlesztést is.
Nagy előnye, hogy rendelkezik egy a machine learning modell mobil eszközökre való telepítését lehetővé tévő modullal (TensorFlow lite), ami megoldja az on-device machine learning legnagyobb korlátozó tényezőit:

- Késleltetést
    - kiiktatja a szerverhez és vissza történő adatátvitelhez szükséges időt
- Adatvédelmet
    - a személyes adatok végig az eszközön maradnak, nem kerülnek sehova továbbításra
- Kapcsolattartást
    - nincs szükség hozzá internetkapcsolatra
- Méretet
    - csökkentett modell méret
- Energiafelhasználást
    - nem használ internetkapcsolatot

A modul lehetővé teszi a TensorFlow modelleket átkonvertálni TensorFlow Lite modellekké, és ezáltal a kész modell a mobil eszközön lévő bemeneti adat alapján képes elkészíteni a predikciókat. [37]

scikit-learn:
A Python könyvtárakra (NumPY, Matplotlib, SciPy stb) építve a Scikit-learn egy egyszerű és hatékony machine learning framework. Támogatja a felügyelt és a felügyelet nélküli gépi tanulást, beleértve az osztályozást, a regressziót és a klaszterezést. Támogatja továbbá a dimenziócsökkentést (a figyelembe veendő véletlen változók számának csökkentését), a modellválasztást (a paraméterek és modellek összehasonlítását, validálását és kiválasztását), valamint az előfeldolgozást (a jellemzők kiemelését és az adatok normalizálását). [38]
Rendelkezik külön idősor előrejelzést támogató könyvtárral. Ez a sktime nevű modul dedikált idősor algoritmusokat és scikit-learn kompatibilis eszközöket biztosít az összetett modellek felépítéséhez, hangolásához és értékeléséhez. Jelenleg támogat előrejelzést, idősor klasszifikációt és idősor regressziót is. [39]

Keras:
Egy Python alapú, magas szintű neurális hálózat API. A library futtaható TensorFlow-n, CNTK-n vagy Theano-n is. A Keras fő fókusza a modularitáson, a felhasználó barátságon és a bővíthetőségen van. Segítségével egyszerűbbé tehető és meggyorsítható a fejlesztés. 2017 óta integráltan megtalálható a TensorFlow-ban. [40]

A machine learning modell fejlesztéséhez választott keretrendszer az a TensorFlow lesz, továbbá hogy a munkát hatékonyabbá tegyem Keras-t is használni fogok.

6. Követelmények

A cél egy olyan előrejelző modell felállítása, ami képes a szülés kezdete óta gyűjtött kontrakciós adatokat felhasználva megbecsülni a gyermek születésének várható idejét. Ehhez az adatsoroknak a szülés lefolyásának időintervallumában az adott időpontbeli kontrakció hosszát kell tartalmaznia, az időponthoz rendelve. A megoldani kívánt probléma egyváltozós, adott időpontokban vett kontrakció hosszokat tartalmaz irreguláris időközönként megadva. Ez azonban átalakítható reguláris adathalmazzá azáltal, hogy rendszeres időközönként vizsgálva azt adjuk meg, hogy az adott időpillanatban van-e méh-izom összehúzódás és ha van, akkor az milyen hosszú. Mivel a szülés lefolyásának statisztikai jellemzőiből kitűnik, hogy az általunk használt idősornak van trendje, ezért az nem stacionárius, tehát a későbbiekben szükség lehet a stacionáriussá tételére.
A vizsgált módszerek összehasonlításakor a szülést tényleges idejét egy órával megelőzően készített előrejelzések kerülnek összevetésre. Ennek megfelelően a modellek bemeneteként a szülést megelőző egy óráig rendelkezésre álló idősor adatok állnak rendelkezésre, kimenetként pedig az ezekből az adatokból készített előrejelzés szolgál. A lehető legoptimálisabb predikció készítéséhez klasszikus idősor előrejelző módszerek és neurális hálózatokon alapuló gépi tanulási technikák kerülnek implementációra és összehasonlításra.

7. Használt adatok

Az előrejelző modell felállításához optimális esetben valós szülésekből vett adatsorok kerülnének felhasználásra. Azonban jelenleg ilyen adatbázis nem áll rendelkezésre, ezért az adatbázist először a szakirodalom alapján vett adatokból szükséges legenerálni. Ehhez az első szülő nők, normál lefolyású szülési jellemzőit veszem alapul.

7.1 Az adatgenerálás célkitűzései

Az adatgenerálás célja, hogy a valóságot lehető legjobban tükrözve létrehozzon egy olyan adatsort, ami egyenletes időközönként mintavételezve, az egyes időpontokhoz tárolja a modellezett szülés során fellépő méhkontrakciók hosszát. A létrehozott adatsorokat egy .csv fájlban tárolja. Ami így végül két oszloppal rendelkezik. Az egyik az idősor mintavételezési időpontjait, a másik az adott időponthoz tartozó kontrakció hosszt kell, hogy tárolja.

7.2 Adatok generálása

Mivel egy szülésnek több, egyedi jellemzőkkel bíró fázisa van, ezért a szülési fájásokat generáló algoritmust ezen fázisok szerinti külön modulokból kell összerakni.

Az egyes modulokban elvégzendő lépések azonosak:

1. Az adott szakasz hosszának előállítása.
2. Az adott szakaszra jellemző adatok alapján fájások generálása a létrehozott időintervallumban.

7.2.1 *Az adott szakasz hosszának előállítása*

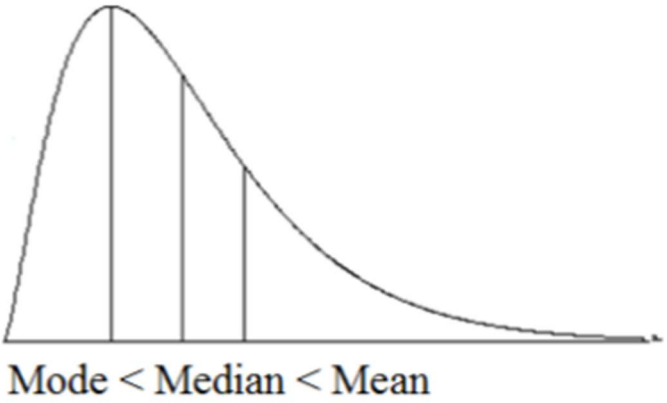
A szülési idősorok előállításához az egyes fázisok teljes hosszát kellett alapul venni, tehát először ezeket kellett létrehozni. A kapcsolódó szakirodalomban az egyes szakaszokat leíró 3 statisztikai érték:

- az átlag (jelölése: mu – μ)
    - Az átlag az a minta elemeinek az összessége, osztva az minta elemszámával.
- a medián (jelölése: Med)
    - A medián az a rendezett minta középső értéke.
- a standard deviancia/ tapasztalati szórás (jelölése: sygma-σ)
    - A standard a deviancia az a variancia négyzetgyöke.
        - A variancia kiszámításához valamennyi sokasági értékből ki kell vonnunk az átlagot, majd a különbségek négyzeteinek átlagát kell vennünk. [41]

	átlag	medián	standard deviancia
Látens fázis (I. szakasz)	11.8 óra	9 óra	7 óra
Aktív fázis (I. szakasz)	7.7 óra	7.21 óra	4.9 óra
Kitolási szakasz (II. szakasz)	54 perc	31 perc	45 perc

7.1. táblázat - Az egyes szülési szakaszokat jellemző statisztikai adatok első szülő nőknél [42] [43] [44]

Ezeknek az értéknek a segítségével felállítható a szükséges disztribúciós függvény az egyes szakaszokra jellemző hosszok eloszlására. Az értékeket vizsgálva megállapítható, hogy az eloszlás jobbra aszimmetrikus lesz, ugyanis az alábbi módon viszonyulnak hozzá a jellemző értékei:



7.1. ábra - Jobbra aszimmetrikus eloszlás [45]

A választott eloszlásnak idomulnia kell az aszimmetriához. Tehát a szimmetrikus disztribúciók, például a normál eloszlás nem jöhettek szóba. Végül a gamma eloszlást választottam, mivel annak az átlag és standard devianciával számolt értékéből visszavezetve a medián értéket, az a legközelebb volt a szakirodalomba találhatóhoz.

	valós medián			eloszlásból számolt medián		
	Látens fázis	Aktív fázis	Kitolási szakasz	Látens fázis	Aktív fázis	Kitolási szakasz
Gamma eloszlás	9 óra	7.21 óra	31 perc	10,45 óra	6,69 óra	42,14 perc

7.2. táblázat - Gamma eloszlásból számolt értékek

A gamma függvény képlete:

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dx \ \ (\alpha > 0)$$

A gamma eloszlású valószínűségi változó sűrűségfüggvényének a képlete:

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} e^{-\frac{x}{\beta}}$$

α: alakparaméter

β: skálaparaméter

A gamma eloszlás számításához a változók az átlag és a standard deviancia értéke alapján az alábbi módon számíthatók [46]:

$$\mu = \alpha \cdot \beta$$
$$\sigma^2 = \alpha \cdot \beta^2$$

Ezt a két egyenletet rendezve kiszámítható β értéke:

$$\beta = \frac{\sigma^2}{\mu}$$

Majd ezután az α értéke is:

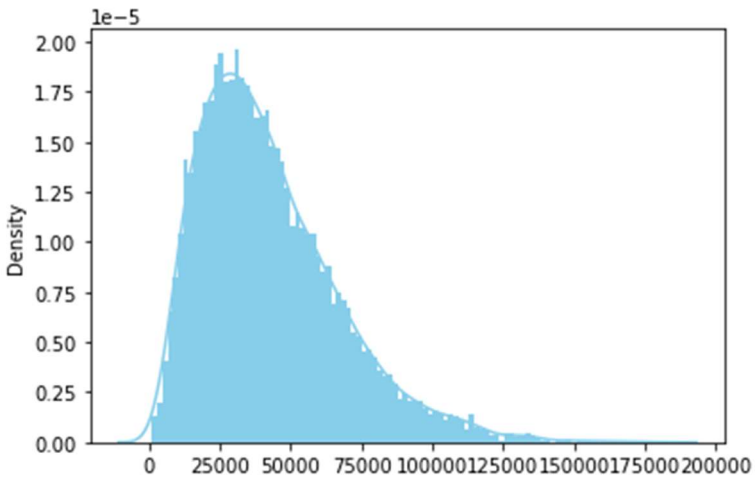
$$\alpha = \frac{\mu}{\beta}$$

A kiszámított változók segítségével ezután a Scipy stats modulját felhasználva 10000 lehetséges szakaszhosszt generáltam minden szakaszból az alábbi módon:

```
gPhaseLength = gamma.rvs(a=Alpha,scale=Beta,loc=0,size=10000)
```

7.2. ábra – Szülés hossz generálás

Ekkor a 10000 lehetséges szakaszhossz egyenletesen oszlik el a gamma eloszlás szerint:



7.3. ábra - A látens szakasz-ra generált hosszok egy lehetséges eredménye

Az egyes szakaszokhoz generált 10000 adatból azok kerültek megtartásra, amelyek a normál lefolyású szülések minimális és maximális szakaszhossz intervallumába bele estek.

7.2.2 *Az adott szakaszhoz tartozó fájások legenerálása*

Következő lépésként az egy adott hosszúságú szakaszhoz történik a szakaszba eső kontrakciók kezdeti idejének, hosszának és az egyes kontrakciók közötti intervallumok hosszának a generálása.

Szülési szakasz vagy fázis	**Kontrakció hossz (másodperc)**	**Pihenő szakasz hossza (perc)**	**Összidő elsőszülőknél**	**Összidő többgyermekeseknél**
Látens fázis (I. szakasz)	25-40	5-30	6-20 óra	8-14 óra
Aktív fázis (I. szakasz)	40-60	3-5	4-8 óra	2-4 óra
II. szakasz	60-90	1-3	40-60 perc ~max 2 óra	20-30 perc

Az egyes szülési szakaszok jellemzőit az alábbi táblázat foglalja össze:

7.3. táblázat - Méhösszehúzódások a vajúdás különböző szakaszaiban [47] [48] [49] [50]

Az adott szülési szakasz előrehaladását az alábbiakban lehet jellemezi:

- a kontrakciók hossza egyre hosszabb lesz, amíg a kezdeti értéktől el nem éri az adott szakaszt jellemző maximális értéket
- az intervallumok egyre rövidebbek lesznek, amíg a kezdeti értéktől az adott szakaszra jellemző minimális értékre nem csökkennek

Ezek alapján mind a kontrakciók hosszának és a közöttük lévő intervallumok hosszának a generálása attól függ, hogy éppen az adott szakaszban hol helyezkedünk el időben.

```
def ContractionLenghtLatent(LatentLength, timeLatent, unitLatentLenght):
    return abs(((minContractionLengthLatent + (unitLatentLenght * timeLatent)))
               + ((minContractionLengthLatent + (unitLatentLenght * timeLatent))
                  *randomnessLengthLatent(LatentLength, timeLatent)))
```

```
def ContractionIntervalLatent(LatentLength, timeLatent, unitLatentInterval):
        return abs(((maxIntervalBetweenContractionsLatent - (unitLatentInterval * timeLatent)))
                   +((minIntervalBetweenContractionsLatent + (unitLatentInterval * timeLatent))
                     *randomnessIntervalLatent(LatentLength, timeLatent)))
```

7.4. ábra - Kontrakció hossz és intervallum generálás

Amennyiben az adott szakasz progressziójában előrébb tart a generálás, úgy lesz a minimum kontrakcióhosszhoz képest egyre nagyobb és a maximum intervallumhosszhoz képest egyre kisebb a generált érték.

Mivel az egymás utáni összehúzódások változása nem egyenletes, hanem viszonylag nagy varianciát követve eltér az elvárható értékektől, ezért ezt a random tényezőt is hozzá kell adni a generált értékekhez. A random értékek mind az egyes szakaszokon belül, mind a szakaszok között időben előre haladva egyre csökkenek, követve azt, hogy a szülési fájások is egyre rendezettebbé és szabályszerűbbé válnak az idő előrehaladtával.

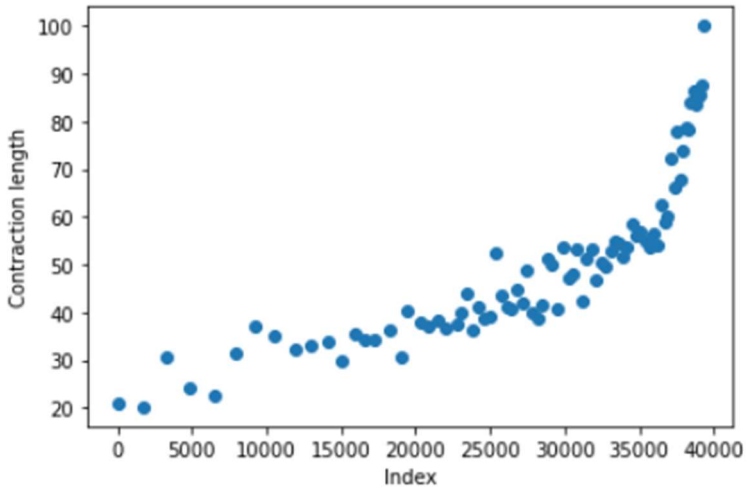
```
#egyre kisebb lesz a szórás az idő előrehaladtával -> egyre kisebb számmal szorzunk
multiplier = (LatentLengthRnd)/(LatentLengthRnd + timeLatentRnd)
```

7.5. ábra - Random értékek időben való változása

7.2.3 *Egy teljes szülés legenerálása*

Az előbbiekben az egy szülést felépítő külön komponensek kerültek generálásra. A teljes szülés a három külön létrehozott fázist összegezve jön létre. Annak érdekében, hogy a szülés lefolyása organikus mintázatot mutasson, azaz túlságosan eltérő hosszúságú fázisok ne kerüljenek össze, a generált és megtartott fázishosszok négy különböző negyedre kerültek felosztásra a hosszúságuk szerint. A teljes szülés szakaszainak kombinálásakor az adott negyedbe tartozó hosszok kerülhetnek csak párosításra. Így elkerülendő, hogy egy olyan szülés, ami lassan indul progressziónak a későbbiekben átváltson rendkívül gyorsra.

Az egyes szülések végén az utolsó, a gyermek születését jelölő kontrakció egy egységes, a generálás során el nem érhető maximumot, 100 másodperces hosszúságot kap. Ezzel egyértelművé tévé a gyermek születését az adatsoron belül.

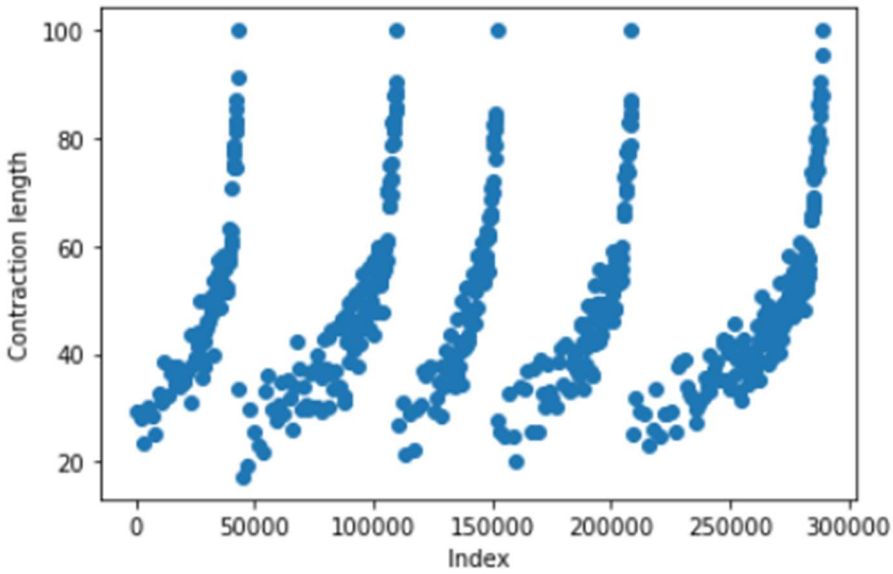


7.6. ábra - Szülés kontrakciójának jelölése

7.2.4 *Több szülés egyszerre történő legenerálása*

Abban az esetben, hogyha adott mennyiségű szülés kerül egyszerre generálásra, azok egy adatsorrá összegezve kerülnek létrehozásra. Ez úgy történik, hogy az egymás után generált szülések jellemző adatsorai össze lesznek fűzve. Ez annak érdekében történik

így, hogy a kész adatsor a szülés idejének előrejelzésekor egy folytonos idősorként lehessen felhasználni.



7.7. ábra - Több szülést tartalmazó idősor

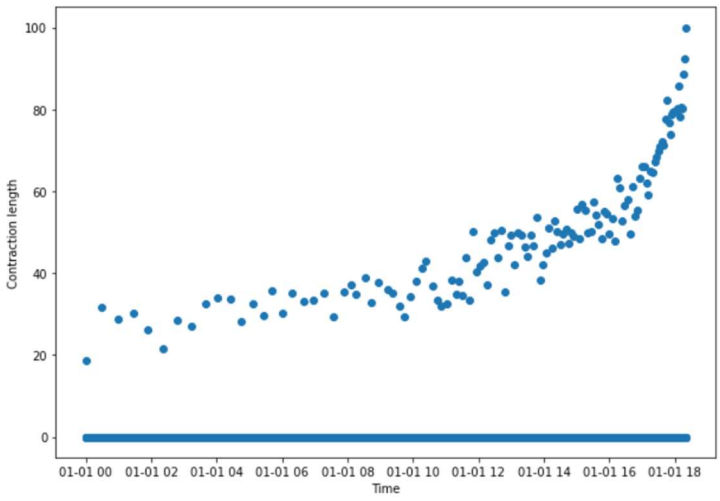
7.2.5 *Idősorok generálása*

Eddig a kontrakciós adatok kerültek generálásra, amelyek adott időpontokat és adott kontrakció hosszokat tartalmaznak.

Az jelenleg rendelkezésre álló adatokat azonban szükséges idősorrá alakítani. Annak érdekében, hogy a tanítási idők csökkenjenek, az idősor létrehozásakor 20 másodpercenkénti mintavételek kerülnek kialakításra. Az idősor idő komponense datetime típusban kerül tárolásra. Mivel a szülésig hátralévő idő előrejelzéséhez a konkrét mintavételi időpont irreleváns, ezért az idősor 2000.01.01. 00:00:00-tól indul 20 másodpercenkénti intervallumokkal, amíg el nem éri a teljes generált időintervallum huszadát az indexelésben.

A kontrakció hosszokat tartalmazó adatsort is szükséges átalakítani, hogy indexelése illeszkedjen az idő adatsorához.

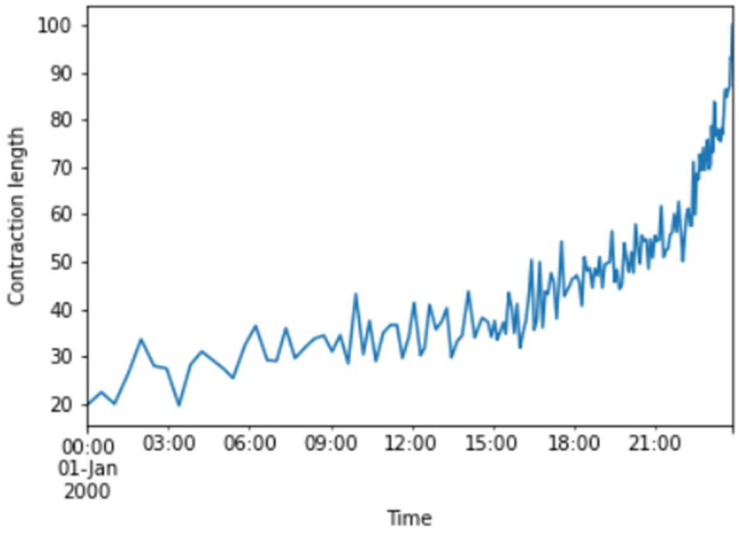
Kezdetben a kontrakciók adatsorában a kontrakció idejéhez tartozó indexnél az adott kontrakció hossza, a kontrakciókat nem tartalmazó indexeknél 0 érték kapott helyet.



7.8. ábra - Adatsor minden időponthoz kontrakciót rendelve

Azonban ebben az esetben a nulla értékek nagy aránya miatt nem lehetett tényleges előrejelzést végző modellt betanítani.

Ezt kiküszöbölendő a kontrakciók adatsorának átalakítására volt szükség úgy, hogy az egyes kontrakciók közötti időintervallumokban is nullától különböző értékek kapjanak helyet.
Ez úgy került megvalósításra, hogy az egymás után következő kontrakció hosszok különbségei el lettek osztva, a kettő közötti nullákat tartalmazó intervallum hosszával, majd az így kapott egységet felhasználva egyenletes lépésekkel került feltöltésre az adott intervallum.



7.9. ábra - Adatsor kiegyenlített kontrakciókkal

7.3 Kész adatok

Az elkészített adatsorok egy .csv fájlban kerülnek eltárolásra.
A fájl egy az idősor mintavételezési időpontjait tároló „ContractionTimes” és egy az adott indexű időpontokhozt artozó kontrakcióhosszokat tároló „ContractionLength” oszlopból áll. Az időpontok másodpercre pontos datetime típusúak és 20 másodpercenként követik egymást, a kontrakció hosszok pedig float típusúak és másodpercben tárolják az egyes időegységekhez tartozó kontrakciók hosszát.

```
export_data = zip_longest(*mergedDataDatetime, fillvalue = '')
with open('BirthTimeSeries.csv', 'w', encoding="ISO-8859-1", newline='') as birthfile:
      wr = csv.writer(myfile)
      wr.writerow(("ContractionTimes", "ContractionLength"))
      wr.writerows(export_data)
birthfile.close()
```

7.10. ábra - Generált adatsor tárolása

```
df = pd.read_csv('BirthTimeSeries.csv'
                 ,parse_dates=['ContractionTimes'],index_col=['ContractionTimes'])
```

```
df.head()
```

ContractionTimes	ContractionLength
2000-01-01 00:00:00	22.000595
2000-01-01 00:00:20	22.093590
2000-01-01 00:00:40	22.186585
2000-01-01 00:01:00	22.279581
2000-01-01 00:01:20	22.372576

7.11. ábra - Generált adatsor beolvasása

7.4 Az adatgenerálás limitációi

A való életben a szülés lefolyása egyénenként rendkívül egyedi lefolyást mutathat, azt hogy a szakirodalom alapján elvárt értékektől a való életben milyen eltérések jelennek meg a generált adatok nem képesek visszaadni, így az előrejelzés során sem lesz része az elkészített modellnek.

8. Implementáció

A szülés előrejelzése egy órával az újszülött világra jövetelének ideje előtt történik. Ez azt jelenti a gyakorlatban, hogy egy órával előre történő predikcióhoz, az előrejelzendő szülés adatsorának utolsó 180 darab 20 másodperces egysége lett felhasználva teszt adatsorként. A cél, hogy erre készítsünk minél jobb előrejelzést úgy, hogy az előrejelzésben szerepeljen az szülést jelölő 100 másodperces fájás és a tesztadatsor ugyanezen fájásához időben a lehető legközelebb helyezkedjen el. A vizsgált módszerekkel felállított modellek ugyanazon az 50 darab adatsoron kerülnek tesztelésre, amelyek mindegyike egy szülést reprezentál. Azért, hogy a sikertelen, illetve az irreálisan nagy hibát generáló teszt adatsorok ne gátolják meg az egyes modellek kiértékelését, abban az esetben, ha az előrejelzés időpontjától vett maximum 7 órán belül, azaz 1260 idősorbeli mintavételi időpont figyelembevételével, ami maximum 6 órás késést jelent, nem jön létre sikeres predikció, akkor az adott előrejelzés sikertelennek lett minősítve.

Az egyes modellek a következő három fő szempont szerint kerülnek összehasonlításra:

1. A tesztadatsoron produkált sikeres előrejelzések prediktált szülési időpontja és az adatsor tényleges szülési ideje közötti percbeli eltérés.

$$Percbeli\ időkülönbség = \frac{|Becsült\ idősor\ hossza - Valós\ idősor\ hossza| * 20}{60}$$

2. A tesztadatsor hossza és a prediktált adatsor hossza közötti százalékos eltérés sikeres előrejelzés esetén.

$$Százalékos\ hiba = \frac{|Becsült\ idősor\ hossza - Valós\ idősor\ hossza|}{Valós\ idősor\ hossza} * 100$$

3. A tesztadatsorokon való előrejelzések futtatása során keletkező sikertelen előrejelzések darabszáma.

A létrehozott modellek egymáshoz viszonyított hatékonysága ezeket az eredményeket figyelembe véve kerül a továbbiakban meghatározásra.

A fejlesztéshez használt nyelv a Python, annak is a 3.8.8-as verziója. A modellek implementációja Python könyvtárak segítségével történik. Többek között a Pandas, a Numpy és a Matplotlib kerül alkalmazásra az adatok kezeléséhez és vizualizációjához. A klasszikus gépi tanulási módszerek esetében a Statsmodels könyvtár, míg a neurális hálózatok esetében a Tensorflow és a Keras segítik az implementációt. A modellek futtatásához szükséges környezet teljes specifikációja a pred_env.yml konfigurációs fájlban található.

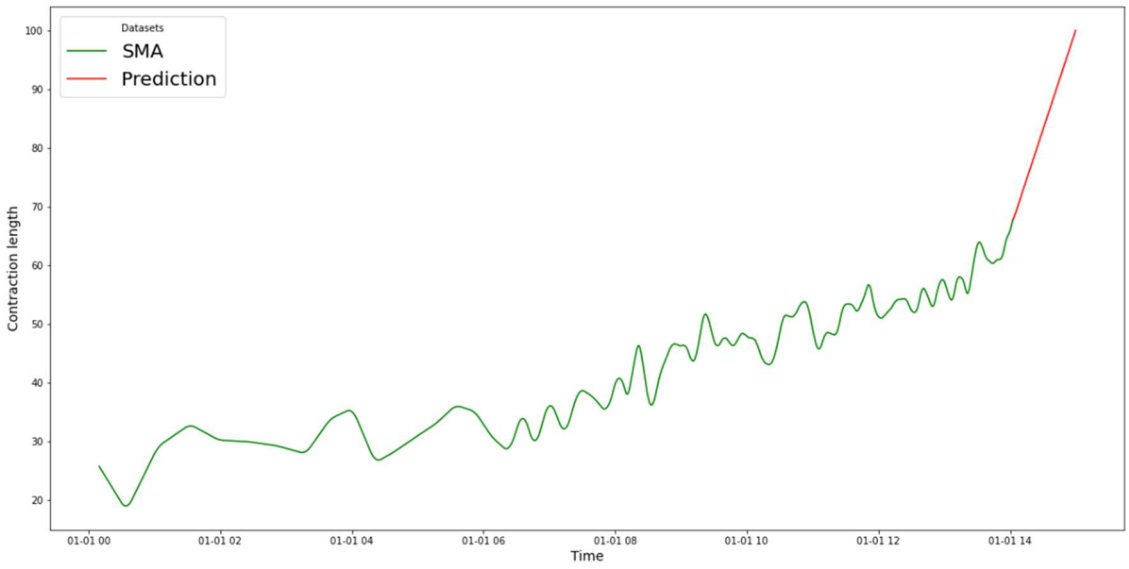
8.1 Alapmodell (Baseline model)

Ahhoz, hogy a létrehozott modellek hatékonyságát elemezni tudjuk, fontos felállítani egy referencia modellt. Ennek a modellnek a hatékonysága lesz az alapja annak, hogy a későbbi modellek hatékonyságát meg tudjuk határozni.

Alapmodell elkészítéséhez az egyszerű mozgóátlag (Simple Moving Average - SMA) módszert választottam. Ez az előrejelzés a számtani közepe a legutóbbi n darab mintából álló idősor ablaknak. Minden előrejelzési ciklusban a legrégebbi minta eltávolításra kerül és egy újabb pedig hozzáadásra. [51] Mivel az SMA-val készített előrejelzéseknek sajátossága, hogy egy idő után beállnak egy állandó értékre, ezért a módszert módosítani kell.
A prediktáláshoz használt adatsorra az eredeti SMA módszer lett alkalmazva. Azonban az előrejelzések elkészítéséhez az idősor ablak értékeiből számolt átlag helyett, a következő érték az idősor ablakbeli értékek deltáiból kapott átlagnak és az utolsó idősor ablakbeli értéknek az összegéből lett számolva.

Egy órával a szülés tényleges ideje előtt készült az előrejelzés és az előrejelzést sikertelennek lett minősítve abban az esetben, ha az előrejelzés 6 óránál többet késett.

Az elsimított tanító adatok és az elkészített előrejelzések mintája az alábbi ábrán látható:



8.1. ábra - SMA-val készített adatsor és előrejelzés

Az elkészített modellt 50 teszt adatsoron futtatva az alábbi eredmények születtek az egy órával előre készített előrejelzések esetében:

	SMA – idősor ablak mérete					
Előrejelző ablak mérete			**1**	**5**	**15**	**30**
	1	Átlagos időbeli eltérés:	**29.4 perc**	35.77 perc	55.4 perc	55 perc
		Átlagos százalékos eltérés:	**49.06%**	59.62%	92.37%	91.6%
		Legjobb eredmény:	**1.33 perc, 2.22%**	0.66 perc, 1.11%	9 perc, 15%	1.66 perc, 2.78%
		Legrosszabb eredmény:	**53.7 perc, 9.4 %**	136.3 perc, 227.2%	348.7 perc, 571.1%	254 perc, 423.3%
		Sikertelen előrejelzések száma:	**20**	21	16	15
	5	Átlagos időbeli eltérés:	30.2 perc	45.78 perc	37.8 perc	62.3 perc
		Átlagos százalékos eltérés:	50.35%	77.98%	63.04%	103.87%
		Legjobb eredmény:	1.33 perc, 2.22%	0.66 perc, 1.11%	3.64 perc, 6.11%	3.67 perc, 6.11%
		Legrosszabb eredmény:	102.3 perc, 170.56%	297 perc. 496%	133.3 perc, 22.22%	303.3 perc, 505.6%
		Sikertelen előrejelzések száma:	20	19	19	14
	15	Átlagos időbeli eltérés:	47.66 perc	50.9 perc	50.1 perc	61.9 perc
		Átlagos százalékos eltérés:	79.43%	84.87%	83.44%	103.1%
		Legjobb eredmény:	0 perc, 0%	0.33 perc, 0.55%	0.33 perc, 0.55%	3.67 perc, 6.11%
		Legrosszabb eredmény:	266.7 perc, 444,4%	269.7 perc, 449.4%	273.66 perc, 456.11%	232 perc, 386.67%
		Sikertelen előrejelzések száma:	16	16	19	15
	30	Átlagos időbeli eltérés:	44.14 perc	46.81 perc	56.7 perc	79.4 perc
		Átlagos százalékos eltérés:	73.56%	78.0%	94.51%	132.3%
		Legjobb eredmény:	3.33 perc. 5.56%	3 perc, 5%	2 perc, 3.33%	1.7 perc, 2.78%
		Legrosszabb eredmény:	199 perc, 331.67%	228 perc, 380%	217 perc, 361.67%	348 perc, 580.6%
		Sikertelen előrejelzések száma:	16	16	16	17

8.1. táblázat - Alapmodell eredmények (egy órás előrejelzés)

A legpontosabb predikciót a modell abban az esetben produkálta, amikor az adatsoron az SMA módszer és az előrejelzés is 1 minta hosszúságú idősor ablakkal került alkalmazásra. Ebben az esetben az átlagos eltérés 29.4 perc és 49.06% volt, valamint a modell 50 teszt adatsorból 20 esetében nem volt képes 6 előrejelzést készíteni. Lényegében ez azt jelenti, hogy az alapmodell leegyszerűsíthető arra, hogy az előrejelzés

úgy jön létre, hogy a következő időpont kontrakcióhossza az utolsó és utolsó előtti kontrakcióhosszok különbségének és az utolsó kontrakcióhossz összege. Azaz:

$$x_{t+1} = x_t + (x_t - x_{t1})$$

A továbbiakban egy olyan modell felállítása a cél, ami képes ennél pontosabb eredményt produkálni, illetve nagyobb arányban sikeres előrejelzéseket készíteni.

8.2 Holt féle exponenciális simítás

Az exponenciális simítás az egy egyváltozós idősoros adatokhoz használt előrejelző módszer. Olyan idősoroknál alkalmazható, ahol van trend, de nincsen szezonalitás, tehát megfelel az általam vizsgált idősorok jellemzőinek. Az ezzel előállított előrejelzések a múltbeli megfigyelések súlyozott átlagai, ahol a régebbi megfigyelések súlya exponenciálisan csökken.

A módszer paramétereinek módosításával megválasztható, hogy a régebbi megfigyelések milyen gyorsan veszítenek a számításbeli súlyukból.

A holt féle exponenciális simításnak, más néven a kétszeres exponenciális simításnak, két komponense van. Ezek a szint és a trend.

A szint komponens súly paramétere az alpha, ez határozza mega a szint simításnak a mértékét azáltal, hogy a szint komponens milyen gyorsan igazodik a legfrissebb adatokhoz. Az alfa értékek 0 és 1 között lehetnek. Az alacsonyabb értékek simábban illesztett vonalakat eredményeznek, mivel nagyobb súlyt adnak a múltbeli megfigyeléseknek, ezáltal átlagolják az időbeli ingadozásokat. A magasabb értékek szabálytalanabb vonalat hoznak létre, mert nagyobb súlyt adnak az aktuális adatoknak, ami csökkenti a régebbi adatok átlagoló hatását. Ugyan a változó adatokra való gyors reagálás elsőre pozitív jellemzőnek tűnik, amennyiben túl magas súlyérték kerül beállításra az szabálytalan előrejelzéseket eredményezhet, mivel a modell reagál a véletlenszerű ingadozásokra, más néven zajokra. Ha a súly értéket 1-re állítjuk az azt eredményezi, hogy a legutóbbi megfigyelésé lesz az összes súly és a korábbi megfigyeléseknek nem lesz hatása a modellre, azaz az előrejelzés értéke egyszerűen az aktuális érték lesz.

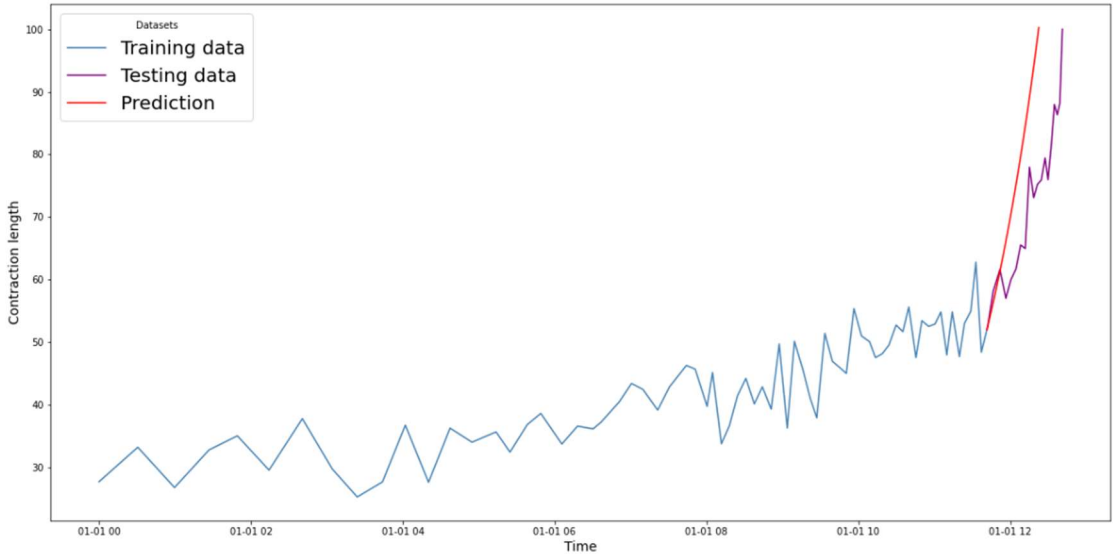
A trend komponens paramétre a beta. Ahogy az alpha, a beta is 0 és 1 közötti értékeket vehet fel. Hasonlóan az előzőekben leírtakhoz, a magasabb értékek nagyobb súlyt helyeznek a közelmúltbeli megfigyelésekre, így a trendkomponens gyorsabban tud reagálni a trend változásaira, míg a kisebb értékek csillapító hatásúak.

Ezzel szemben a mozgóátlag módszer minden korábbi megfigyelést, amik a mozgóátlag ablakba esnek egyformán súlyoz, és az ablakon kívüli megfigyeléseket nulla súllyal veszi figyelembe. [52] [53]

A Holt féle exponenciális simítást a „Statsmodels” nevű Python könyvtár segítségével alkalmaztam. Ez a csomag osztályokat és függvényeket biztosít számos különböző statisztikai modell becsléséhez, valamint statisztikai tesztek elvégzéséhez és statisztikai adatok vizsgálatához. [54] Az általam használt „ExponentialSmoothing” osztályt az idősor analízisre használt „tsa” modul tartalmazza.

A módszert multiplikatív trenddel alkalmaztam az adatsorokon, mivel a trend exponenciális változást mutat.

- A tanító és teszt adatok, valamint az elkészített előrejelzések mintája az alábbi ábrán látható: Legjobb eredmény:
    - átlagos eltérés: 29.4 perc / 49.06%
    - Sikertelen előrejelzések száma: 20
    - SMA ablak mérete: 1
    - Előrejelző alak mérete: 1



8.2. ábra - Holt féle exponenciális simítással készített előrejelzés

A komponensek súly paramétereinek optimális értékét iteratív módszerrel kerestem meg. Az 50 teszt adatsoron lefuttattam a modellt a lehetséges súlykombinációkkal, az így kapott táblázatból leolvasható a legjobb eredményt produkáló súlykombináció. Az alábbi táblázatban modell segítségével egy órával előre készített előrejelzések átlagos százalékos, valamint percbeli eltérései és a sikertelen predikciók darabszáma látható az adott alpha és beta értékek esetén:

		Beta								
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Alpha	**0.1**	55.50% 33.3 p 19	52.76% 32.9 p 22	59.79% 35.9 p 21	49.84% 49.8 p 21	83.75% 50.2 p 19	90.82% 54.5 p 19	87.44% 52.5 p 20	76.74% 46.0 p 19	67.90% 40.7 p 18
	0.2	59.08% 35.4 18	80.2% 48.1 p 17	64.69% 38.8 p 20	74.44% 44.7 p 19	86.84% 52.1 p 18	59.70% 35.8 p 20	58.59% 35.2 p 20	61.67% 37.0 p 20	58.02% 34.8 p 22
	0.3	69.74% 41.8 p 16	71.26% 42.8 p 17	67.50% 40.5 p 20	84.91% 50.9 p 20	61.56% 36.9 p 19	46.11% 27.7 p 21	45.24% 27.1 p 20	50.48% 30.3 p 21	59.62% 35.8 p 21
	0.4	64.46% 38.7 p 16	57.22% 34.3 p 20	71.23% 42.9 p 22	65.20% 39.1 p 20	47.85% 28.7 p 20	44.98% 27.0 p 20	46.54% 27.9 p 20	66.59% 40.0 p 19	64.52% 38.7 p 20
	0.5	64.90% 38.9 p 16	49.56% 29.7 p 20	84.37% 50.6 p 19	49.41% 29.6 p 20	47.17% 28.3 p 20	46.02% 27.6 p 20	55.93% 33.6 p 20	50.44% 30.3 p 20	64.21% 38.5 p 21
	0.6	49.14% 29.5 p 17	51.42% 30.9 p 21	80.89% 48.5 p 20	48.52% 29.1p 20	46.17% 27.7 p 20	48.65% 29.2 p 20	47.91% 28.7 p 20	53.96% 32.4 p 20	65.67% 39.4 p 20
	0.7	63.42% 38.0 p 16	60.83% 36.5 p 20	63.61% 38.2 p 20	47.01% 28.2 p 21	46.83% 28.1p 20	54.39% 32.6 p 19	48.25% 28.9 p 20	47.83% 28.7 p 21	73.78% 44.3 p 19
	0.8	60.54% 36.3 p 16	80.78% 48.5 p 18	61.59% 37.0 p 20	63.06% 37.8 p 19	47.85% 28.7 p 20	45.69% 27.4 p 20	48.83% 29.3 p 21	50.91% 30.5 p 20	69.17% 41.5% 20
	0.9	60.35% 36.2 p 17	75.80% 45.5 p 18	60.96% 36.6 p 20	62.08% 37.4 p 19	78.78% 47.3 p 19	**44.83%** **26.9 p** **20**	49.74% 29.7 p 21	57.62% 34.6 p 22	51.89% 31.1 p 22

8.2. táblázat - Holt féle exponenciális simítással készített előrejelzések eredménye az adott alpha és béta értékek esetén

A kapott eredmények alapján az optimális alpha értéke 0.9 és beta értéke 0.6, ekkor az átlag eltérés 26.9 perc és 44.83%. Ez 2.5 perccel és 4.23 százalékkal jobb eredmény, mint az alapmodell esetében. Látható, hogy a holt féle exponenciális simítás módszerével sikerült növelni a predikciók pontosságát, azonban a sikertelen előrejelzések aránya változatlan maradt.

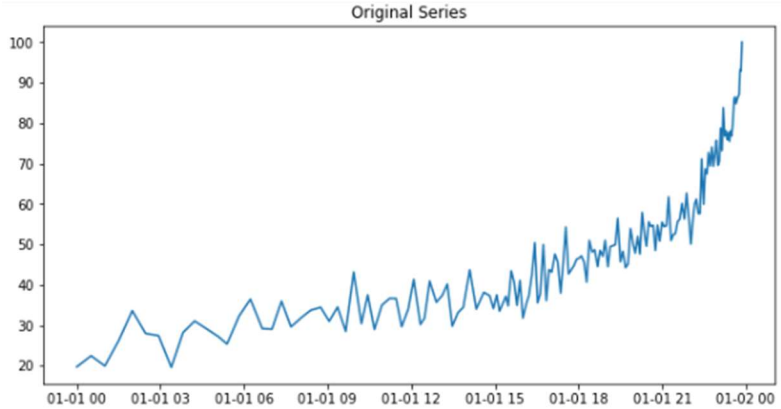
8.3 ARIMA

Az Autoregresszív integrált mozgóátlag (Autoregressive Integrated Moving Average) modell, azaz ARIMA modell, ahogy a rövidítés is mutatja három különböző komponensből áll. Az autoregresszió (AR) egy regressziós modell, ami egy adott megfigyelés korábbi megfigyelésektől való függőségét használja, paramétere a modellezéshez használt múltbeli tagok száma, jelölése „p”. A mozgóátlag (MA) modell, egy olyan regressziós folyamatot jellemez, ahol a hibatag az a jelenlegi és a múltbeli hibatagok lineáris kombinációja, a modellhez tartozó paraméter a „q”, ami a mozgóátlaghoz tartozó megfigyelések száma. Az integrált (I) tag arra utal, hogy a nem stacionárius folyamat stacionáriussá tehető a megfigyelések differenciálásával, ennek a tagnak a paramétere „d”-vel jelölt, értéke pedig megegyezik a differenciálások számával. [55]

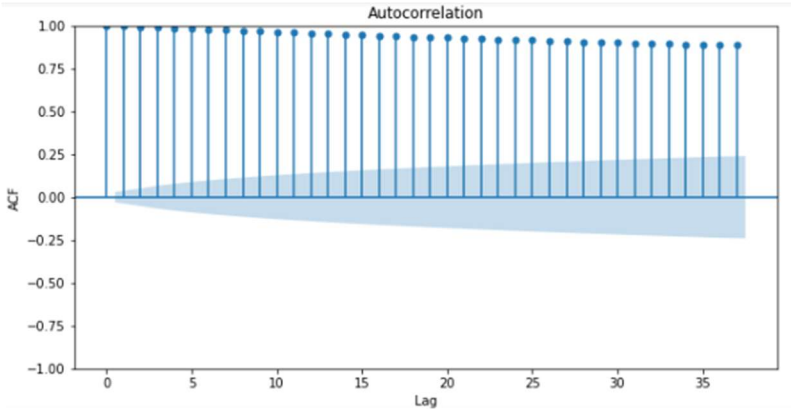
Az első lépés az ARIMA modell felállításához annak vizsgálata, hogy az idősor stacionárius-e. Az idősort csak abban az esetben szükséges differenciálni, ha az nem stacionárius. A stacionárius idősorok olyan idősorok, amelyek tulajdonságai nem függenek a megfigyelés időpontjától, azaz a középértéke és varianciája állandó, valamint a kovariancia független az időtől. A stacionaritás vizsgálata a kiterjesztett Dickey-Fuller teszttel (ADF Test) történt, amihez a használt „adfuller” metódust a Statsmodels könyvtár „tsa” modulja tartalmazza. Az ADF teszt nullhipotézise az, hogy az idősor nem stacionárius. Tehát, ha a teszt p-értéke kisebb, mint a szignifikancia szint (0.05), akkor elvetjük a nullhipotézist, és arra következtetünk, hogy az idősor valóban stacionárius, viszont, ha P érték > 0.05, akkor tovább lépünk annak megállapításával, hogy az idősort hányszor szükséges differenciáli a stacionáriussá tételhez. [56] A teszt eredménye p = 0.998755 lett és mivel ez meghaladja a 0.05-ös szignifikancia szintet, ezért az adatsor differenciálásra került.

Ezután meg kell határozni, hogy mi az integrált paraméter értéke, ami az idősort stacionáriussá teszi, azaz, hogy hányszor kell az adatsort differenciálni. Ezt az adatsor ábrázolásából és az autokorrelációs diagramból olvashatjuk le. Ha van trend az adatokban, a kis „Lag”-ek autokorrelációi általában nagyok és pozitívak, és az idő múlásával lassan csökkennek, ahogy a „Lag”-ek nőnek.

Az eredeti idősor ábrája és autokorrelációs diagramja:



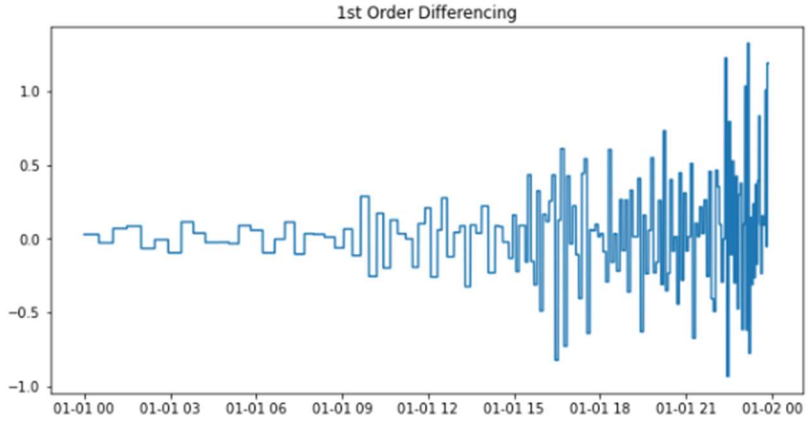
8.3. ábra - Az eredeti idősor ábrázolása



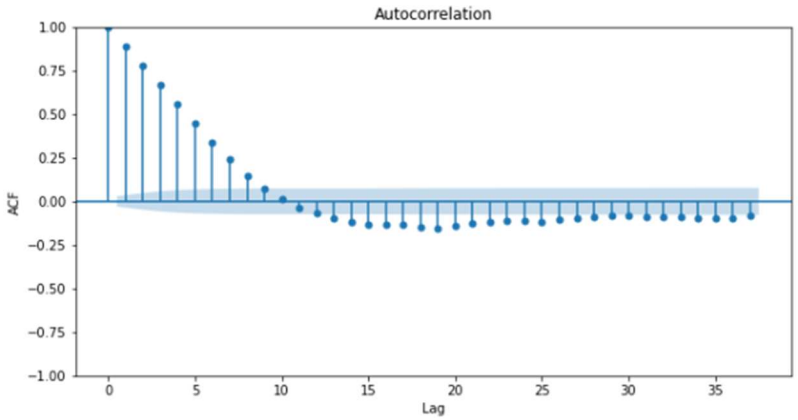
8.4. ábra - Az eredeti idősor autokorrelációs diagramja

Mindkét ábrából leolvasható, hogy az adatsornak erős trendje van, tehát nem lehet stacionárous.

Az egyszeresen differenciált idősor ábrája és autokorrelációs diagramja:



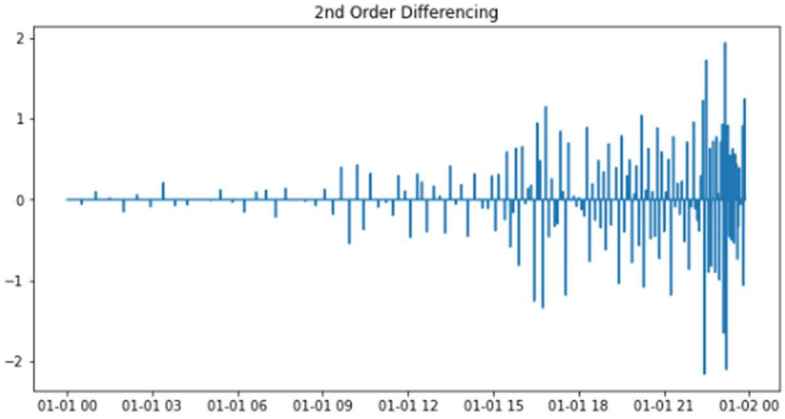
8.5. ábra - Az egyszeresen differenciált idősor ábrázolása



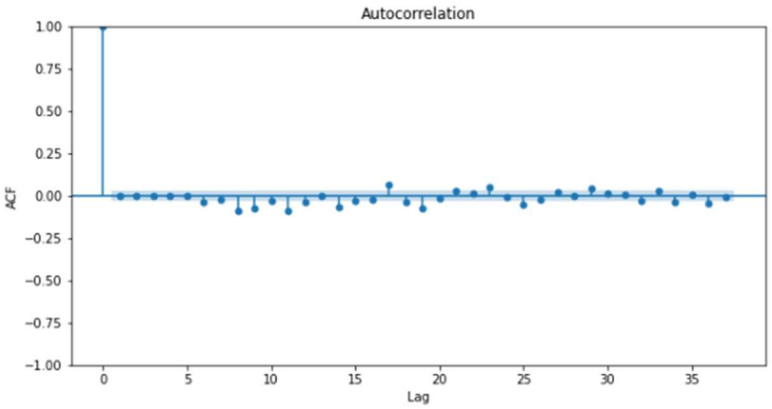
8.6. ábra - Az egyszeresen differenciált idősor autokorrelációs diagramja

Az ábrákon látható, hogy az egyszer differenciált adatsor trendje ugyan csökkent az eredetihez képest, de még mindig jelen van. Tehát szükség van még további differenciálásra a stacionáriussá tételhez.

A kétszeresen differenciált idősor ábrája és autokorrelációs diagramja:



8.7. ábra - A kétszeresen differenciált idősor ábrázolása



8.8. ábra - A kétszeresen differenciált idősor autokorrelációs diagramja

A második differenciálás után látható, hogy a trendet sikerült megszüntetni, tehát az idősor stacionárius lett. Az integrált paraméter értéke tehát d=2, hiszen a kétszeresen differenciált idősor vált stacionáriussá.

Ahhoz, hogy a modell a lehető legoptimálisabb p és q paraméter értékeket kapassa, iteratív módszerrel, adott értékpárokkal készítve az ARIMA előrejelzést, lett megállapítva a legjobb modellt produkáló értékpár. A paraméter értékek keresésekor kapott modell eredmények az átlagos százalékos és időbeli eltéréssel, valamint a sikertelen előrejelzések számával, az alábbi táblázatban láthatóak:

		q					
		0	1	2	3	4	5
p	0	77.0% 46.2 p 20	77.0% 46.2 p 20	77.0% 46.2 p 20	77.0% 46.2 p 20	77.0% 46.2 p 20	77.0% 46.2 p 20
	1	77.0% 46.2 p 20	77.0% 46.2 p 20	76.96% 46.2 p 20	77.98% 46.19 p 20	77.02% 46.21p 20	104.78% 62.87 p 20
	2	77.0% 46.2 p 20	77.0% 46.2 p 20	235.13% 146.1 p 19	149.0% 89.42 p 17	142.8% 86.69 p 20	90.39% 54.23 p 20
	3	77.0% 46.2 p 20	77.0% 46.2 p 20	187.24% 112.3 p 18	221.2% 132.73 p 17	345.85% 207.5 p 18	143.94% 86.37 p 20
	4	77.0% 46.2 p 20	77.0% 46.2 p 20	145.05% 87.0 p 19	167.12% 100.87 p 19	251.27% 150.8 p 18	124.6% 76.8 p 19
	5	77.0% 46.2 p 20	77.0% 46.2 p 20	97.91% 58.74 p 20	150.82% 90.5 p 19	184.43% 110.7 p 19	94.89% 56.9 p 20

8.3. táblázat - ARIMA-val készített előrejelzések eredményei

Az eredmények alapján látható, hogy a legjobb eredményeket az ARIMA modell akkor volt képes elérni, amikor a „p” vagy „q” értékek kicsire (0-ra vagy 1-re) voltak állítva, azaz amikor csak a közvetlen múltbeli értékek alapján végezte az előrejelzést. Ezekben az esetekben a kapott eredmények szinte mind azonosak is lettek.

8.4 LSTM

A Long Short-Term Memory (LSTM) egyfajta Recurrent Neural Network (RNN), amely képes megjegyezni a korábbi állapotok értékeit a későbbi felhasználás céljából.

Az LSTM modellek implementációja a Tensorflow és Keras könyvtárak segítségével történt. A kiválasztott modell létrehozása során több architektúra is kipróbálásra került, hogy a módszer lehető legoptimálisabb, azaz legjobb teljesítmény nyújtó modellje kerüljön kiválasztásra.

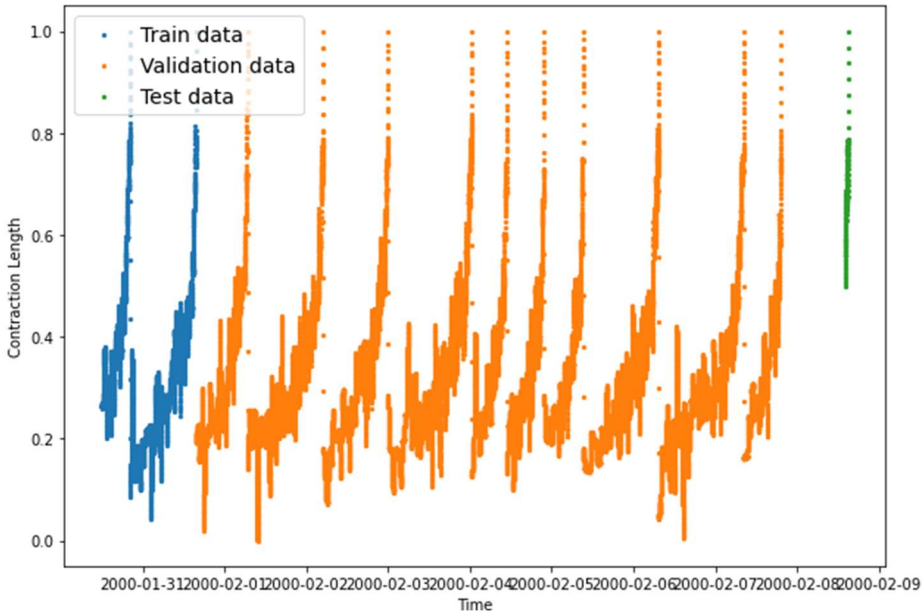
A tanítást egy a következő féleképpen felépített adatsoron történt: Tanító adatsorként egy 50 szülést folyamatos idősorként összefűző adatsort használ, amit ugyanehhez az idősorhoz illesztve egy 20 validációs szülési adatsor követ.

A tanítás befejeztével 50 db teszt adatsoron kerül tesztelésre az elkészített modell, amelyek mindegyike egy darab szülés idősorát tartalmazza.

A használt adatsorok adatai normalizált formában kerültek alkalmazásra, az ehhez használt normalizációs módszer a min-max skálázás volt. A min-max skálázás során az

adatsor adatai transzformálásra kerülnek úgy, hogy az értékeik a 0 és az 1 közötti tartományba essenek, ahol a 0 az adatsor legkisebb értékének, az 1 pedig az adatsor legnagyobb értékének felel meg. A skálázáshoz használt függvény egyenlete a következő: $x_{scaled} = \frac{x - \mathrm{mi}\ \ (x)}{\max(x) - \min\ (x)}$. A normalizálás implementálásához a Scikit-learn preprocessing osztályának a MinMaxScaler metódusa került alkalmazásra.

A használt adatsorok végső formája az alábbi ábrán látható.



8.9. ábra - Adatsor ábrázolása

Ahhoz, hogy az idősor használható legyen a felügyelt tanulásra, olyan formára kell átalakítani, ahol bemeneti adatokhoz várható kimenet van társítva. Ehhez a bemeneti adatokat a múltbeli értékek, a várható értéket pedig a használt múltbeli értékek után következő érték szolgáltatja. Az így létrehozott bemeneteket bemeneti ablakoknak hívjuk, a hozzájuk tartozó várható értékeket pedig horizontnak. A tanítás során használt bemeneti ablak 180 adat hosszúságú, ami a szülésben 1 órát jelent. Ezen adatokkal mindig a közvetlenül az ablak adatai után következő adat kerül előrejelzésre, tehát a horizont hosszúsága egy lesz. Ezután ezzel az új adattal kerül előállításra a következő előrejelzéshez használt adat ablak.

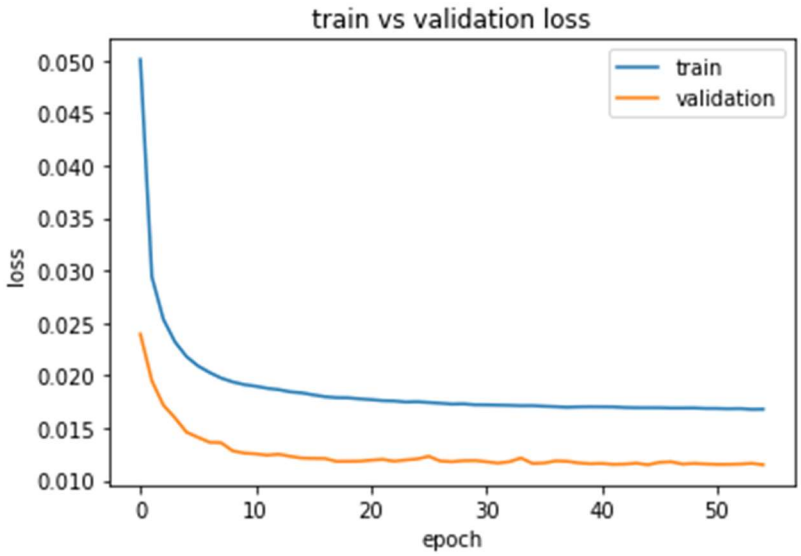
Az elkészített LSTM architektúrák között „vanilla” LSTM hálózatok szerepelnek, amelyek csak egy LSTM réteggel rendelkeznek, valamint „stacked” LSTM hálózatok, amelyek jelen esetben két LSTM réteget tartalmaznak.

Az LSTM hálózat maximális epoch számának 200 került beállításra, a batch size 128 volt.

A neurális hálózatok hajlamosak a túltanulásra, amikor is a hálózat túlzottan ráilleszkedik a tanítási adatsorra. Ez a modell teljesítményének csökkenéséhez vezet, amikor addig ismeretlen adatokon kerül kiértékelésre. Azért, hogy ennek a valószínűsége csökkenjen, különböző technikák kerülnek alkalmazásra. A dropout egy olyan regularizációs

módszer, ami tanítás alatt ignorál néhány neuront, tehát az aktív neuronok véletlenszerűen kerülnek kiválasztásra. [57] Az Early Stopping megállítja a tanulást, amennyiben a modell teljesítménye nem javul tovább. A teljesítményt a validációs adathalmaz loss értékével vizsgálja. Amennyiben az előre meghatározott iterációszámot túllépi a modell úgy, hogy ha a validációs loss nem csökken, akkor megszakítja a tanítást.

Az előbbiek alapján 0.1-es dropout és 0.2-es recurrent dropout, valamint Early Stopping lett beállítva 10 patience-el. A használt optimalizáció az Adam, aminek 0.0001-es learning rate lett beállítva. A kiválasztott aktivációs függvény a ReLu. A tanítás során a loss értékek mean absolute error-al (MAE) lettek számolva. A legjobb modell a validation loss figyelésével lett elmentve a tanítás során a Keras ModelCheckpoint funkcióját használva. A tesztelés ezen a mentett modellen valósult meg.



8.10. ábra - Az LSTM_8 loss diagramja

A végső előrejelzés több lépéssel a jövőbe igényel predikciót. Ezt a modell az egylépéses predikciók iteratív összefűzésével állítja elő. Ez a gyakorlatban azt jelenti, hogy az utolsó ismert bemeneti ablak értékei alapján a modell elkészíti az előrejelzést a következő adat értékére, majd ezt az ablakhoz fűzve és az ablak legrégebbi értékét elhagyva, megkapja a következő lépés előrejelzéséhez szükséges új ablakot. Ezeket a lépéseket addig folytatja, amíg a legutolsó előrejelzett érték el nem éri a szülést jelző fájás hosszának értékét, ami ebben a normalizált formában 1.0-nek felel meg.

A szülés tényleges ideje és az előrejelzett időpont különbsége végül megkapható az adott szüléshez tartozó eredeti adatsor és az előrejelzett adatsor hosszának különbségéből. Mivel egy lépés az idősorban 20 másodpercek felel meg, a különbséget 20-szal megszorozva megkapjuk az előrejelzés másodpercbeli pontatlanságát.

A modell végső kiértékelése a valós szülési idő és a becsült szülési idő közötti percbeli időkülönbséget, valamint százalékos hibáját használva történik.

A kipróbált implementációk eredményei, az egyes modellek LSTM rétegeinek unit számával feltüntetve:

Modell	Units	Átlagos időbeli eltérés (perc)	Átlagos százalékos eltérés	Sikertelen előrejelzések száma
LSTM_4	4	57.27	95.46%	0
LSTM_8	**8**	**46.66**	**77.77%**	**0**
LSTM_16	16	69.61	116.02%	0
LSTM_32	32	96.05	160.08%	0
LSTM_8_2	8+2	Overfitted		50
LSTM_8_4	8+4	Overfitted		50
LSTM_12_2	12+2	Overfitted		50

8.4. táblázat - LSTM eredmények I.

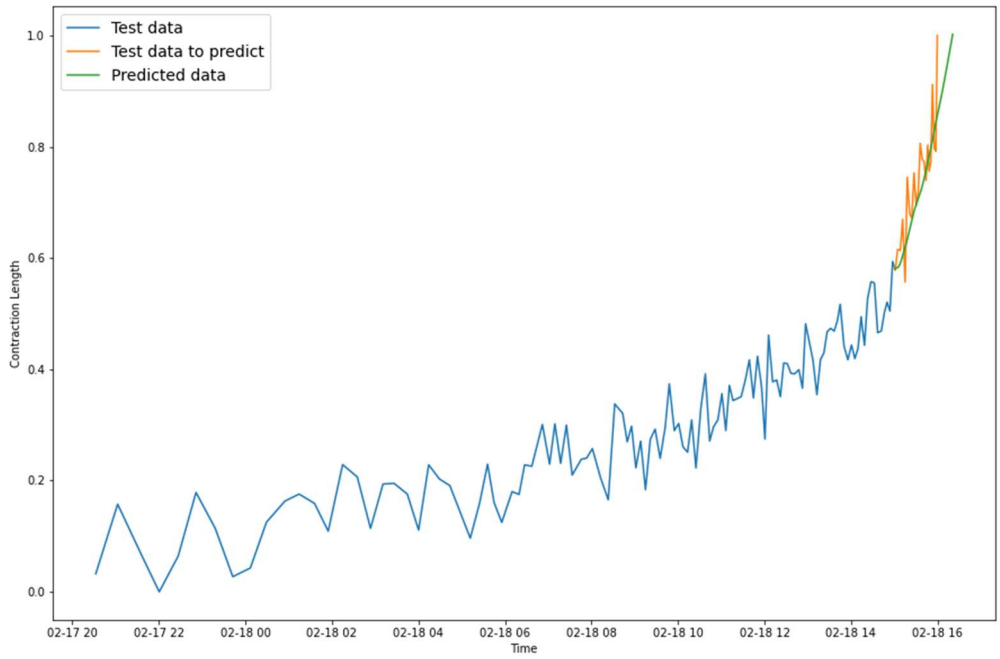
Az előző tanító adatsoron overfitted modellek túltanulását, valószínűleg az okozta, hogy a használt modell túl komplex volt az adatsor méretéhez képest. Ezért ezek a túltanult modellek újra tanításra kerültek, egy dupla ekkora méretű, 100 szülés hosszú tanító és 40 szülés hosszú validációs adatsoron, 0.00001-es learning rate-el. Az újratanulás eredményei az alábbi táblázatban láthatóak:

Modell	Átlagos időbeli eltérés (perc)	Átlagos százalékos eltérés	Sikertelen előrejelzések száma
LSTM_8_2	77.97	129.94%	0
LSTM_8_4	74.14	123.57%	0
LSTM_12_2	79.69	132.82%	0
LSTM_12_4	71.16	118.6%	0
LSTM_16_2	66.09	110.14%	0
LSTM_16_4	78.93	131.54%	0

8.5. táblázat - LSTM eredmények II.

Az eredmények alapján a legpontosabb előrejelzést a neurális hálózat abban az esetben adta, amikor egy LSTM réteg került alkalmazásra 8 neuronnal. Ez a modell a tényleges szülési időhöz képest átlagosan 46.66 perc eltéréssel prediktálta meg a várható időt.

A kész előrejelzés a modelleket felhasználva az alábbi formában jelenik meg az elvárt értékekhez képest:



8.11. ábra - Az LSTM_8 modellel készített előrejelzés ábrája

8.5 CNN

A konvolúciós neurális hálózatok alkalmasak arra, hogy az idősoros adatokat olyan sorozatként kezeljék, amiken ugyanúgy konvolúciós műveletek hajthatók végre, mint a képek esetében. Ezt úgy tudják megvalósítani, hogy a sorozatot lényegében egy egydimenziós képként kezelik.

A CNN-t tartalmazó neurális hálózat implementációjához Tensorflow és Keras került felhasználásra. A használt Conv1D réteget a Keras „layers” modulja tartalmazza. A Conv1D réteg a ReLu aktivációs függvényt használja.

A tanításhoz az egy Conv1D réteget tartalmazó modellek esetében egy 50 szülés adatsorát tartalmazó tanító adatsor és egy másik 20 szülés adatsorát tartalmazó validációs adatsor került felhasználásra, míg a több Conv1D réteget tartalmazó modelleknél 100 szülés hosszú tanító és 40 szülés hosszú validációs adatsor volt használva. Az adott modell tanításának befejezte után a tesztelés 50 db olyan adatsoron történt, ami csak egy szülést tartalmazott.
Az adatsorok az LSTM modelleknél már leírt módon min-max skálázással lettek normalizálva, majd tanító ablakokra, valamint elvárt értékekre bontva.

A neurális hálózat optimalizációs algoritmusaként az Adam lett kiválasztva. A modellek maximálisan 200 epoch-ig tanulhattak, és a batch_size 128 lett. Azért, hogy az esetleges túltanulás megelőzésre kerüljön Early Stopping lett alkalmazva 5-ös patience-el. Az

LSTM modellekhez hasonlóan a legjobb modell a validation loss figyelésével a ModelCheckpoint segítségével lett elmentve. A loss értéke Mean Absolute Error-ral (MAE) került számításra.

A Conv1D réteg tartalmaz egy aktivációs függvényt, ami ebben az esetben a ReLu. Mivel a vizsgált sorozat egy idősor, ezért a „padding" paraméternek a „causal" lett meghatározva. A kauzális konvolúció biztosítja azt, hogy a jövőbeli adatok ne legyenek a múltbeli pontokról hozzáférhetőek. Az optimális modell keresésekor több filter és kernel mérettel is tesztelve lettek a modellek.

A megfelelő CNN architektúrák megtalálásához egy és több Conv1D réteget tartalmazó modellek is vizsgálatra kerültek.

Az egy Conv1D réteget tartalmazó modellek szerkezete:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda_1 (Lambda)            (None, 1, 180)            0

conv1d (Conv1D)              (None, 1, 16)             518416

global_max_pooling1d (Globa  (None, 16)                0
lMaxPooling1D)

dense (Dense)                (None, 1)                 17

=================================================================
Total params: 518,433
Trainable params: 518,433
Non-trainable params: 0
_________________________________________________________________
```

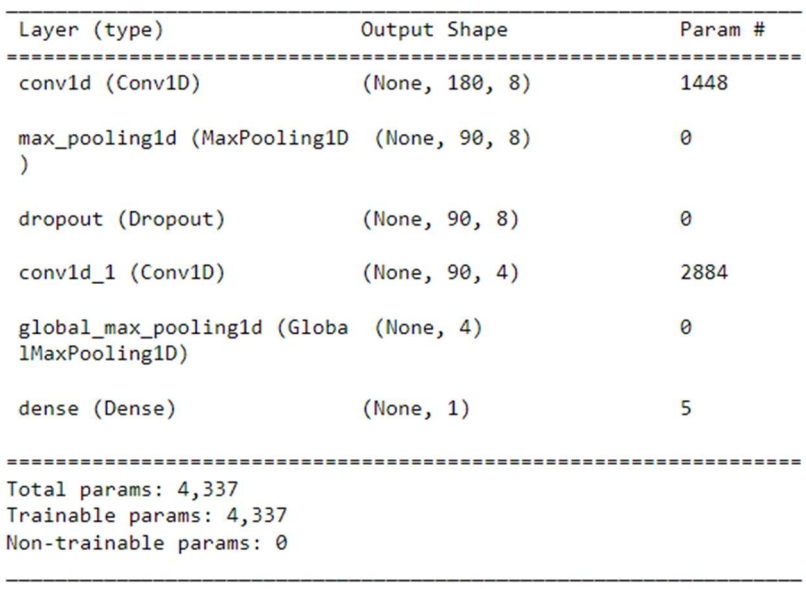
8.12. ábra - 1D_CONV_16_180 modell szerkezete

Az egy Conv1D réteget tartalmazó modellek optimalizációjának tanulási rátájának 0.001 lett meghatározva. A Conv1D réteg után még egy GlobalMaxPooling1D réteg került. Ezeknek a modelleknek az eredményeit az alábbi táblázat foglalja össze:

Modell	**Filter**	**Kernel size**	**Átlagos időbeli eltérés (perc)**	**Átlagos százalékos eltérés**	**Sikertelen előrejelzések száma**
1D_CONV_4_180	4	180	26.23	43.7%	0
1D_CONV_8_180	8	180	16.31	27.18%	0
1D_CONV_16_180	**16**	**180**	**12.6**	**20.67%**	**0**
1D_CONV_4_90	4	90	16.78	27.97%	0
1D_CONV_8_90	8	90	15.97	26.61%	0
1D_CONV_16_90	16	90	14.74	24.75%	0

8.6. táblázat - CNN eredmények I.

Az több Conv1D réteget tartalmazó modellek szerkezete:



8.13. ábra - A 1D_CONV_8_4-180_90 modell szerkezete

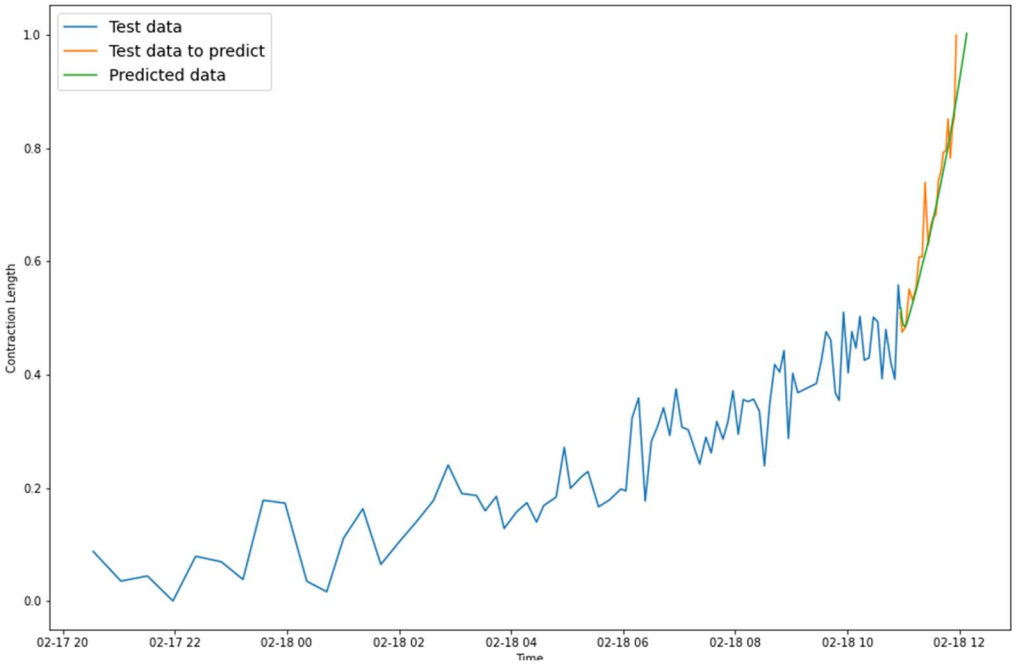
Az több conv1D réteget tartalmazó modellek optimalizációjának tanulási rátája 0.0001 volt. Az egyes Conv1D rétegek közé egy MaxPooling1D, majd további regularizációként egy dropout réteg került 0.4-es értékkel. Az utolsó Conv1D réteg után még egy GlobalMaxPooling1D réteg volt használva. Ezeknek a modelleknek az alábbi táblázatban láthatóak az eredményei:

Modell	**Filter**	**Kernel size**	**Átlagos időbeli eltérés (perc)**	**Átlagos százalékos eltérés**	**Sikertelen előrejelzések száma**
1D_CONV_8_4_180_180	8+4	180+180	26.23	43.72%	0
1D_CONV_8_4_180_90	8+4	180+90	24.39	40.66%	0
1D_CONV_12_6_180_180	12+6	180+180	19.34	32.73%	0
1D_CONV_12_6_180_90	12+6	180+90	26.63	44.38%	17

8.7. táblázat - CNN eredmények II.

Ahogy a fenti táblázatokon is látható, az egydimenziós konvolúciós hálóval készített modellek közül a legjobb teljesítményt az egy Conv1D réteggel készített 16-os filterrel és 180-as kernel mérettel paraméterezett modell adta. Ez a modell a tesztadatokon átlagosan 12.6 perc eltéréssel jelezte előre a születési időt, és az előre jelzett időintervallum hossza átlagosan 20.97%-kal tért el a valós hátralévő időtől. Az alábbi

ábrán látszik egy példa előrejelzés, ahol az előrejelzett idősor a tényleges idősorral együtt van feltüntetve:



8.14. ábra - CNN-nel előrejelzett szülés diagramja

9. Eredmények összegzése

9.1 Eredmények kiértékelése

A vizsgált idősor előrejelző technikák legjobb modelljeinek eredményeit az alábbi táblázat foglalja össze:

Modell	Átlagos időbeli eltérés (perc)	Átlagos százalékos eltérés	Sikertelen előrejelzések száma
Alapmodell	29.4 perc	49.06%	20
Holt féle exponenciális simítás	26.09 perc	44.83%	20
ARIMA	46.19 perc	76.97%	20
LSTM	46.66 perc	77.77%	0
CNN	**12.6 perc**	**20.67%**	**0**

9.1. táblázat - Eredmények összefoglalása

Mint az eredményekből az látszik, általánosságban elmondható, hogy neurális hálózatokon alapuló idősor előrejelző módszerek jelentősen jobban teljesítettek a sikeres előrejelzések létrehozásában, mint a klasszikus idősor előrejelző módszerek. A klasszikus módszerek csak a tesztesetek 60%-ban voltak képesek előrejelzésre, míg a neurális hálózatokat alkalmazók az esetek 100%-ában képesek voltak előrejelzést létrehozni. Ennek oka, hogy a statisztikai megközelítést alkalmazó klasszikus módszerek, nagyban támaszkodtak az előrejelzés időpontját közvetlenül megelőző adatokra, és nem voltak képesek a nem lineáris és komplex mintázatok felismerésére. Ezért abban az esetben, ha az előrejelzés időpontja egy olyan intervallumba esett, ahol a kontrakció hosszok szórása miatt egy hosszabb kontrakciót egy rövidebb követett, ezt a közelmúltbeli lefelé mutató trendet túl nagy súllyal vette számításba és vagy egyáltalán nem is volt képes predikciót létrehozni, vagy az előrejelzés kívül esett a 6 órás hibahatáron. Ezzel ellentétben a neurális hálózatokat alkalmazó modellek képesek voltak a szórás ellenére felismerni a szülési idősorok trendjét, és nem a szóráson, hanem ezen a felismert általános trenden alapuló előrejelzést készíteni.

A két megközelítés módszereinek előrejelzési eredményei nagyban függtek attól, hogy az adott módszer mennyire volt képes konkrétan a szülési idősor jellemzőihez igazodva végezni az előrejelzést. Az alapmodellhez hasonlóan a Holt féle exponenciális simítás és az ARIMA is abban az esetben volt képes a legpontosabb predikciót létrehozni, amikor az utolsó megfigyeléseket vette figyelembe és ezek trendje éppen pozitív volt. Ennek oka, hogy az idősor alapvetően nem veszít a trendjéből a szülés előrehaladtával, az csak nőni fog a gyermek születésének bekövetkeztéig. Azonban ennek következtében az előrejelzés nagyban függ attól, hogy az előrejelzés időpontja mennyire van közel a szülés tényleges idejéhez, hiszen mivel nem képes a trend növekedésének figyelembevételére, ezért a születés idejétől távolodva egyre kisebb trendet prediktál a teljes szülésre.

Arra, hogy az előbb említett növekvő trendet modellezve készítse az előrejelzést, az egydimenziós konvolúciós neurális háló volt képes, mivel fel tudta ismerni a szülés lefolyásának jellegzetes mintázatait. Egy órával szülés tényleges ideje előtt 12.6 perc eltéréssel volt képes előrejelezni a szülés várható idejét. Az alapmodell előrejelzéséhez képest ez 28.4%-os pontosságbeli növekedést, valamint a sikeres előrejelzések 40%-os növekedését jelentette.

9.2 További fejlesztési lehetőségek

A dolgozat elkészítésekor csak a mesterségesen generált szülési adatsorok álltak rendelkezésre, azonban valós adatsorok rendelkezésre állása esetén lehetőség nyílhatna a szintetikus adatokon létrehozott modellek valós adatokon való validálására. Továbbá a páciensek adatait felhasználva lehetőség lenne a szülési adatsorok vizsgálatára azzal kapcsolatban, hogy a páciensek egyes paraméterei, mint például a kor és a testsúly, befolyásolják-e a szülés lefolyását, és amennyiben igen, akkor ez lehetőséget biztosíthatna az adatsorok kategorizálására és a kategóriáknak megfelelő modellek létrehozására.

A dolgozat megírásának időkorlátai nem adtak lehetőséget minden felhasználható modell implementálására. További munka ráfordítása esetén lehetséges lenne újabb módszerek kipróbálása, mint például a Facebook Prophet és az Amazon Forecast, vagy a bemutatott módszerekből létrehozott hibrid modellek vizsgálata is, mint például az Holt féle exponenciális simítással társított LSTM, vagy az egydimenziós konvolúció LSTM-mel való használata.

Végül hosszú távú fejlesztési célként lehetséges lenne az optimalizált modellt egy a való életben is használható alkalmazásba integrálni, ami képes lenne valós időben gyűjtött szülési adatok alapján előrejelzést készíteni a folyamatban lévő szülésekről.

10. Összefoglalás

A munkám célja egy olyan előrejelző modell felállítása volt, ami képes a szülés kezdete óta gyűjtött kontrakciós adatokat felhasználva megbecsülni a gyermek születésének várható idejét. Az általam felállítani kívánt modell alapjául az a tendencia szolgált, hogy a szülés előrehaladtával a kontrakciók közötti időintervallum csökken, a kontrakció hossza pedig nő, azaz egyre gyakrabban egyre hosszabb összehúzódások jelentkeznek.

Mivel a szülés paraméterei nagyban függnek attól, hogy az anya hányadik szüléséről beszélünk, illetve, hogy az normális vagy kóros lefolyású-e, ezért dolgozat elkészítéséhez szükséges volt szűkíteni a vizsgált szülések halmazát. A modell felállításához ebben az esetben első alkalommal szülő nők, normális lefolyású szüléseinek jellegzetességeit vettem alapul.

Mivel olyan adatbázis jelenleg még nem elérhető, ami az általam használni kívánt adatsorokat tartalmazná, ezért szükséges volt egy olyan algoritmust létrehozni, ami képes a modellek tanításához használható idősorok előállítására.

A létrehozott algoritmus alapjául a rendelkezésre álló szakirodalmi adatok szolgáltak. A szülés a gyermek megszületéséig 3 fő szakaszra bontható, ezen szakaszoknak külön-külön statisztikai jellemzőik vannak, de általánosságban elmondható, hogy a szakaszok hossza a szülés előrehaladtával exponenciálisan csökken, az egyes szakaszok alatt pedig a kontrakció hosszok addig nőnek, amíg el nem érik az adott szakaszra jellemző maximális hosszúságot és a kontrakciók közötti időintervallum pedig addig csökken, amíg el nem éri az adott szakaszra jellemző minimum értéket. Az egyes szakaszok hosszának létrehozásához, a szakirodalomból rendelkezésre álló átlagot, mediánt és standard devianciát használtam. Mivel ezekből az adatokból látható, hogy egy aszimmetrikus eloszlásról van szó, ezért egy olyan eloszlást típust kellet keresnem, ami képes ezt visszaadni. A megfelelő eloszlás keresésekor az eloszlás számításához szükséges paramétereket az átlagból és a mediánból kiszámolva képes voltam megkapni az éppen vizsgált eloszlást. Majd az ebből visszakérdezett a medián értéket a tényleges mediánnal összehasonlítva képes voltam meghatározni, hogy melyik a szülési fázisoknak legjobban megfelelő eloszlás, ebben az esetben ez a gamma eloszlás volt. Az így megkapott gamma eloszlással már megkaptam a felhasznált szakaszhosszokat, amikhez ezután már csupán a kontrakció hosszokat és intervallumokat kellett hozzárendelni. A létrehozott szülési adatsor irreguláris időközönként tartalmazta az adatokat, ahhoz viszont, hogy a különböző idősor előrejelzési technikákat alkalmazni lehessen, szükség volt az adatsor regulárissá való átalakítására.

A dolgozatomban klasszikus és neurális hálózatokon alapuló idősor előrejelző módszereket vizsgáltam. Mind két kategóriában két-két módszer került implementálásra és vizsgálatra. A klasszikus idősor előrejelző módszerek közül az Integrált Autoregresszív Mozgóátlag Modellt, azaz ARIMA-t, illetve a Holt féle exponenciális simítást vizsgáltam. A Neurális hálózatokon alapuló idősor előrejelző módszerek közül pedig az LSTM azaz Long-Short term memory és az egydimenziós konvolúciós neurális

hálózat került megvalósításra. Továbbá referencia modellként létrehoztam egy alapmodell, ami egy módosított Simple Moving Average-et használt.

Az egyes módszerek eredményei alapján elmondható, hogy a klasszikus idősor előrejelző módszerek, az előrejelzés készítéséhez az előrejelzés időpontját közvetlenül megelőző értékekre hagyatkoztak, ezáltal nem tudták felismerni az idősorok trendjét.

Ellenben a neurális hálózatokat alkalmazó előrejelző módszerek a trend komponenst felismerve és azt alkalmazva voltak képesek predikciókat készíteni, ami hozzájárult ahhoz, hogy a sikeres előrejelzések aránya 100%-ra emelkedett. A legjobb eredményeket a konvolúciós rétegeket alkalmazó modellek adták, amik ki tudták szűrni a szülésre való jellemző mintázatokat, és az alapmodellhez képest majdnem 30%-os pontosságbeli növekedést értek el.

11. Summary

My work aimed to set up a predictive model that can estimate the expected time of childbirth using contraction data collected since the beginning of labor. The model I endeavoured to set up was based on the tendency that as labor progresses, the time interval between contractions decreases, and the length of the contraction increases, i.e. longer and longer contractions occur more and more often.

Since the parameters of the birth depend to a large extent on the mother's number of previous children and whether it has a normal or abnormal pregnancy, it was necessary to narrow down the set of examined births in order to prepare the thesis. In order to set up the model, in this case, I took as a basis the characteristics of women giving birth for the first time, with a normal pregnancy.

There is currently no database available that would contain the dataset I want to use. Therefore, it was necessary to create an algorithm capable of generating time series that can be used to train models.

The algorithm created was based on the available related academic literature. Labor can be divided into 3 main stages until the birth of the child, these stages have separate statistical characteristics, but in general, it can be said that the length of the stages decreases exponentially as labor progresses, and during each stage, the contraction lengths increase until reaching the maximum length for the given stage. Meanwhile, the time interval between contractions decreases until it reaches the minimum value characteristic of the given stage. To create the length of each section, I used the mean, median, and standard deviation available from the academic literature. Since it is visible from this data that we are dealing with an asymmetric distribution, I had to look for a distribution type that can reproduce this. When searching for the appropriate distribution, I was able to obtain it by calculating the parameters required for the distribution which could be derived from the mean and median. Then, by comparing the resulting median value with the original median, I was able to determine which distribution best suited the phases of labor, in this case, it was gamma distribution. With the gamma distribution chosen I could calculate the section lengths, to which only the contraction lengths and intervals had to be assigned. The birth data series created in this way took the form of irregular intervals, but to be able to use the different time series forecasting techniques, it was necessary to transform the data series into regular intervals.

In my thesis, I investigated time series forecasting methods based on classical and neural networks. Two methods were implemented and tested in each of the two categories. Among the classic time series forecasting methods, I examined the Integrated Autoregressive Moving Average Model, i.e. ARIMA, and Holt's exponential smoothing. Among the time series forecasting methods based on neural networks, LSTM i.e. Long-Short term memory and the one-dimensional convolutional neural network were implemented. Furthermore, as a reference model, I created a base model that used a modified Simple Moving Average.

Based on the results of the individual methods, it can be said that the classic time series forecasting methods relied on the values immediately preceding the time of the forecast to make the forecast, and were thus unable to recognize the trend of the time series.

On the other hand, forecasting methods using neural networks were able to make predictions by recognizing and applying the trend component, which contributed to the increase in the rate of successful predictions by 100%. The best results were given by the models using convolutional layers, which were able to recognize the patterns typical of childbirth and achieved an increase in accuracy of almost 30% compared to the base model.

12. Irodalomjegyzék

[1] J. Hutchison, H. Mahdy and J. Hutchison, "Stages of Labor," *StatPearls,* 2021.

[2] Z. Dr. Papp, A szülészet-nőgyógyászat tankönyve, 4. ed., Budapest: Semmelweis kiadó, 2009, pp. 258-263..

[3] N. H. S. E. www.nhs.uk. [Online]. Available: https://www.nhs.uk/pregnancy/labour-and-birth/signs-of-labour/what-happens-at-the-hospital-or-birth-centre/. [Accessed 12 05 2021].

[4] P. G. Mola, in *Caring for PNG women and their newborns: A Manual for Community Health Workers and Nurses in Papua New Guinea*, New Delhi, University of Papua New Guinea Press, 2018, p. 88.

[5] J. W. A. K. J. D. A. B. Penny Simkin, Pregnancy, Childbirth, and the Newborn: The Complete Guide, Minnesota: Meadowbrook Press, 2010.

[6] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang and P. Campell, "Introduction to Machine Learning, Neural Networks, and Deep Learning," *Translational Vision Science & Technology,* vol. 9, no. 2, 2020.

[7] "Machine Learning," 15 07 2020. [Online]. Available: https://www.ibm.com/cloud/learn/machine-learning. [Accessed 12 05 2021].

[8] A.-I. Georgevici and M. Terblanche, "Neural networks and deep learning: a brief introduction," *Intensive Care Medicine,* vol. 45, pp. 712-714, 2019.

[9] S. I. Gallant, "Perceptron-based learning algorithms," *IEEE Transactions on Neural Networks,* vol. 1, no. 2, pp. 179-191, 1990.

[10] G. Bonaccorso, Machine Learning Algorithms, Birmingham: Packt Publishing, 2017.

[11] M. Altrichte, G. Horváth, B. Pataki, G. Strausz, G. Takács and J. Valyon, Neurális hálózatok, Panem Kft., 2006.

[12] N. Buduma, Fundamentals of Deep Learning, Sebastopol: O'REILLY, 2017.

[13] Microsoft_Azure, "Gépi tanulási algoritmusok," [Online]. Available: https://azure.microsoft.com/hu-hu/overview/machine-learning-algorithms/. [Accessed 12 05 2021].

[14] Microsoft, "https://docs.microsoft.com/," Microsoft, 05 08 2018. [Online]. Available: https://docs.microsoft.com/en-us/analysis-services/data-

mining/training-and-testing-data-sets?view=asallproducts-allversions. [Accessed 13 05 2021].

[15] P.-N. Tan, M. Steinbach and V. Kumar, Bevezetés az adatbányászatba, Budapest: Panem kft., 2011.

[16] M. T. Jones, "Models for machine learning," 05 12 2017. [Online]. Available: https://developer.ibm.com/articles/cc-models-machine-learning/#reinforcement-learning. [Accessed 11 05 2021].

[17] H. Abdi and L. J. Williams, "Principal component analysis," *John W iley & S ons, Inc.,* vol. 2, pp. 433-459, 2010.

[18] F. Bodon and K. Buza, "Idősorok elemzése," in *Adatbányászat*, 2014, pp. 353-363.

[19] "ibm.com," IBM, [Online]. Available: https://www.ibm.com/docs/en/streams/5.5?topic=series-regular-irregular-time. [Accessed 15 05 2021].

[20] R. J. Hyndman and G. Athanasopoulos, "Stationarity and differencing," in *Forecasting: Principles and Practise*, Malbourne, OTexts, 2018.

[21] R. Nau, "Statistical forecasting: notes on regression and time series analysis," Duke University Fuqua School of Business, 18 08 2020. [Online]. Available: http://people.duke.edu/~rnau/411home.htm. [Accessed 15 05 2021].

[22] "corporatefinanceinstitute.com," CFI Education Inc., [Online]. Available: https://corporatefinanceinstitute.com/resources/knowledge/other/autoregressive-integrated-moving-average-arima/. [Accessed 10 5 2022].

[23] R. J. Hyndman and G. Athanasopoulos, "Forecasting: Principles and Practice," Melbourne, OTexts, 2013, p. 171.

[24] G. Shmueli and K. C. Lichtendahl, "Practical Time Series Forecasting with R: A Hands-On Guide," Axelrod Schnall Publishers, 2016, pp. 89-95.

[25] A. Merabet and M. Elsaraiti, "Application of Long-Short-Term-Memory Recurrent Neural," *Applied Sciences,* vol. 11, no. 2387, 2021.

[26] C. Hu, Q. Wu, H. Li, S. Jian, N. Li and Z. Lou, "Deep Learning with a Long Short-Term Memory Networks Approach for Rainfall-Runoff Simulation," *Water,* vol. 10, no. 1543, 2018.

[27] I. Goodfellow, A. Courville and Y. Bengio, Deep Learning, MIT Press, 2016.

[28] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," arXiv, 2015.

[29] N. Ganatra and A. Patel, "Performance Analysis Of Fine-Tuned Convolutional Neural Network Models For Plant Disease Classification," *International Journal of Control and Automation,* vol. 13, no. 3, pp. 293-305, 2020.

[30] I. Koprinska, D. Wu and Z. Wang, "Convolutional Neural Networks for Energy Time Series Forecasting," in *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, 2018.

[31] R. Yamashita, M. Nishio, R. Kinh Gian Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging,* vol. 9, pp. 311-629, 2018.

[32] B. N. Oreshkin, D. Carpov, N. Chapados and Y. Bengio, "Nbeats: Neural basis expansion analysis for interpretable time series forecasting," in *International Conference on Learning Representations*, 2020.

[33] V. Lendave, "analyticsindiamag.com," [Online]. Available: https://analyticsindiamag.com/a-guide-to-different-evaluation-metrics-for-time-series-forecasting-models/. [Accessed 11 Május 2022].

[34] K. Rink, "towardsdatascience.com," [Online]. Available: https://towardsdatascience.com/time-series-forecast-error-metrics-you-should-know-cc88b8c67f27. [Accessed 11 Május 2022].

[35] P. Fergus, A. Hussain and D. Al-Jumeily, "A Machine Learning System for Automatic Detection of Preterm Activity Using," *International Journal of Adaptive and Innovative Systems,* vol. 2, no. 2, pp. 161-179, 2015.

[36] H. Dongmei, P. Jin, W. Ying, Z. Xiya and Z. Dingchang, "Evaluation of convolutional neural network for recognizing uterine," *Computers in Biology and Medicine,* vol. 113, 2019.

[37] tensorflow, "tensorflow.org," 24 04 2021. [Online]. Available: https://www.tensorflow.org/lite/guide. [Accessed 15 05 2021].

[38] H. Gottipaty, "Top AI/ML frameworks and libraries," https://www.linkedin.com/, 27 08 2020. [Online]. Available: https://www.linkedin.com/pulse/top-aiml-frameworks-libraries-hari-gottipati?trk=read_related_article-card_title. [Accessed 15 05 2021].

[39] sktime, "https://www.sktime.org/," 2020. [Online]. Available: https://www.sktime.org/en/v0.4.3/#:~:text=sktime%20provides%20dedicated%2

0time%20series,tuning%2C%20and%20evaluating%20composite%20models.. [Accessed 15 05 2021].

[40] Keras, "keras.io," [Online]. Available: https://keras.io/about/. [Accessed 15 05 2021].

[41] K. Dániel, "Középértékek," in *Valószínűségszámítás és statisztika*, Pécsi Tudományegyetem, Közgazdaságtudományi Kar, 2021, pp. 26-34.

[42] G. J. Gorwoda, ,. M. Schiff and L. L. Albers, "The length of active labor in normal pregnancies," *Obstetrics & Gynecology,* vol. 87, no. 3, pp. 355-359, 1996.

[43] L. L. Albers, "The Duration of Labor in Healthy Women," *Journal of Perinatology,* vol. 19, no. 2, pp. 114-119, 1999.

[44] H. C. Bergsjø P, "Duration of the second stage of labor," *Acta Obstet Gynecol Scand.,* vol. 59, no. 3, pp. 193-196, 1980.

[45] D. P. Doane and L. E. Seward, "Measuring Skewness: A Forgotten Statistic?," *Journal of Statistics Education,* vol. 19, no. 2, 2011.

[46] D. C. Montgomery and G. C. Runger, "Erlang and gamma distributions," in *Applied Statistics*, John Wiley & Sons, Inc, 2011, pp. 139-140.

[47] P. G. Mola, in *Caring for PNG women*, New Delhi, University of Papua New Guinea Press, 2018, p. 88.

[48] Y. W. M. M. B. L. M. A. S. M. M. A. B. M. P. Cheng, "Length of the First Stage of Labor and Associated Perinatal Outcomes in Nulliparous Women," *Obstetrics & Gynecology,* vol. 116, no. 5, p. 1127–1135, 2010.

[49] E. L. P. J. C. A. M. K. T. L. D. M. L. C. S. S. J. M. &. C. A. B. Tilden, "Describing latent phase duration and associated characteristics among 1281 low-risk women in spontaneous labor," *Birth issues in perinatal care,* vol. 46, no. 4, p. 592–601, 2019.

[50] R. P. E. T. Østborg TB, "Duration of the active phase of labor in spontaneous and induced labors," *Acta Obstet Gynecol Scand.,* vol. 96, no. 1, pp. 120-127, 2017.

[51] M. M. Ali, Z. M. Babai, E. J. Boylan és A. A. Syntetos, „On the use of Simple Moving Averages for supply chains where information is not shared," *IFAC-PapersOnLine,* %1. kötet48, %1. szám3, pp. 1756-1761, 2015.

[52] H. V. Ravinder, „Determining The Optimal Values Of Exponential Smoothing Constants – Does Solver Really Work?," *American Journal Of Business Education,* %1. kötet6, %1. szám3, pp. 347-360, 2013.

[53] J. Frost, „statisticsbyjim,” 2022. [Online]. Available: https://statisticsbyjim.com/time-series/exponential-smoothing-time-series-forecasting/. [Hozzáférés dátuma: 1 10 2022].

[54] J. Perktold és S. Seabold, „statsmodels: Econometric and statistical modeling with python,” in *Proceedings of the 9th Python in Science Conference*, 200.

[55] S. Siami-Namini, N. Tavakoli és A. Siami Namin, „A Comparison of ARIMA and LSTM in Forecasting Time Series„” in *17th IEEE International Conference on Machine Learning and Applications*, 2018.

[56] Z. Xiao és P. C. B. Phillips, „An ADF coefficient test for a unit root in ARMA models of unknown order with empirical applications to the US economy,” *Econometrics Journal,* %1. kötet1, %1. szám2, pp. 27-43, 1998.

[57] M. Azizjon, A. Jumabek és W. Kim, „1D CNN based network intrusion detection with normalization on imbalanced data,” in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2020.

13. Ábrajegyzék