**Sapientia Erdélyi Magyar Tudományegyetem**

**Csíkszeredai Kar**

**Gazdasági Informatika Szak**

**A munkanélküliség előrejelzése a székelyföldi megyékben, ARIMA modellt és mesterséges neuronhálókat tartalmazó, Django webalkalmazással**

**Végzős Hallgató:**

**Károlyi Krisztián**

**Témavezető:**

**Dr. Madaras Szilárd, egyetemi adjunktus**

**2024**

**TDK fedlap ide**

**A munkanélküliség előrejelzése a székelyföldi megyékben, ARIMA modellt és mesterséges neuronhálókat tartalmazó, Django webalkalmazással**

***Károlyi Krisztián1***

*1Sapientia Erdélyi Magyar Tudományegyetem, Csíkszeredai Kar,  
Gazdasági Informatika szak alapképzés, E-mail: karolyiakrisztian@uni.sapientia.ro*

**Témavezető:**

**Dr. Madaras Szilárd, e**gyetemi adjunktus, e-mail: madarasszilard@uni.sapientia.ro

*Sapientia Erdélyi Magyar Tudományegyetem, Csíkszeredai Kar, Üzleti Tudományok Tanszék, Csíkszereda*

A dolgozatom fő témája, hogy összehasonlítsam a hagyományos Box-Jenkins eljáráson alapuló ARIMA modellek előrejelzési teljesítményét egy előrecsatolt (*feed-forward*) és egy visszacsatolt (*rekurrens*) neurális hálózatra épülő regressziós modellel.

Ehhez Hargita, Kovászna és Maros megye 2010 január és 2022 július közötti havi munkanélküliségi rátáit elemezve és tanító adatbázisként felhasználva, Python csomagok segítségével megkerestem a legjobban illeszkedő és általánosító ARIMA (autoregresszív mozgóátlag), MLP (többrétegű perceptron), LSTM (Long Short-Term Memory) modelleket, ezekkel pedig egy féléves előrejelzést készítettem.

A teszt-adatbázisnak 2022 augusztus – 2023 július közötti megfigyeléseket használtam.  
Az általánosító-képességeket az MSE, RRMSE és MAPE mutatókkal hasonlítottam össze.

A kutatás során egy Django webalkalmazást készítettem, amely Excel fájlban előkészített idősorokat képes beolvasni, illetve lehetővé teszi, hogy a feltöltött idősorokkal tetszőleges paraméterezéssel előrejelzési modelleket készítsünk és értékeljünk, valamint az eredményeket grafikonokkal és táblázatokkal vizualizálja és összefoglalja.

Az eredményekből azt a következtetést vontam le, hogy mindhárom idősor esetében az MLP modellek teljesítettek a legjobban.

**Tartalomjegyzék**

[1. Bevezetés 6](#_Toc162362215)

[2. Szakirodalmi áttekintés 7](#_Toc162362216)

[1.1 A munkanélküliségi ráta fogalmának meghatározása 10](#_Toc162362217)

[1.2 Felhasznált statisztikai mutatók és fogalmak 11](#_Toc162362218)

[Adatbeolvasás 12](#_Toc162362219)

[3. Az idősorok elemzése 15](#_Toc162362220)

[4 Előrejelzés Box-Jenkins módszerrel 17](#_Toc162362221)

[4.1 A stacionaritás vizsgálata 18](#_Toc162362222)

[4.2 Autoregresszív és mozgóátlag modellek (AR, MA, ARMA, ARIMA) 19](#_Toc162362223)

[4.3 Autokorrelációs függvény használata mozgóátlag komponens meghatározására 20](#_Toc162362224)

[4.4 Parciális autokorrelációs függvény használata autoregresszív komponens meghatározására 21](#_Toc162362225)

[4.5 ARIMA modellek implementálása 21](#_Toc162362226)

[5 Neurális hálózatok 23](#_Toc162362227)

[5.1 Neuronok 24](#_Toc162362228)

[5.2 Aktivációs függvények 25](#_Toc162362229)

[5.3 Perceptron, MLP 26](#_Toc162362230)

[5.4 LSTM (Long short-term memory) 28](#_Toc162362231)

[5.5 Felhasznált tanítási stratégia 29](#_Toc162362232)

[5.5.1 Az adatbázis előkészítése a neuronhálók betanítására 30](#_Toc162362233)

[5.5.2 A tanítóadatok normalizálása 31](#_Toc162362234)

[5.7 Az MLP modellek implementációja 32](#_Toc162362235)

[5.7 Az LSTM modellek implementációja 34](#_Toc162362236)

[6 Eredmények 35](#_Toc162362237)

[7 A Django webalkalamzás bemutatása 40](#_Toc162362238)

[7.1 MVC 40](#_Toc162362239)

[7.2 Django 41](#_Toc162362240)

[8 Következtetések 42](#_Toc162362241)

[5. Irodalomjegyzék 43](#_Toc162362242)

# 1. Bevezetés

A munkanélküliség hosszú ideje jelentős gazdasági mutató és központi téma a közgazdaságtani kutatásokban. A munkanélküliség alakulása és változása az adott régió gazdasági egészségét tükrözi, és fontos információkat szolgáltathat a gazdasági kilátásokról.

A COVID-19 pandémia súlyos gazdasági hatásokkal járt számos országban, beleértve Romániát is. A járvány okozta korlátozások és gazdasági visszaesés jelentős mértékben befolyásolta a munkaerőpiacot és az álláskeresési helyzetet. Kis- és nagyvállalkozások egyaránt bezártak, bocsátottak el dolgozókat vagy küldték őket kényszerszabadságra.   
Az idősorok elemzése és a megbízható előrejelzések készítése kulcsfontosságú eszköz lehet az adott régió gazdaságának és munkanélküliségi helyzetének változásainak megértésében, valamint az ehhez igazított gazdasági intézkedések megalapozásában. (Davidescu, Apostu, & Paul, 2021)

Ebben a dolgozatban Hargita, Kovászna és Maros megye havi munkanélküliségi rátáit vizsgálom 2010 január és 2022 július között. Az adatokat Románia országos statisztikai hivatalának (Institutul Național de Statistică) hivatalos oldaláról töltöttem le.   
Célom, hogy statisztikai elemzést készítsek ezekről az idősorokról, valamint megvizsgáljam, hogy ezen idősorok esetében, 1 év távlatában (2022 augusztus – 2023 július) a Box-Jenkins eljárással készült ARIMA (autoregresszív integrált mozgóátlag) modellek, vagy a gépi tanuláson alapuló MLP (többrétegű perceptron), vagy az LSTM (long short-term memory) mesterséges neuronháló modellek nyújtanak pontosabb előrejelzéseket. A legjobban illeszkedő és általánosító modellekkel pedig a 2023 augusztus – 2024 január közötti időszakra készítek előrejelzéseket.

A kutatás során az adatok beolvasásához, feldolgozásához és az eredmények megjelenítéséhez egy felhasználóbarát Django webalkalmazás segítségével biztosítok felhasználói felületet, ezzel szemléletesebbé és egyszerűbbé téve a különböző statisztikai számításokat. Az volt a célom, hogy a nehezen elérhető Python modulokat felhasználóbarát módon elérhetővé tegyem és egy egységes felületbe szervezzem. A főbb funkciók a következők: statisztikai elemzés, ARIMA modell kiválasztásához szükséges tesztek, a neuronhálók specifikációja az előrejelzések számára, ezenkívül adatvizualizáció táblázatok és grafikonok segítségével. A webalkalmazás lényegében bármennyi és bármilyen idősort képes elemezni a megfelelően előkészített adatforrásokból, tehát a jövőben még fel lehet használni más idősorral végzett kutatásra is.

# 2. Szakirodalmi áttekintés

A COVID-19 világjárványt követő gazdasági körülmények kapcsán kiemelt szerepet kapott a munkanélküliség vizsgálata, ezen belül a különböző munkanélküliségi előrejelző módszertanok. A szakirodalomból ismert konvencionális előrejelzési modellek a Box-Jenkins módszertanon alapulnak. Az Box-Jenkins módszertant használták a következő idősorok előrejelzésére:

Madaras (2014) Románia országos havi munkanélküliségi rátáit vizsgálta 2005 január – 2013 június között és 1 éves távlatban készített előrejelzést ARIMA (1, 1, 4) modellel.

Adenomon (2017) Nigéria munkanélküliségi rátáit jelezte előre 4 év távlatában,   
ARIMA (2, 1, 2) modellel, 1972 és 2014 közötti éves adatok felhasználásával.

Ramli és társai (2018) Malajziában, 1960 és 2016 közötti adatok segítségével jelezték előre a munkanélküliségi rátákat 10 éves távlatban, ARIMA (2, 1, 2) modellel. MAE, RMSE, MAPE mutatókkal hasonlították össze az ARIMA és Holt-Winters modellek általánosító képességét.

Didiharyono és Muhammad (2020) Indiában jelezték előre egy évre a regionális munkanélküliségi rátákat Dél-Celebesz tartományban, ARIMA (1, 2, 1) modellel, 1968 és 2018 közötti negyedéves munkanélküliségi ráták felhasználásával.

A Box-Jenkins eljárással való előrejelzésre a különböző mesterséges neuronháló (ANN) modellek reális alternatívát jelentenek, számos kutatás foglalkozott már azzal, hogy a két megközelítést összehasonlítsák különböző idősorok esetében, és számos alkalommal a gépi tanulás (*Machine Learning*) jobban teljesített.

Davidescu és társai (2021) eredményei alapján a NNAR (neural network autoregression) típusú neuronháló modell jobban jelezte előre a romániai munkanélküliségi rátákat 2018 és 2020 között, mint a SARIMA, amely a szezonalitást is képes kezelni.

Ajoodha és Mulaudzi (2020) Dél-Afrika éves munkanélküliségi rátáinak előrejelzésére készítettek összehasonlító elemzést, ahol háromféle hagyományos – köztük az ARIMA – és hétféle gépi tanulást használó modell – köztük az MLP és LSTM – általánosító képességét vizsgálták, a MAPE (átlagos abszolút százalékos hiba) mutató és R2 alapján. A legjobbnak az MLP, majd az ARIMA, végül az LSTM mutatkozott.

Madaras (2018) Hargita és Brassó megye esetében megállapította, hogy rövidtávon a mesterséges neuronháló-alapú NAR (nemlineáris autoregresszív) modell, középtávon a Box-Jenkins eljárással készített ARMA (1, 1) modellek nyújtottak pontosabb becséleseket.

Tufaner és Sözen (2021) Törökországban végeztek hasonló kutatást, MSE alapján hasonlították össze az ARIMA (3, 1, 2) modell előrejelző pontosságát egy két rejtett réteggel rendelkező, logisztikus (szigmoid) aktivációs függvényt használó MLP neuronhálós modellel. Az MLP modell sokkal jobban teljesített.

D. S. Yamacli és S. Yamacli (2023) Szintén Törökországban kutatták ezt a területet, főleg arra voltak kíváncsiak, hogy hogyan teljesítenek az ARIMA és az ANN modellek a COVID-19 világjárvány által érintett időszak alatti megfigyelések felhasználása mellett és anélkül. Az derült ki, hogy az ARMA (2, 1) modell a járvány időszakán kívül jobban teljesített, azonban a bizonytalan időszakban a két rejtett rétegű, előreterjesztő (feed-forward) típusú neuronháló modell lényegesen kevesebbet hibázott, mint az autoregresszív mozgóátlag modell.

Az MLP (*Multilayer Perceptron*) egy előreterjesztő mesterséges neurális hálózat, ahol neuronokat szervezünk (rejtett) rétegekbe, valamilyen aktivációs függvényt használva, ami az adott bemenetekre valamilyen kimeneteket számít, a neuronok közötti kapcsolatok aktuális súlyai alapján. A munkanélküliségi rátákon kívül számos tudományos cikkben vizsgálták ennek előrejelzési teljesítményét a gazdasági és pénzügyi idősorok terén.

<https://scholar.google.com/scholar?q=MLP+economic+time+series+forecasting&hl=en&as_sdt=0%2C5&as_ylo=2010&as_yhi=>

https://scholar.google.com/scholar?hl=en&as\_sdt=0%2C5&as\_ylo=2010&q=MLP+financial+time+series+forecasting&btnG=

<https://www.sciencedirect.com/science/article/pii/S0169207010000920>

Az LSTM (Long short-term memory)…  
Különböző szerzők vizsgálták az előrejelző modellek teljesítményét. [A hibaváltozók tesztelésére leggyakrabban használt statisztikai mutatók: MAPE (mi az), ki használta melyik cikkben]

<https://arxiv.org/ftp/arxiv/papers/1803/1803.06386.pdf>

[XY evszam arra a következtetésre jutott, hogy az MLP sokkal jobban teljesít az LSTM-hez képest a tőzsde index előrejelzés területén. 3-4 cikk]

Davidescu AA, Apostu S-A, Paul A. Comparative Analysis of Different Univariate Forecasting Methods in Modelling and Predicting the Romanian Unemployment Rate for the Period 2021–2022. Entropy. 2021; 23(3):325. <https://doi.org/10.3390/e23030325>

## A munkanélküliségi ráta fogalmának meghatározása

A dolgozat során felhasznált adatok a Romániai Statisztikai Hivataltól (INSTITUTUL NATIONAL DE STATISTICA) származnak. Az ő módszertanuk a következőképpen definiálja a munkanélküliséget és a munkanélküli rátát:

A **munkanélküliek** a BIM (Biroul Internaţional al Muncii) szerint azok a 15-74 év közötti személyek, akik egyidejűleg teljesítik a következő három feltételt:

* a mérés pillanatában nincs bejelentett munkahelyük
* a következő két héten belül munkába tudnának állni
* az elmúlt négy hétben aktívan munkát kerestek.

A **munkanélküliségi ráta** egy százalékos arányszám: a munkanélküliek arányát mutatja a munkaerőhöz viszonyítva egy adott térségre. Tehát beszélhetünk országos, regionális vagy megyei munkanélküliségi rátáról.

A gazdaságilag aktív népesség a bázisidőszakban az áruk és szolgáltatások előállítására rendelkezésre álló munkaerőt biztosító valamennyi személyt magában foglalja, beleértve a foglalkoztatottakat és a munkanélkülieket is. (INSSE, 2016)

Tehát a munkanélküliségi ráta megmutatja, hogy a munkaképes lakosság hány százaléka nem rendelkezik a mérés pillanatában munkahellyel, azonban képes lenne és szeretne dolgozni.

A következő fejezetekben ismertetem a statisztikai mutatókat, modelleket, amelyeket felhasználtam a dolgozat során.

## Felhasznált statisztikai mutatók és fogalmak

**Átlag**: A megfigyelések összegét elosztjuk a megfigyelések számával.

**Szórás**: A szórás azt jelzi, hogy a megfigyelések mennyire térnek el az átlagtól.

**A variancia:** Aszórás négyzete, vagyis az átlagtól való eltérések négyzeteinek az átlaga.

**Medián:**A sorrendbe állított x1, x2, ..., xn megfigyelések középső megfigyelése. Ha n páratlan, akkor egészen egyszerű; a medián a (n + 1) / 2 sorrendű megfigyelés.

Ha n páros, akkor 2 középső megfigyelés van, ilyenkor a kettő szám átlaga adja a mediánt:

Az előrejelzések pontosságának megállapításához, a szakirodalomból ismert (\*\*\*\*összes ide felsorolva) három mutatót használtam, amelyek minél alacsonyabbak, a modellnek annál pontosabbak az előrejelzései.

az **átlagos négyzetes eltérést** (Mean Squared Error, MSE), amely a tényleges és becsült adatok közötti különbségek négyzeteinek az átlaga:

**a relatív átlagos négyzetes** eltérések gyökét (Relative Root Mean Square Error, RRMSE), amely relatívan, normalizálva adja meg a hibákat, százalékos értékben. Az RRMSE az MSE-t normalizálja az aktuális értékek átlagával, és azok szórásával.

Az **átlagos abszolút százalékos eltérés** (MAPE) százalékban kifejezve mutatja meg, hogy mennyire nagy az átlagos eltérés a tényleges és a becsült értékek között.

*MAPE* =

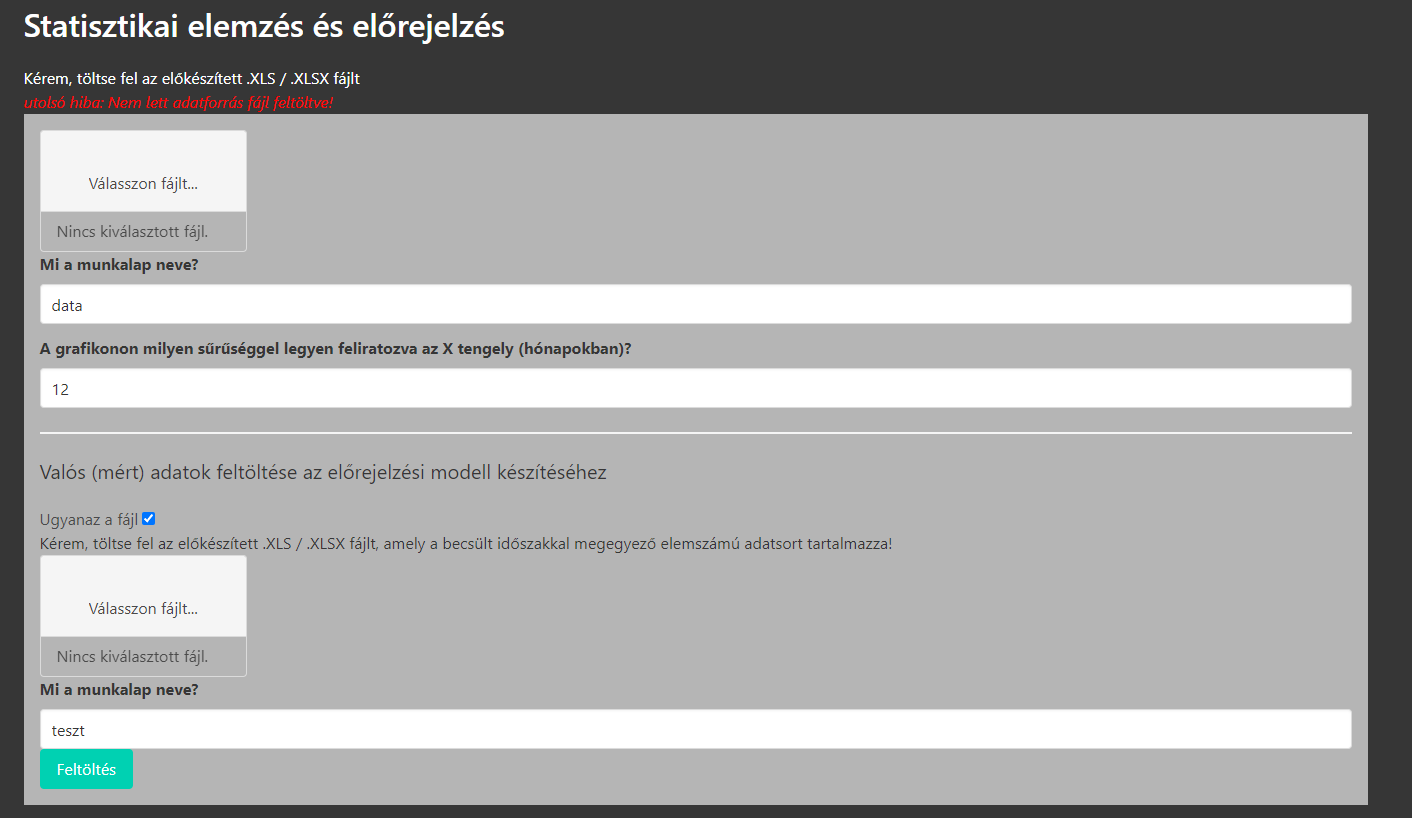
Davidescu és társai (2021) a RMSE MAE és MAPE teszteket használták.

RMSE https://arxiv.org/ftp/arxiv/papers/1803/1803.06386.pdf

## Adatbeolvasás

A romániai Nemzeti Statisztikai Hivatal (INSSE) ingyenesen hozzáférhető adatbázisából (TEMPO Online) letöltöttem Hargita, Kovászna és Maros megye 2010 január – 2023 július közötti havi munkanélküliségi rátáit, amelyet beolvastam a Python programomba, két részre osztva: az első rész a 2010 január – 2022 július közötti időszak volt (150 megfigyelés mindegyik megyére). Ez az úgynevezett tanítóhalmaz, amelyet megvizsgálok az előrejelzési modellek készítése előtt, és ami egyúttal az előrejelzési modellek illeszkedését, „tanulását” biztosítja majd. A második szakasz a 2022 augusztus – 2023 július időszak volt (12 megfigyelés), amelyet arra használok fel, hogy a „betanított” statisztikai modellekkel ugyanerre az időtávra készített előrejelzések pontosságát meghatározzam. A legjobban teljesítő modellel fogok további 6 hónapra (2023 augusztus – 2024 január) előrejelzést készíteni.

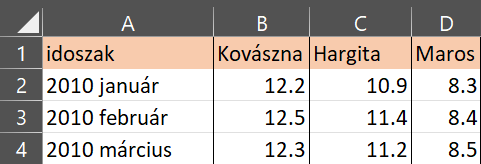
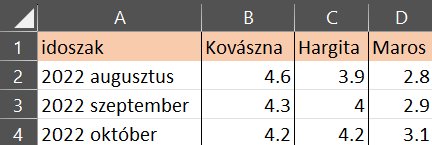
Az adatok beolvasása egy egyszerű HTML űrlap segítségével történik, ez a Django webalkalmazás kezdőlapja, amely a 1. ábrán látható:



**1. Ábra**: a webalkalamzás kezdőlapja

Az adatok egy vagy kettő előkészített Excel fájlban kell legyenek, elkülönítve a tanító- és a teszt adatokat. Egyetlen fájl esetén két külön munkalapra kell helyezni az adatokat. Több idősort is be tud olvasni a program, feltéve, hogy azok egymás mellett elhelyezett, egyenlő elemszámmal rendelkező oszlopok. Az első oszlopot a megfigyelések független változójának olvassa be a program, vagyis ebből lesz az x tengely feliratozása, ami az én esetemben a megfigyelések időpontjait tartalmazó oszlop. Az Excel munkalapnak az első sorát pedig fejléckánt kezeli az alkalmazásom, tehát az idősorok megnevezéseit innen veszi át. A 2. ábrán egy helyesen előkészített Excel fájl két munkalapját (tanítóadatok és tesztadatok) szemléltetem.

**2. ábra:** Példa a Django webalkalamzás számára előkészített tanító- és teszt adatokat tartalmazó Excel táblázatainak szerkezetére



A Python programom az objektumorientált paradigmát követi, így minden beolvasott idősor egy Stat osztály példánya (models.py) lesz, amelynek különböző adattagjai (tanító- és tesztadat lista, mutatók, ARIMA, MLP, LSTM modell példány, diagrammok, srtb.) és függvényei vannak.  
A sikeres adatbeolvasást és feldolgozást követően a program először egy HTML táblázatban megjeleníti a beolvasott adatokat ellenőrzés céljából, majd egy közös grafikonon ábrázolja a beolvasott idősorokat úgy, hogy az y tengely határai és beosztása automatikusan alkalmazkodnak a megfigyelések szélsőértékei alapján, valamint az x tengely beosztása az űrlapban beállított gyakoriság szerint történik.



***3. ábra:*** *A Székelyföldi megyék regionális munkanélküliségi rátáinak grafikonja*

# 3. Az idősorok elemzése

Az előrejelzések készítése előtt fontos megvizsgálni az idősorok alapvető statisztikai tulajdonságait, mivel iránymutató szerepe lesz a későbbiekben. Ebben a részben az adatok eloszlását is megvizsgálom.

A harmadik ábrán látszik, hogy Maros megyében szinte végig a legalacsonyabb a munkanélküliségi ráta, amely a megye ipari fejlettségének, ezáltal a több munkahelynek is tulajdonítható, míg Kovásza megyében a legmagasabb. Sok periódusban megfigyelhető, hogy télen magasabb volt a mutató, mint a többi évszakban, ez például a szezonális munkákhoz (pl. építkezések) is köthető. Összességében 2020 tavaszáig csökkenő trend volt megfigyelhető Székelyföldön.   
A diagram alatt egy statisztikai összefoglaló táblázatot is készít számunkra a program, amely a három megye 2010 január és 2022 július közötti idősorai esetében így néz ki:

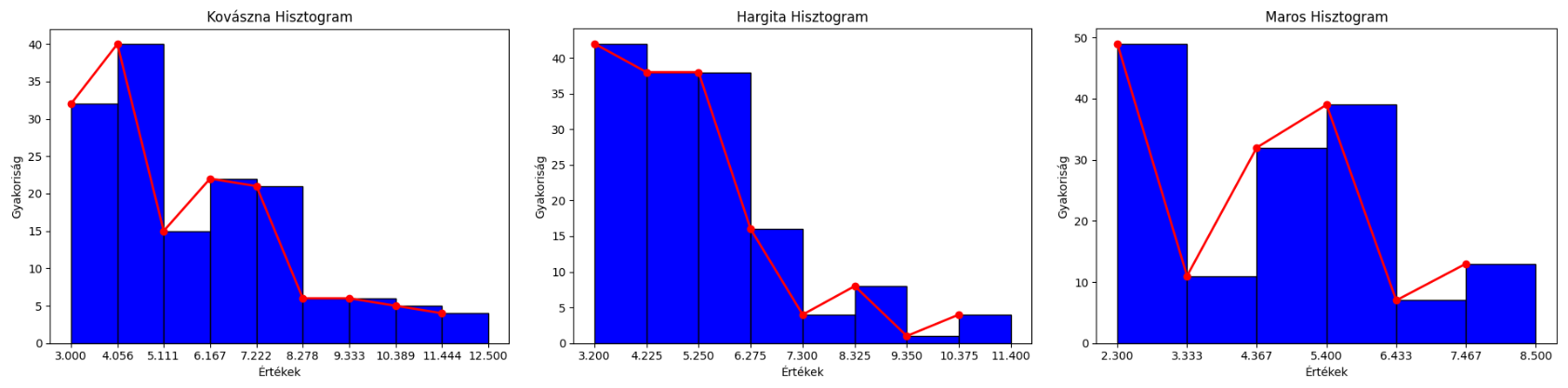
**1. táblázat:** A székelyföldi megyék munkanélkülisági rátáinak statisztikai adatai

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| mutató | Kovászna megye | | Hargita megye | | Maros megye | |
| Átlag | 5.98% | | 5.43% | | 4.69% | |
| Szórás | 2.26% | | 1.65% | | 1.73% | |
| Variancia | 5.13% | | 2.71% | | 3.01% | |
| Medián | 5.20% | | 5.20% | | 4.70% | |
| Minimum | 3.0% | 2019 május | 3.2% | 2021 november | 2.3% | 2020 május |
| Maximum | 12.5% | 2010 február | 11.4% | 2010 február | 8.5% | 2010 március |
| Téli átlag | 6.29% | | 5.94% | | 4.91% | |
| Tavaszi átlag | 5.95% | | 5.31% | | 4.75% | |
| Nyári átlag | 5.89% | | 5.13% | | 4.51% | |
| Őszi átlag | 5.76% | | 5.31% | | 4.59% | |

Kovászna és Hargita megyében a vizsgált időszakban 2010 februárjában volt a legmagasabb a munkanélküliségi ráta (12.5% valamint 11.4%), míg Maros megyében 2010 márciusában, 8.5%-os értékkel. Ez bizonyára a 2008-ban kirobbant gazdasági világválság hatása, amely elérte Romániát is. Az ország nagymértékben ki volt téve az ingatlanspekulációnak, és külföldi banki tőkének. Az ország gazdasága jellemzően az alacsony és közepes képzettségű munkaerőt használó, viszonylag kevés technológiát felhasználó és kevés hozzáadott értékű iparágakon alapult. A gazdasági recesszió miatt rengeteg munkahely szűnt meg, vagy jelentősen csökkentette dolgozóinak létszámát. (Georgeta, 2015)

A táblázatban is látható, hogy a téli hónapokban átlagosan magasabb volt a munkanélküliség, mint a többi évszakban. Maros megyében 2020 májusában volt a legalacsonyabb a mutató, 2.3%, Hargita megyében 2021 novemberében 3.2%, míg Kovászna megyében 2019 májusában 3% volt. Sajnos a koronavírus járvány miatt 2020 tavaszától 2021 novemberig jelentős ütemben növekedett a munkanélküliek száma, amelyhez a járvány visszaszorítására irányuló intézkedések jelentősen hozzájárultak. Azt, hogy Maros megyét miért érintette kevésbé a koronavírus járvány, azt egy külön kutatás keretein belül lehetne megválaszolni.

Megnéztem, hogy az adatok követik-e a normál eloszlást, mert, ha nem normál eloszlású az idősor, nagy valószínűséggel torzítják a középértékeket a kiugró értékek. A programom az idősorok hisztogramjait is automatikusan elkészíti, valamint lefuttatja rajtuk Kolmogorov-Smirnov próbát a normál eloszlásra tekintve.



**XY. ábra:** az adatok eloszlása

A nullhipotézis az, hogy a vizsgált idősor empirikus eloszlása megegyezik a feltételezett eloszlással. Ha a p-érték kisebb, mint 0.05, akkor elutasítjuk a nullhipotézist. Az xy. táblázatban találhatóak a KS teszt p-értékek, amely szerint egyik idősor sem normál eloszlású. Ezt figyelembe vettem a neurális hálók tanítása során, és kipróbáltam a különböző normalizációs megoldásokat, amelyek közül a standardizáció bizonyult a legjobbnak.

**XY. táblázat:** a A Kolmogorov-Smirnov próba eredményei

| Idősor | Statisztika | p-érték |
| --- | --- | --- |
| Kovászna | 1.00 | 0.00 |
| Hargita | 1.00 | 0.00 |
| Maros | 0.99 | 0.00 |

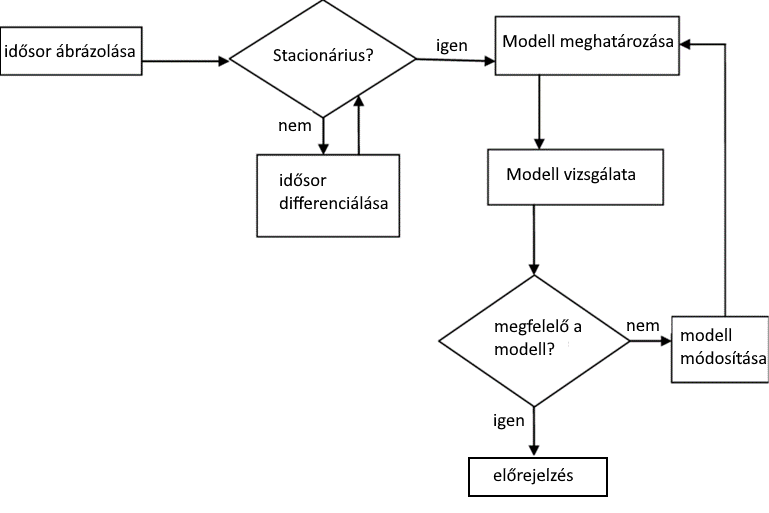
# 4 Előrejelzés Box-Jenkins módszerrel

Az eljárás nevét két fő proponenséről, George Box-ról és Gwilym Jenkins-ről kapta. Az alapgondolat az, hogy az idősorokat ARIMA (q, d, q) modellel írjuk le. A paramétereket a lehető legjobban kell behatárolni a modell pontosságának érdekében.

Az eljárás főbb lépései a következők:

1. A stacionaritás vizsgálata (pl. ADF és KPSS tesztek segítségével). Ha nem stacioner az idősor, differenciálni kell.
2. AR(p) és MA(q) komponensek paramétereinek kezdeti behatárolása PACF és ACF tesztek segítségével. Modell „tanítása” a tanítóadatokkal.
3. Modellminősítés: A legjobban illeszkedő modell (AR/MA/ARMA/ARIMA) a legkisebb Akaike Information Criterion (AIC) értékkel rendelkező modell lesz.
4. Előrejelzés készítése, majd annak pontosságának meghatározása (például MSE, RRMSE, MAPE mutatókkal, hibák eloszlása, heteroszkedaszicitás, reziduumok autokorrelációjának vizsgálata).

Ha nem elég jók az eredmények, másféle modelleket is ki kell próbálni, tehát a második, harmadik és a negyedik lépés addig ismétlődik, amíg meg találjuk a legjobb modellt.



**5. ábra**: A Box-Jenkins eljárás folyamatábrája

A következő lépésben elvégzem a stacionaritás vizsgálatát a három idősorra és ismertetem az eredményeket.

## 4.1 A stacionaritás vizsgálata

A stacionaritás az idősorok statisztikai tulajdonságainak időbeni (közelítő) állandóságát jelenti. Egy stacionárius idősor várható értéke, varianciája és autokorrelációs függvénye állandó, vagy csak időben állandó konstans eltolódásokkal változik. A stacionárius idősorok könnyebben modellezhetők és előrejelzhetőek. (Sándor, 2019) A stacionaritást a következő két teszttel vizsgáltam:

Az **Augmented Dickey-Fuller (ADF) teszt** nullipotézise (H0) az, hogy az idősorban van egységgyökér, vagyis az idősor nem stacionárius míg az alternatív hipotézis (H1) az, hogy nincs egységgyökér, tehát az idősor stacionáriusnak mondható.

Ha a p-érték (szignifikancia szint) kisebb, mint 0.05, akkor elutasítjuk a nullhipotézist, tehát az idősor stacioner, mert nincs kimutatható egységgyökér.   
A programomban a Python *statsmodels.tsa.stattools* csomagjából az *adfuller* függvényt használtam fel a teszt elvégzéséhez.

A **Kwiatkowski-Phillips-Schmidt-Shin (KPSS) teszt** nullhipotézise az, hogy az idősor szigorúan stacionárius (az ADF nullhipotézisével ellentétben), tehát nincs egységgyökér. Akkor fogadjuk el H0-t, ha a p-érték nagyobb, mint 0.05. A teszt elvégzésére a programomban a Python *statsmodels.tsa.stattools* csomagjából a kpss függvényt használtam fel.

***3. táblázat****: az ADF és KPSS tesztek eredményei*

| megye | **ADF teszt** | | | **KPSS teszt** | | |
| --- | --- | --- | --- | --- | --- | --- |
| Statisztika | p-érték | Kritikus Érték (5%) | Statisztika | p-érték | Kritikus Érték (5%) |
| **Kovászna** | -2.79 | 0.06 | -2.88 | 1.53 | 0.01 | 0.46 |
| **Hargita** | -2.64 | 0.08 | 1.48 | 0.01 |
| **Maros** | -2.37 | 0.15 | 1.65 | 0.01 |

Mindegyik megye esetében az ADF teszt p-értéke nagyobb, mint 0.05, tehát nem utasítjuk ez a nullhipotézist, vagyis az idősorok nem stacionáriusok. A KPSS tesztek p-értékei kisebbek, mint 0.05, tehát elutasítjuk a nullhipotézist, vagyis az idősorok eszerint sem stacionáriusok. Emiatt mindenképpen ki kell próbálni az olyan ARIMA modelleket, ahol a d 1, tehát legalább egyszer differenciálásra kerülnek az idősorok. A következő lépésben ismertetem az autoregresszív, mozgóátlag, autoregresszív mozgóátlag és autoregresszív integrált mozgóátlag modelleket, amelyek képesek rövidtávon pontos előrejelzéseket adni.

## 4.2 Autoregresszív és mozgóátlag modellek (AR, MA, ARMA, ARIMA)

**Az autoregressziós komponens (AR)** azt jelenti, hogy az aktuális időpontbeli becsült értéket p darab korábbi időpontbeli érték határozza meg, tehát az aktuális érték korrelál az előző időpontbeli értékekkel. Egy p-rendű autoregresszív AR(p) modell, a következőképpen írható le:

Az ismeretlen paraméterek (autoregresszív együtthatók) és az a hibaváltozó, amit **fehérzajnak** feltételezünk, vagyis olyan folyamat, amelynek várható értéke 0, varianciája konstans és autokorrelációja 0, valamint a hibaváltozó kovarianciája az minden késleltetett értékével 0. Az AR(p) tulajdonképpen egy többváltozós lineáris modell, ahol a regresszorok (független változók) a függőváltozó késleltetett értékei.

A **mozgóátlag** **(MA)** azt jelenti, hogy az aktuális időpontbeli értéket a korábbi időpontbeli hibák lineáris kombinációjaként becsüljük meg. Az MA arra utal, hogy az aktuális érték korrelál az előző időpontbeli hibákkal, és az "q" paraméter megadja a mozgóátlag rendszámát, azaz hány korábbi hibaértéket használunk az aktuális érték becsléséhez.  
Egy q rendű mozgóátlag folyamat MA(q) jelzéssel, a következőképpen írható le:

A két folyamat kombinációja az **ARMA (p, q)** (autoregresszív mozgóátlag) folyamat, amely komplexebb idősorokat is képes leírni.

P az autoregressziós folyamat rendje, q a mozgóátlag folyamat rendje, az idősorozat aktuális értéke, a konstans érték, az autoregresszív együtthatók, az adott időpontbeli fehérzaj, a mozgóátlag együtthatók.

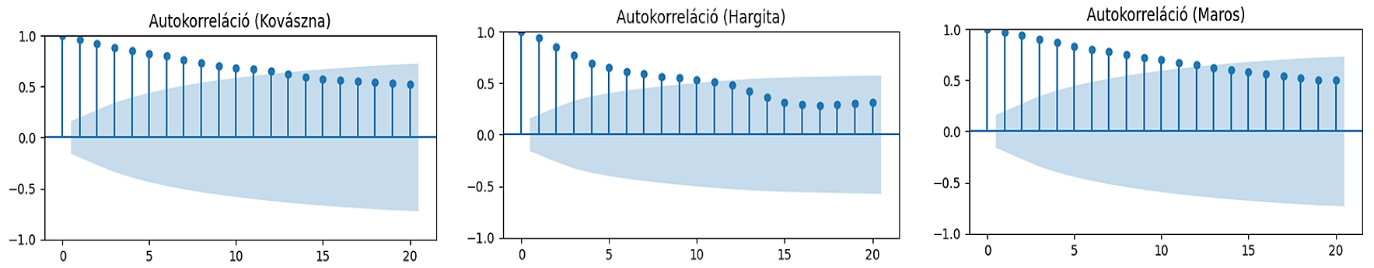
Az **ARIMA** (p, d, q) modellekben az I (integrated) azt jelenti, hogy az idősort d alkalommal differenciáljuk. A differenciálás célja az idősorok stacionaritásának eléréséhez szükséges trendek, szezonális mintázatok eltávolítása. Ilyenkor az idősor különbségeit számoljuk ki az eggyel korábbi időpont megfigyelés alapján:

(Sándor, 2019)

## 4.3 Autokorrelációs függvény használata mozgóátlag komponens meghatározására

Egy idősor autokorrelációs függvénye (ACF) az autokorrelációk sorozata:  
 ahol a k-ad rendű autokovariancia.  
Az ACF segít azonosítani a mozóátlag (MA) folyamatot azáltal, hogy megmutatja, hány időegységnyi korreláció van az aktuális és az előző időpontok között, miközben figyelmen kívül hagyja a köztes időpontokat.   
Ha egy stacionárius folyamat ACF-je teljesíti azt a feltételt, hogy   
 (Sándor, 2019)

Az általam vizsgált idősorok autokorrelációs tesztjei a következőképpen néznek ki:



***6. ábra:*** *Az autokorrelációs függvények diagramjai*

Ha MA(q) folyamat lenne, akkor az első néhány lépés után az autokorrelációk értékei hirtelen zuhannának, viszont a fent látható grafikonok nem ezt mutatják, hanem lineáris, fokozatos csökkenést, emiatt kizárható az, hogy MA(q) folyamatokról lenne szó bármelyik megye esetében is.

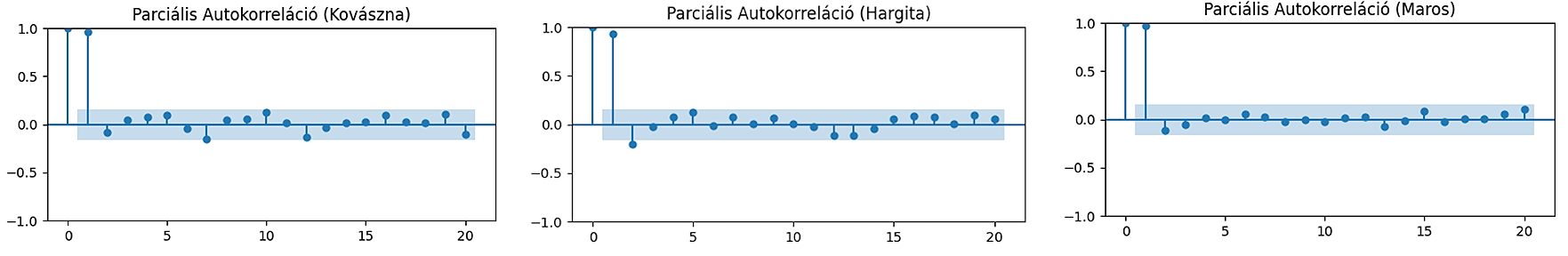
A kék beszínezett tér a vonalak mögött a 95%-os konfidencia intervallum, ami azt jelzi, hogy hol várható az autokorrelációs együtthatók értéke egy adott szinten.

Amikor egy ACF plotot elemzünk, az autokorrelációs együtthatók értékeit vesszük figyelembe, és megvizsgáljuk, hogy azok túllépik-e a biztonsági határt vagy sem. Ha egy autokorrelációs együttható a kék beszínezett térbe esik, akkor azt mondhatjuk, hogy nincs statisztikailag jelentős autokorreláció az adott eltolódásban.

## 4.4 Parciális autokorrelációs függvény használata autoregresszív komponens meghatározására

Egy AR(1) folyamat esetében a pk autokorrelációkfolyamatosan (exponenciálisan) csökkennek, ami általában minden AR(p) folyamatra igaz, azonban p > 1 rendű folyamatok esetében nem feltétlenül monoton a csökkenés. Ha egy stacionárius folyamat PACF értékei csak a p-edik időbeli lépésben nem nullák, vagyis minden további lépésben megközelítőleg nullák, akkor AR(p) folyamatról van szó. (Sándor, 2019)

Az általam vizsgált idősorok parciális autokorrelációs tesztjei a következőképpen néznek ki:



***7. ábra:*** *A parciális autokorrelációs függvények diagramjai*

Itt mindhárom megye esetében az látszik a PACF tesztek eredményein, hogy az első kettő lépésben az autokorrelációs érték 1, míg az összes többiben elhanyagolhatóak az autokorrelációs kapcsolatok, tehát valószínűleg AR (2) folyamatról beszélünk.

## 4.5 ARIMA modellek implementálása

A grafikonok azt szemléltetik, hogy valószínűleg mindhárom megye esetében az AR (2) modellel érdemes próbálkozni az előrejelzéshez, viszont a stacionaritás hiánya miatt megvizsgáltam az ARIMA modelleket is, és ezek közül kiválasztottam a legkisebb AIC (Akaike Information Criterion) értékű modellt a Sándor (2019) alapján, mert valószínűleg ez a modell fog a legjobban illeszkedni az adott idősorra. Az AIC az illeszkedés minőségét és a modell bonyolultságát összehasonlító mutató. Képlete:

, ahol *k* a a modell paramétereinek száma, *L* a modell valúszínűségi értéke, ami azt méri, hogy mennyire valószínű, az, hogy a modell előállítja a megfelelő megfigyelt adatokat a paramétereinek ismeretében. Az AIC értékeket a python-ben létrehozott ARIMA modell példány model\_fit,aic attribútumából kértem le. A következő táblázatban összefoglalom, hogy a különböző modellek milyen mértékben illeszkedtek a tanítóadatokra:

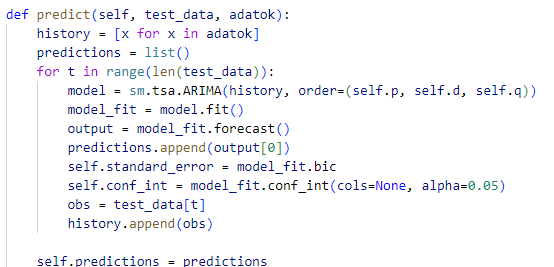
**4. táblázat**: a letesztelt ARIMA modellek AIC értékei

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| megye | AR(2) | ARIMA(2,1,0) | ARIMA(2,1,1) | ARIMA(2,1,2) | ARIMA(1,1,0) |
| Kovászna | 123.7 | 114.81 | 114.93 | **92.59** | 114.77 |
| Hargita | 114.95 | 109.28 | 106.3 | **103.58** | 109.41 |
| Maros | 3.54 | -2.47 | -0.49 | 1.35 | **-4.47** |

Jóval több modell is tesztelve lett, azonban csak a legjobb négyet tüntettem fel a táblázaton. A 4. táblázaton az látható, hogy Hargita és Kovászna megye esetében az ARIMA (2,1,2), Maros megyénél az ARIMA (1, 1, 0) jelentősen jobban illeszkedik, mint az AR (2) modell.

A programomban a statsmodels.tsa.arima.model.ARIMA ingyenesen telepíthető csomag segítségével implementáltam a modelleket és készítettem az előrejelzéseket, (Brownlee, 2023) alapján. Az adatokat a program két külön listába menti el, tanítóadatok és tesztadatok szintjén. Az előbbi arra szolgál, hogy a modelleket illesszük az idősorra, hogy ezek alapján minél jobb előrejelzéseket tudjon adni, míg az utóbbi a modell előrejelzési pontosságának meghatározására szolgál. Az én esetemben a tanítóadatok a 2010 január és 2022 július közötti megfigyelések, míg a tesztadatok a 2022 augusztus – 2023 júliusi adatok voltak.

A modellek illesztését követően megnéztem, hogy hogyan általánosítanak, vagyis mennyire pontosan becsülik meg a teszthalmaz értékeit. Ehhez (Brownlee, How to Create an ARIMA Model for Time Series Forecasting in Python, 2023) alapján, az előrelépő (*walk forward*) validációt alkalmaztam. Ilyenkor a modell az addigi ismeretei alapján egyszerre egy lépést jelez előre, majd az előrejelzett értékkel frissíti a tanítóadatoknak használt segédlistát, amivel újra illeszti önmagát.



**XY. ábra:** ARIMA előrejelzés előrelépő validációval Python-ben.

A következő táblázatban összefoglaltam a legjobban illeszkedő ARIMA modellek általánosító képességét a teszthalmazra tekintve:

**XY. táblázat:** az ARIMA modellek pontossága a teszt adatokra.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Megye | Modell | MSE | RRMSE | MAPE |
| Hargita | ARIMA(2, 1, 2) | 1.98% | 3.43% | 3.04% |
| Kovászna | ARIMA(2, 1, 2) | 7.13% | 6,55% | 4.69% |
| Maros | ARIMA(1, 1, 0) | 1.26% | 3.83% | 2.84% |

# 5 Neurális hálózatok

A mesterséges neuronhálók (ANN) lényege az emberi agyban található neurális hálózatok működésének utánzása. Ezek olyan szoftveresen vagy hardveresen megvalósított, elosztott működésű rendszerek, amelyeket jellemzően sok, hasonló vagy azonos típusú, lokális adatfeldolgozást végző műveleti elemekből, vagyis neuronok topológiája alkot. A neuronhálók párhuzamosan épülnek fel és működnek, ezért rendkívül nagy számítási kapacitásra képesek. A neuronok közötti kapcsolatok irányított gráfként írhatóak le, és a hálózati csomópontok a neuronok.

Egy neuronháló általában háromféle rétegből áll:

* Bemeneti réteg: Ez az a réteg, amely fogadja az adatokat vagy az információkat, és továbbítja azokat a háló többi részébe.
* Rejtett rétegek: Ezek az a rétegek, amelyek a bemeneti adatokat feldolgozzák és összetettebb mintázatokat fedeznek fel az adatokban. Ezek a rétegek felelősek az összetett döntéshozatalért és az adatokban rejlő rejtett összefüggések feltárásáért.
* Kimeneti réteg: Ez az a réteg, amely a neuronháló kimenetét adja. Ez lehet egy előrejelzés, egy osztályozás vagy bármilyen más kimeneti forma, amely az adott problémától függ.

A neuronhálók tanulni képesek azáltal, hogy A neurális hálózatok működése két fő fázisból áll:

* Tanulási fázis: a hálózatban valamilyen módon eltároljuk a kívánt információfeldolgozó eljárást.
* Előhívási fázis (recall): a tárolt eljárás felhasználásával elvégezzük az információfeldolgozást.

Tanulás során Az adatokból kapott visszacsatolás alapján a neuronhálók módosítják a paramétereiket (eltolási értékek és súlyok).   
Rendszerint, a tanulási fázis lassú, több iterációt, sok sikertelen tanulási szakaszt hordozhat.  
A tanítás áltában korszakokra (epoch) van lebontva, és egy-egy korszak lefuttatása után a tanítási paramétereket újra lehet hangolni. Az előhívási fázisban a pillanatnyi bemeneti értékek alapján meghatározzuk a neuronháló kimenetét.

(Brassai, 2019)

## 5.1 Neuronok

Egy neuron olyan feldolgozó elem, amely több bemenetet fogad és egy kimenetet generál.   
Az aktuális kimeneti értéket általában úgy adja, hogy a bemenetére kapott jelek súlyozott összegét egy nemlineáris transzferfüggvényben (vagy aktivációs függvény) kiértékeli.



**XY. ábra**: Egy általános neuron szerkezete. Forrás: Brassai Sándor: Neurális hálózatok, 24. oldal

A neuronok a következő tényezőket használják:

* *x1, x2, ...x.i...., xN*: a neuron bemenetei, ezeket tartalmazza az X = [*x1, x2, …xi..., xN*] vektor, ahol N a neuron bemeneteinek száma.
* A egy konstans bemenet, azaz az eltolási érték (bias), amely az érkező jelek súlyozott összegéhez hozzáadódik. Jellemzően a kimeneti rétegen kívül minden rétegnek van.
* *wi*: az i-edik bemenethez tartozó súlytényező, ezeket a súlyokat tartalmazza a   
  W = [*w1, w2 ..wi.. wN*] vektor.

A súlytényezők a lokális környezetben levő más neuronokkal való kapcsolatok irányát és erősségét reprezentálják. Ezen súlytényezőket kell finomhangolni a tanulás során.

* ϕ-vel (phi) jelöljük az aktivációs függvényt (általában nemlineáris transzferfüggvény)
* *v*-veljelöljük a bemenetek súlyozott összegét, vagyis az ingert
* *y* neuron kimenete, más szóval válasz (activation)*.*

Egy neuron kimenete tehát a bemenetek súlyozott (és eltolt) összegének, az aktivációs függvény által átalakított értéke:

(Brassai, 2019)

## 5.2 Aktivációs függvények

Az aktivációs függvények matematikai függvények, amelyek meghatározzák egy neurális hálózat rétegeinek kimenetét az adott bemeneti adatok alapján. Céljuk, hogy minden egyes neuronhoz egy aktiválási állapotot rendeljenek (aktív vagy inaktív). Ez az állapot jelzi, hogy a neuron milyen mértékben járuljon hozzá a réteg kimenetéhez., Általában ezen függvényeknek nemlineárisnak vagy differenciálhatónak kell lenniük, mivel számos optimalizálási algoritmus a hálók súlyait gradiensek segítségével hangolja. A neuronháló akkor lesz nemlineáris, ha legalább egy nemlineáris aktivációs függvényt tartalmaz. (Brassai, 2019)

A legelterjedtebb transzferfüggvények közé tartozik például a logisztikus (szigmoid), tangens hiperbolikus, ReLU, Gauss, Lépcsőfüggvény, Telítéses lineáris függvény. A kutatás során a ReLu aktivációs függvényt használtam mindegyik idősorra, mivel (Bamberger, Heckel, & Krahmer, 2023), Leírták, hogy bizonyos jelenségek approximációja esetén milyen aktivációs függvényeket érdemes használni. Például, a homogén függvények esetében a ReLU megfelelőnek bizonyult. Mivel munkanélküliségi rátát modellezi, a ráta arányt jelent, tehát a kimenet is egy arányt jelent, azaz az f függvénye 0-ad rendű homogén, azaz skála-invariáns, így elvi szempontból a ReLU függvények használhatóak.  
**A ReLU** (Rectified Linear Unit, azaz rektifikált lineáris egység):   
Egyszerűen a bemenetet adja vissza, ha az inger pozitív, és nullát, ha az inger negatív. Képlete:



A ReLu (Rectified Linear Unit) aktivációs függvény grafikonja

Rendkívül gyors és kicsi a számításigénye: Deriváltja a (nullán kívül) mindig 1. A 0 kimenetet generáló neuronok kihagyhatóak neuronhálóból, csökkentve a számításigényt és nem okoz gradiens-elhalást. A gradiens-elhalás probléma akkor fordul elő, amikor a gradiensek túl kicsik lesznek, és így a súlyokat nehéz vagy lassú frissíteni a tanulás során. Sok rétegből álló (mély) neuronhálók esetében sokkal jobb, mint például a szigmoid vagy a tangens hiperbolikus függvény. (Brassai, 2019)

## 5.3 Perceptron, MLP

A perceptron egy régebbi típusú mesterséges neurális hálózat, egyrétegű előrecsatolt neurális hálónak is nevezik. Ebben a hálózatban csak egy feldolgozó egység található, ami általában lépcsőugrás aktivációs függvényt alkalmaz, amely egy adott küszöbérték felett, illetve alatt konstans kimenetet ad. Eredetileg Frank Rosenblatt javasolta egy olyan hálózatként, amely megfelelő beállítás és tanítás után képes szétválasztani két (lineárisan szeparáható) mintahalmazt. Ebből adódóan képes bemeneteket két osztályba sorolni, tehát egy lineáris osztályozó algoritmusnak tekintjük.



Egy perceptron felépítése

Az egyszerű perceptron képtelen bonyolultabb feladatok megoldására, viszont a több perceptron rétegből álló hálók (a multilayer perceptron, MLP) sokkal komplexebb feladatok elvégzésére is képesek, például szövegfelismerés, approximáció, regresszió és előrejelzés.  
Ezek a hálók a ki- és bemeneti rétegen egy vagy több rendezett rejtett réteget tartalmaznak, ahol az információ egy irányba halad, tehát nincsenek elemi visszacsatolások (nem rekurrens), vagyis a hálószerkezet előrecsatolt. Az egyes neuronok kimenete a vele összekapcsolt, következő rétegbeli neuron (egyik) bemenetét fogja képezni. Amikor minden szomszédos neuron kapcsolódik egymáshoz, azt teljesen összekötött topológiának nevezzük. Az ilyen összetett hálózatok képesek a deep learning-re, vagyis a mély tanulásra, amely során összetettebb mintázatokat és hierarchikus jellemzőket tanulhatnak meg.

(Brassai, 2019)

## 5.4 LSTM (Long short-term memory)

## 5.5 Felhasznált tanítási stratégia

A neuronhálók tanítása egy olyan többváltozós optimalizációs folyamat, egy előre meghatározott költségfüggvény (E(ξ), például átlagos négyzetes eltérés, MSE) alapján. A legtöbb optimalizációs eljárás a gradienseket használja. (Brassai, 2019)

Én a felügyelt, más szóval az ellenőrzött tanulás (supervised learning) módszerét alkalmaztam, amely során előre megadott és előkészített tanítóhalmazt adunk meg a hálónak, amely bemeneti adatokból (független változó) és az ahhoz tartozó elvárt kimeneti értékekből (függőváltozó) alkotott párokból áll. A következő fejezetben részletesen kifejtem ennek módszerét.

A tanulás során a neuronhálóháló lépésenként, az addigi ismeretei alapján kiszámítja, hogy az adott bemeneti adatból milyen kimeneti érték következik (a tanítóhalmaz következő kimenetét becsüli, majd megnézi, hogy mennyit tévedett). A hiba-visszaterjesztéses (back-propagation) algoritmus a tanító adatokon végig iterálva, a kapott kimeneti értékekből származó hiba alapján módosítja a hálózat súlyait úgy, hogy csökkentse a hibát. A végtelen ciklusok elkerülése érdekében korlátozott lépésszámban ismételjük az optimalizációs folyamatot. Túl sok tanítási ciklus során könnyen előfordulhat, hogy a modellt túltanítjuk (overfitting). Ilyenkor a tanító halmazon ugyan a hiba csökken (egyre jobban illeszkedik rá), viszont a teszthalmaz elvárt eredményeitől egyre jobban távolodik. Ilyenkor le kell állítani a tanítási folyamatot, újra kell gondolni a háló szerkezetét, paramétereit, tanítási algoritmusát. (Brassai, 2019)

### 5.5.1 Az adatbázis előkészítése a neuronhálók betanítására

Az idősor adatokat át lehet alakítani felügyelt tanulási problémaként. Ezt úgy tehetjük meg, hogy az előző időpontokat bemeneti változókén, és a következő időpontot a kimeneti változóként kezeljük. Az idősorokból (Brownlee, 2018) 44. oldal alapján készítettem el a felügyelt tanításhoz szükséges tanító- és tesztelő párokat (*x\_train, y\_train, x\_test, y\_test*), az úgynevezett *sliding window*, a statisztikában késleltetett értékeknek nevezett (*lag*) módszerrel. Ilyenkor az idősor t-edik elemét az előző *t-1*, *t-2*, …, t-lag db megfigyelés határozza meg.

Én a bemenetekhez *lag = 3* lépést választottam, mivel ez adta a legjobb eredményeket. Ilyenkor az idősor 3 egymást követő adata lesz xi, és minden negyedik megfigyelés lesz yi.   
Például Maros megye esetében a tanítóadatok így néznek ki:

x\_train[0] = [8.3 8.4 8.5] y\_train[0] = 8.2

x\_train[1] = [8.4 8.5 8.2] y\_train[1] = 7.9

…

A januári, februári ás márciusi adatokból következik az áprilisi érték, majd a februári, márciusi és áprilisi értéktől függ a májusi megfigyelés. A tanítóadatoknál ugyanez a logika.

A webalkalmazásomban természetesen tetszőlegesen meg lehet adni a késleltetett értékek számát is.

### 5.5.2 A tanítóadatok normalizálása

A mesterséges neuronhálók tanítása során gyakran alkalmaznak valamilyen normalizációt az inputokra, annak érdekében, hogy a modell gyorsabban konvergáljon, segíthet kiküszöbölni a lokális minimumokba való ragadást és az eltűnő gradiensek problémáját azáltal, hogy az adatokat valamilyen egységes skálára hozza, így a kiugró értékek dominanciáját is visszaszorítja. A legelterjedtebb módszerek a következők:

A **standardizálás** megpróbálja az adatok átlagát nullára és szórását egyre állítani N(0,1), így közelebb hozva a mintát a normál eloszlásra. Ilyenkor a megfigyelésekből kivonjuk az átlagot, majd elosztjuk a szórással.

(forrás: https://risoprint.ro/ebooks/lazar\_ede\_bt\_e-book.pdf)

A **min-max** **normalizáció** az adatokat átskálázza [0, 1] intervallumra, az adatsor minimum és maximum érétkeinek felhasználásával.

A **robusztus skálázás** az adatokat úgy alakítja át, hogy azok eloszlása ne függjön a kiugró értékektől. Interkvartilis terjedelem (IQR) a felső és alsó kvartilisok értékeinek különbsége (Q3–Q1), ez az a tartomány, ahol az adatsor értékeinek középső fele helyezkedik el.

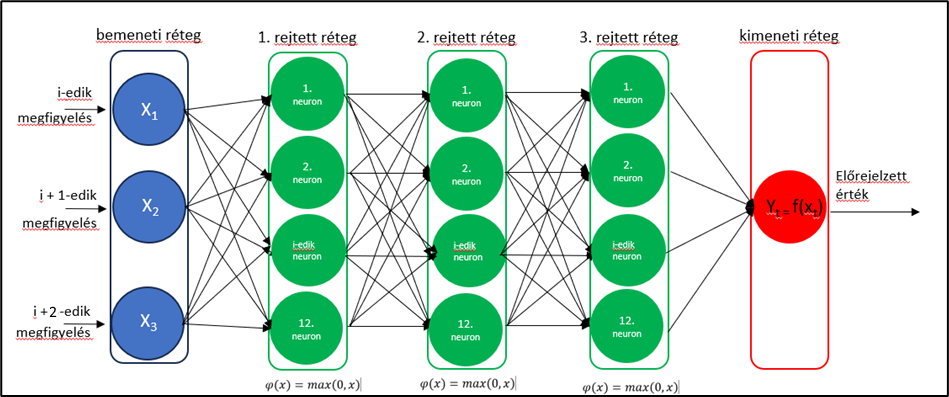
A programomban a sklearn.preprocessing osztály segítségével mindhárom skálázásra lehetőséget biztosítok az MLP és LSTM neurális hálók készítése során. Ezenkívül az adatok **logaritmizálását** is beépítettem a programba, amely csökkentheti az adatok ferdeségét, így szimmetrikusabbá és közelebb hozva őket a normális eloszlás felé.

Ilyenkor a becsült értékeket a végén visszaalakítjuk az exponenciális függvény segítségével.

A legjobb modell kiválasztása során kipróbáltam mindegyik skálázást, és a standardizálás mutatkozott a legjobbnak a tesztadatokon való teljesítményt tekintve.

## 5.7 Az MLP modellek implementációja

Az MLP számos kutatásban jól teljesített, a nehézséget a megfelelő hálószerkezet és egyéb paramétereinek megtalálása jelenti. Ezt igyekeztem megkönnyíteni azzal, hogy a webalkalmazásomban egy külön felületet biztosítok az MLP modellek készítésére és teljesítményük szemléltetésére. A modellek szoftveres megvalósítására a Python kiegészítő csomagok között ingyenesen telepíthető sklearn.neural\_network.MLPRegressor osztályt használtam. A modell felépítéséhez minden szükséges paramétert meg lehet adni az űrlapon.  
Mindhárom megye esetében sok próbálgatás után 3 db, 12 neuronból álló rejtett réteget használtam, amelyek ReLU aktivációs függvényt használtak.   
Mivel a tanítás során úgy jártam el, hogy 3 bemeneti adatból következik egy kimenet, ezért a bemeneti rétegnek mindig 3 neuronja van. Az optimalizálási ciklus maximális lépésszáma mindhárom idősor esetében 2000 lépés volt, A következő ábrán szemléltetem a neuronhálók szerkezetét.



**XY. ábra**: Az előrejelzésekhez használt MLP hálók szerkezete.

A kutatásom során az MLP és LSTM modelleknél összehasonlítottam, hogy az egyes idősorok esetében melyik optimalizációs algoritmus, illetve milyen transzformáció (standardizálás, min-max normalizáció, robosztus skálázás, logaritmizálás, vagy egyik sem) használata mellett (ADAM/LBFGS) illeszkedett, illetve általánosított a legjobban a modell. Az összehasonlítás megyénkként kilencféle modellt jelentett. Az MLP esetében lehetőség van egy súlyozási kezdőérték megadására, amely a modell eredményének reprodukálhatóságát biztosítja. Mivel állandó paramétek mellett, eltérő kezdőérték esetén máshogy konvergál a modell, ezért a webalkalamzásomban lehetőséget biztosítok két megadott érték között, a legkisebb MSE értékkel rendelkező (a legjobban általánosító modell) kezdőértékének megtalálására. Ugyanakkor egy cél MSE értéket is be lehet állítani, vagyis, ha a modell a teszthalmazra elég kicsi hibával jelzett előre, akkor a keresés véget ér. A folyamat időigényes lehet, mivel minden egyes kezdőértékkel lefuttatja a modell tanítását és tesztelését.   
A következő táblázatban láthatóak a legjobb kombinációk az egyes megyékre.

**XY. táblázat:** A legjobban általánosító MLP modellek.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Megye | MSE | RRMSE | MAPE | optimalizációs algoritmus | Kezdőérték | normalizálás |
| Hargita | 0.68% | 2.02% | 1.74% | ADAM | 49 | standardizálás |
| Kovászna | 3.25% | 4.42% | 3.00% | LBFGS | 82 | min-max |
| Maros | 0.89% | 3.25% | 2.70% | ADAM | 32 | standardizálás |

## 5.7 Az LSTM modellek implementációja

A LSTM modell esetében felétel a tanító adatok normál eloszlása. A bemeneti adatok normál eloszlását at Kolmogorov-Smirnov teszttel (Lázár Ede 133 oldal <https://risoprint.ro/ebooks/lazar_ede_bt_e-book.pdf> vizsgáltam. (Táblázat)

Táblázat

Mivel nem normál eloszlásúak, a normalizálást a \*\*\* számú\*\* képlettel végztem el.

## 6 Eredmények

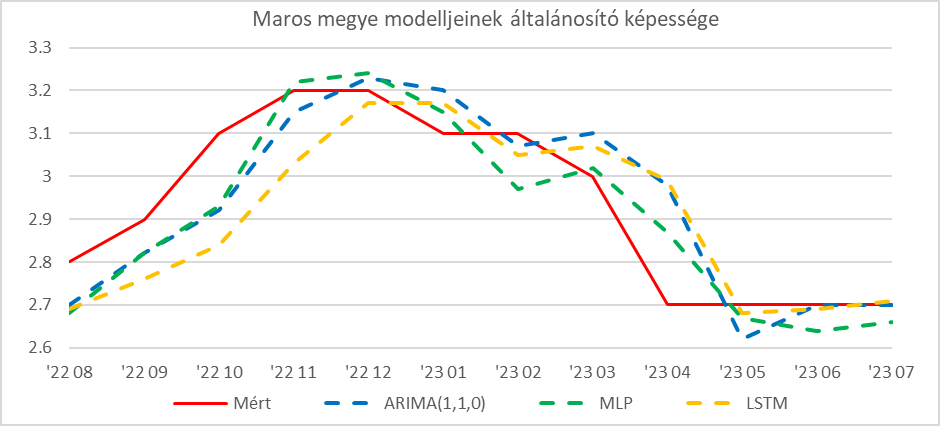
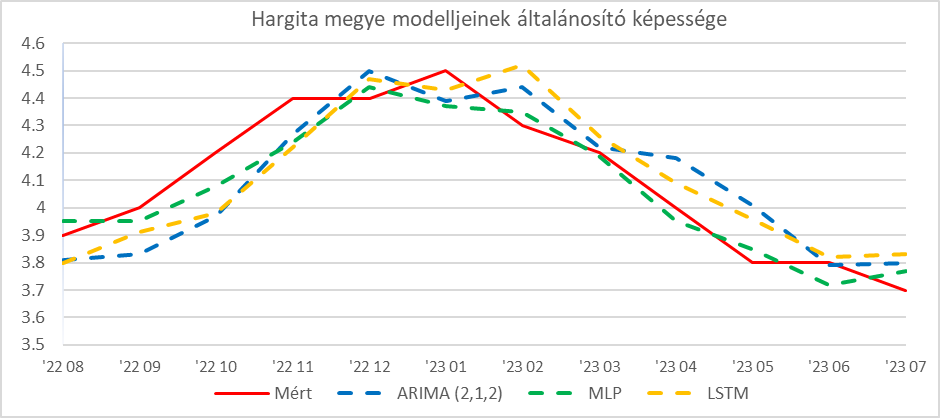
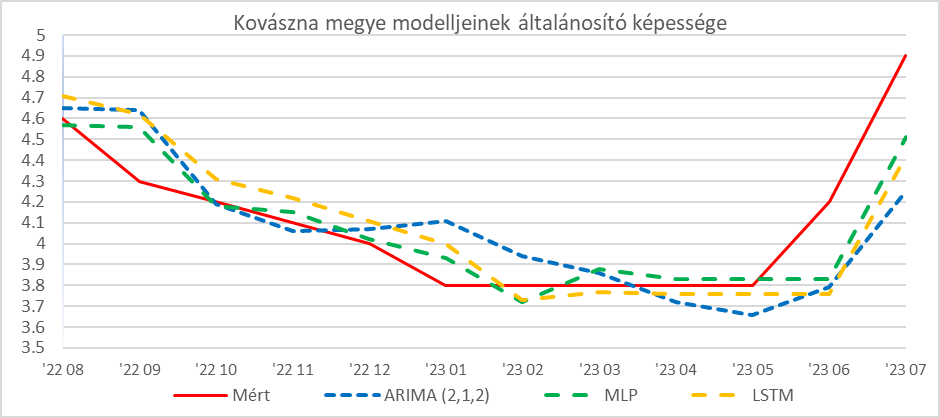
Ebben a részben összefoglalom, hogy a kiválasztott modellek hogyan teljesítettek a teszt időszakra való előrejelzésekre.

**XY. táblázat:** A megyék valódi megfigyelései a teszt időszak alatt, és a különböző modellek becslései

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| megye | **Kovászna** | | | | | **Hargita** | | | | **Maros** | | | |
| hónap | **ValódiAdat**  **(%)** | | ARIMA | MLP | LSTM | **ValódiAdat** | ARIMA | MLP | LSTM | **ValódiAdat** | ARIMA | MLP | LSTM |
| 2022 08 | 4.6 | 4.65 | | 4.57 | 4.71 | 3.9 | 3.81 | 3.95 | 3.8 | 2.8 2.7 2.68 2.69  2.9 2.82 2.82 2.76  3.1 2.92 2.93 2.84  3.2 3.15 3.22 3.03  3.2 3.23 3.24 3.17  3.1 3.2 3.15 3.17  3.1 3.07 2.97 3.05  3 3.1 3.02 3.07  2.7 2.98 2.87 2.99  2.7 2.62 2.67 2.68  2.7 2.7 2.64 2.69  2.7 2.7 2.66 2.71  2.93 2.93 2.91 2.90  0.20 0.21 0.21 0.192.8 | 2.68 | 2.68 | 2.69 |
| 2022 09 | 4.3 | 4.64 | | 4.56 | 4.62 | 4 | 3.83 | 3.95 | 3.91 | 2.9 | 2.82 | 2.82 | 2.76 |
| 2022 10 | 4.2 | 4.19 | | 4.18 | 4.31 | 4.2 | 3.97 | 4.08 | 3.98 | 3.1 | 2.93 | 2.93 | 2.84 |
| 2022 11 | 4.1 | 4.06 | | 4.15 | 4.22 | 4.4 | 4.27 | 4.24 | 4.22 | 3.2 | 3.22 | 3.22 | 3.03 |
| 2022 12 | 4 | 4.07 | | 4.02 | 4.11 | 4.4 | 4.5 | 4.44 | 4.47 | 3.2 | 3.24 | 3.24 | 3.17 |
| 2023 01 | 3.8 | 4.11 | | 3.93 | 4 | 4.5 | 4.39 | 4.37 | 4.43 | 3.1 | 3.15 | 3.15 | 3.17 |
| 2023 02 | 3.8 | 3.94 | | 3.72 | 3.73 | 4.3 | 4.44 | 4.35 | 4.52 | 3.1 | 2.97 | 2.97 | 3.05 |
| 2023 03 | 3.8 | 3.86 | | 3.88 | 3.77 | 4.2 | 4.22 | 4.19 | 4.26 | 3 | 3.02 | 3.02 | 3.07 |
| 2023 04 | 3.8 | 3.72 | | 3.83 | 3.76 | 4 | 4.18 | 3.95 | 4.09 | 2.7 | 2.87 | 2.87 | 2.99 |
| 2023 05 | 3.8 | 3.66 | | 3.83 | 3.76 | 3.8 | 4.01 | 3.85 | 3.96 | 2.7 | 2.67 | 2.67 | 2.68 |
| 2023 06 | 4.2 | 3.79 | | 3.83 | 3.76 | 3.8 | 3.79 | 3.72 | 3.82 | 2.7 | 2.64 | 2.64 | 2.69 |
| 2023 07 | 4.9 | 4.25 | | 4.51 | 4.42 | 3.7 | 3.8 | 3.77 | 3.83 | 2.7 | 2.66 | 2.66 | 2.71 |
| átlag: | 4.11 | 4.08 | | 4.08 | 4.10 | 4.10 | 4.10 | 4.07 | 4.11 | 2.93 | 2.91 | 2.91 | 2.90 |

**XY. táblázat:** A modellek becsléseinek reziduumai

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| modell | Kovászna | | | Hargita | | | Maros | | |
| hónap | ARIMA | MLP | LSTM | ARIMA | MLP | LSTM | ARIMA | MLP | LSTM |
| 2022 08 | -0.05 | 0.03 | -0.11 | 0.09 | -0.05 | 0.1 | 0.1 | 0.12 | 0.11 |
| 2022 09 | -0.34 | -0.26 | -0.32 | 0.17 | 0.05 | 0.09 | 0.08 | 0.08 | 0.14 |
| 2022 10 | 0.01 | 0.02 | -0.11 | 0.23 | 0.12 | 0.22 | 0.18 | 0.17 | 0.26 |
| 2022 11 | 0.04 | -0.05 | -0.12 | 0.13 | 0.16 | 0.18 | 0.05 | -0.02 | 0.17 |
| 2022 12 | -0.07 | -0.02 | -0.11 | -0.1 | -0.04 | -0.07 | -0.03 | -0.04 | 0.03 |
| 2023 01 | -0.31 | -0.13 | -0.2 | 0.11 | 0.13 | 0.07 | -0.1 | -0.05 | -0.07 |
| 2023 02 | -0.14 | 0.08 | 0.07 | -0.14 | -0.05 | -0.22 | 0.03 | 0.13 | 0.05 |
| 2023 03 | -0.06 | -0.08 | 0.03 | -0.02 | 0.01 | -0.06 | -0.1 | -0.02 | -0.07 |
| 2023 04 | 0.08 | -0.03 | 0.04 | -0.18 | 0.05 | -0.09 | -0.28 | -0.17 | -0.29 |
| 2023 05 | 0.14 | -0.03 | 0.04 | -0.21 | -0.05 | -0.16 | 0.08 | 0.03 | 0.02 |
| 2023 06 | 0.41 | 0.37 | 0.44 | 0.01 | 0.08 | -0.02 | 0 | 0.06 | 0.01 |
| 2023 07 | 0.65 | 0.39 | 0.48 | -0.1 | -0.07 | -0.13 | 0 | 0.04 | -0.01 |
| átlag | 0.03 | 0.02 | 0.01 | 0.00 | 0.03 | -0.01 | 0.00 | 0.03 | 0.03 |



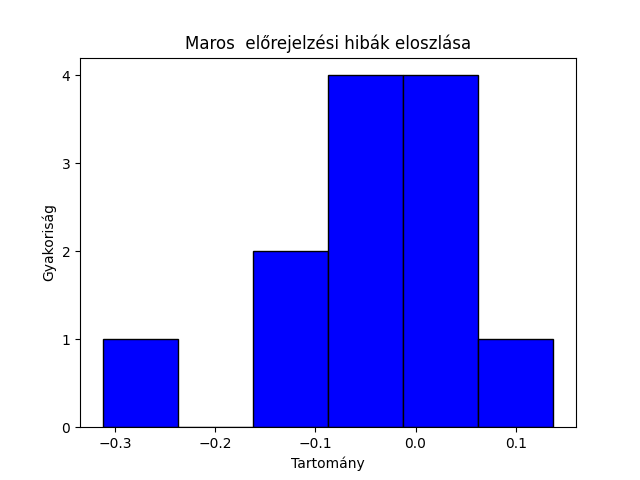
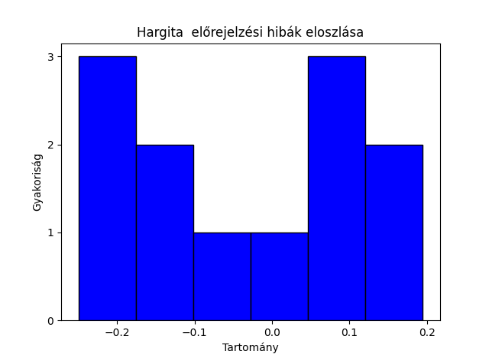
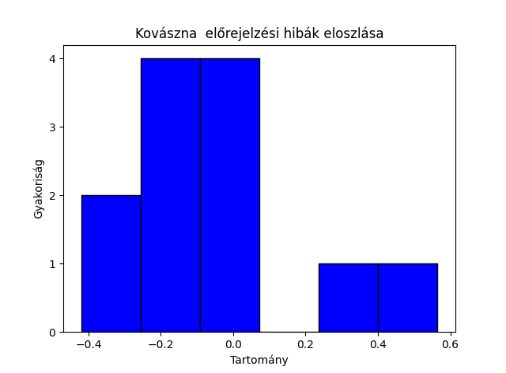
**XY. ábra:** A különböző modellek előrejelzései a teszt időszakra

Mindhárom megye esetében az MLP modell becsült a legkisebb átlagos hibával. Maros megyénél az ARIMA (1, 1, 0) pontosabb volt, mint az LSTM modell, viszont a Hargita és Kovászna megyénél az ARIMA (2, 1, 2) volt a leggyengébb.

**XY. táblázat:** A modellek becsléseinek reziduumai

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Megye** | **Modell** | **MSE** | **RRMSE** | **MAPE** |
| Hargita | **MLP** | **0.68%** | **2.02%** | **1.74%** |
| LSTM | 1.76% | 3.24% | 2.86% |
| ARIMA (2,1,2) | 1.98% | 3.43% | 3.04% |
| Kovászna | **MLP** | **3.25%** | **4.42%** | **3.00%** |
| LSTM | 5.22% | 5.56% | 4.03% |
| ARIMA (2,1,2) | 7.13% | 6.55% | 4.69% |
| Maros | **MLP** | **0.89%** | **3.25%** | **2.70%** |
| ARIMA (1, 1, 0) | 1.26% | 3.83% | 2.84% |
| LSTM | 1.88% | 4.68% | 3.48% |

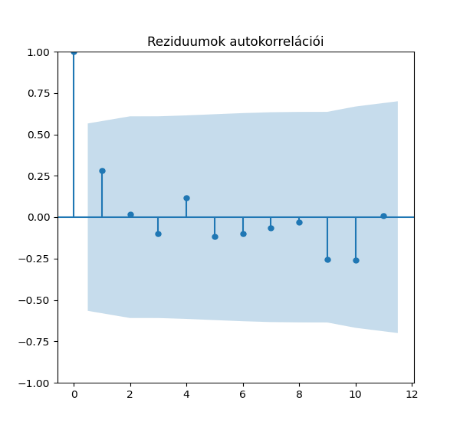
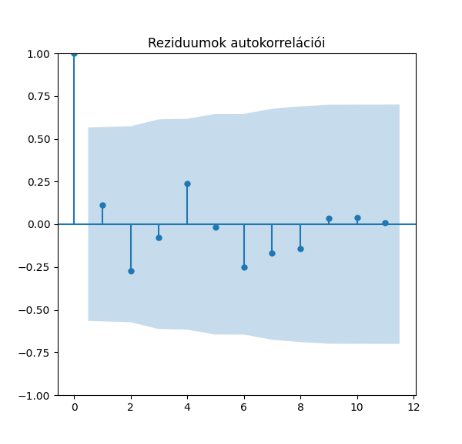
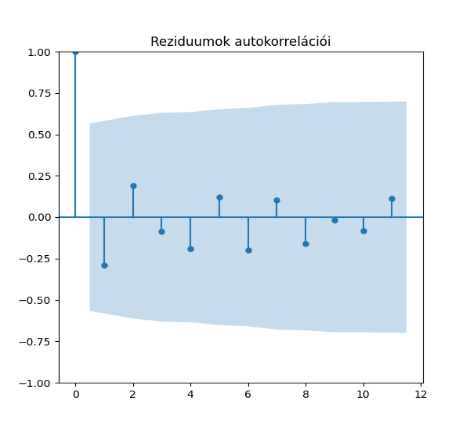
Az MLP modelleken az előrejelzés előtt elvégeztem néhány diagnosztikát: Kolmogorov-Smirnov próbával ellenőriztem (a kiugró értékek megőrzésével), hogy a hibák követik-e a normál eloszlást, (Lázár Ede 133 oldal <https://risoprint.ro/ebooks/lazar_ede_bt_e-book.pdf>



**XY. Ábra**: az MLP modellek hibáinak hisztogramjai

White-teszt segítségével megnéztem, hogy állandó-e a hibák varrianciája (homoszkedaszticitás megléte), valamint az ACF ábrázolásával vizsgáltam, hogy van-e autokorreláció a hibák között.

A Kolmogorov-Smirnov teszt p-értéke Kovászna megyénél 0.44, Hargitánál 0.56, Marosnál 0.98 volt, tehát nem utasítjuk el a nullhipotézist, tehát nem mondhatjuk azt, hogy a hibák nem normális eloszlásúak.



**XY. Ábra**: az MLP modellek becslési hibáinak autokorrelációi

Mindhárom modellnél kizárhatóak a hibák közötti korrelációk, mivel mindegyik eltolódásban bőven a 95%-os konfidencia intervallumon belül voltak az autokorrelációs együtthatók.

# 7 A Django webalkalamzás bemutatása

Ebben a fejezetben a webalkalmazásom múködését és az ahhoz felhasznált techonlógiákat ismertetem.

## 7.1 MVC



Az MVC alapvető működési elve

Az MVC rövidítés az "Model-View-Controller" (Modell-Nézet-Vezérlő) kifejezést jelenti, és egy szoftvertervezési mintázatot vagy architektúrát takar. Az MVC célja az alkalmazások strukturális szervezésének javítása, hogy könnyebben karbantarthatók és kiterjeszthetők legyenek. Az MVC három fő komponenst tartalmaz:

* Model (Modell): A modell reprezentálja az alkalmazás adatstruktúráit és logikáját. Ez felelős az adatok kezeléséért, az üzleti logika végrehajtásáért, és értesíti a View-t, amikor adatai megváltoznak.
* View (Nézet): A nézet a felhasználói felületet vagy az adatok megjelenítését kezeli. A View értesül a Model változásairól, és frissíti magát, hogy megjelenítse az aktuális adatokat.
* Controller (Vezérlő): A vezérlő a felhasználói bemeneteket kezeli, például gombok lenyomásait vagy más eseményeket. Ezután a vezérlő frissíti a Model-t vagy a View-t a felhasználói interakciók eredményeként.

Az MVC minta alkalmazása segíthet javítani az alkalmazások karbantarthatóságát, kiterjeszthetőségét és tesztelhetőségét. Sok keretrendszer és fejlesztési környezet támogatja az MVC architektúrát, például a Ruby on Rails, a Django (Python), az ASP.NET, Laravel (PHP) és mások.

## 7.2 Django

A Django egy magas szintű Python-alapú webes keretrendszer, amely biztonságos és karbantartható webhelyek gyors fejlesztését teszi lehetővé. Ingyenes és nyílt forráskódú, aktív fejlesztői közösséggel. Rendkívül alkalmas vizualizálni a kutatáshoz használt számításokat.

Ez a keretrendszer is követi az MVC szemléletet, viszont a kontroller fájl szerepét itt a views.py fájl tölti be, ahol ugyanúgy függvényekben dolgozzuk fel a szükséges adatot, majd előállítjuk dinamikusan a nézetet. A nézetek szerepét a hagyományos „view” fájlok helyett „template”, azaz html sablon fájlok veszik át.



URL-ek: Egy URL leképezőt használnak arra, hogy az HTTP kéréseket az érintett nézethez irányítsák a kérés URL-je alapján. A URL leképező képes meghatározott karakterláncok vagy számok mintázataira is illeszkedni a URL-ben, és ezeket adatként továbbítani egy nézetfüggvénynek.

Nézet: A nézet egy kéréskezelő függvény, amely fogadja az HTTP kéréseket, és HTTP válaszokat ad vissza. A nézetek az adatokhoz azokhoz a modellekhez férnek hozzá, amelyekre a kérések teljesítéséhez szükség van, és a válasz formázását sablonokra bízzák.

Modellek: A modellek olyan Python objektumok, amelyek meghatározzák egy alkalmazás adatstruktúráját, és mechanizmusokat biztosítanak a rekordok kezeléséhez (hozzáadás, módosítás, törlés) és lekérdezéséhez az adatbázisban.  
Sablonok: Egy sablon egy szöveges fájl, amely meghatározza egy fájl struktúráját vagy elrendezését (például egy HTML oldalét), ahol a helykitöltők a tényleges tartalmat képviselik. Egy nézet dinamikusan létrehozhat egy HTML oldalt egy HTML sablon segítségével, adatokkal feltöltve azt a modellből.

(Foundation, developer.mozilla.org, 2024)

Fel kell sorolni mit tud?

milyen változókat tud beolvasni a felhasználó?

+ könyv Django

<https://ieeexplore.ieee.org/abstract/document/9550822>

<https://www.irjmets.com/uploadedfiles/paper/volume3/issue_6_june_2021/11748/1628083474.pdf>

https://iopscience.iop.org/article/10.1088/1757-899X/1042/1/012019/meta



# 8 Következtetések

A Django lehetővé teszi hogy a neuronháló modell használata felhasználóbarát legyen

A nehezen elérhető Python elemzési modulok akárcsak az adat vizualizációt könnyen elérhetővé teszi, akárcsak bármelyik mai korszerű statisztikai program.

Eredményeink összhangban vannak a [xy eredményeivel], abból a szempontból, hogy az MLP neurális háló átlagosan jobban teljesített a három idősorra tekintve a MAPE hibastatisztika alapján.

# 5. Irodalomjegyzék