

TUDOMÁNYOS DIÁKKÖRI KONFERENCIA 2024

SAPIENTIA EMTE - CSÍKSZEREDAI KAR

A MUNKANÉLKÜLISÉG ELŐREJELZÉSE A SZÉKELYFÖLDI MEGYÉKBEN, ARIMA MODELLT ÉS MESTERSÉGES NEURONHÁLÓKAT TARTALMAZÓ DJANGO WEBALKALMAZÁSSAL

Hallgató: Károlyi Krisztián, gazdasági informatika szak, III. év

Témavezető: Dr. Madaras Szilárd, Üzleti Tudományok Tanszék

Csíkszereda

2024. május 7.

Tartalomjegyzék

Kivonat.....	1
1. Bevezetés.....	2
2. Szakirodalmi áttekintés	3
3. A kutatás módszertana és elméleti megalapozás.....	6
3.1. A munkanélküliségi ráta meghatározása.....	6
3.2. Felhasznált statisztikai mutatók	7
3.2.1. Előrejelzések pontosságát mérő mutatók	8
3.3. Normalitásvizsgálat.....	9
3.4. Box-Jenkins eljárás	10
3.4.1. A stacionaritás vizsgálata	11
3.4.2. Autoregresszív integrált mozgóátlag modellek.....	12
3.4.3. AR és MA komponensek meghatározása ACF és PACF segítségével.....	13
3.5. Neurális hálózatok.....	15
3.5.1. Neuron, Perceptron, MLP	16
3.5.2. Az MLP modell optimális szerkezetének kiválasztása	17
3.5.3. ReLU aktivációs függvény	17
3.5.4. Ellenőrzött tanítási stratégia	19
3.5.5. Optimalizációs algoritmusok.....	20
3.5.6. Visszacsatolt neuronhálók és az LSTM	20
4. A kutatási gyakorlati része	22
4.1. A Django webalkalmazás működése.....	22
4.2. Az idősorok elemzése.....	26
4.3. Az ARIMA modellek implementációja	30
4.3.1. A Reziduumok vizsgálata.....	32

4.4. A neuronhálók implementációja	34
4.4.1. A tanító és tesztelő adatbázis előkészítése	34
4.4.2. Az adatok standardizálása	35
4.4.3. Az MLP modellek implementációja	36
4.4.4. Az LSTM modellek implementációja	37
4.4.5. Rekurzív előrejelzés a betanított neuronhálókkal	38
5. Eredmények.....	40
5.1. modellek általánosító képességeinek összehasonlítása	40
5.2. Előrejelzések	43
6. Következtetések	45
7. Irodalomjegyzék.....	47

Kivonat

A dolgozat fő célja az, hogy összehasonlítsa a hagyományos Box-Jenkins módszeren alapuló ARIMA modellek és a mesterséges neurális hálózatok (ANN) előrejelző teljesítményét. Az összehasonlítás során egy MLP (többrétegű perceptron) és egy Long Short-Term Memory (LSTM) regressziós modellt használtam. A legjobban illeszkedő és általánosító modellekkel féléves előrejelzést készítettem, a 2023 augusztus és 2024 január közötti időszakra.

Tanító adatbázisnak Hargita, Kovászna és Maros megye 2010 január és 2022 július közötti havi munkanélküliségi rátáit, tesztelő adatbázisnak pedig a 2022 augusztus és 2023 július közötti megfigyeléseket használtam fel. Utóbbival a modellek általánosító képességét vizsgáltam MSE, RRMSE és MAPE mutatók segítségével.

Minden számítást Python-ben végeztem. A kutatás során egy Django webalkalmazást készítettem, amely Excel fájlban előkészített idősorokat képes beolvasni. Összekapcsoltam a különböző Python csomagokat, amelyek az idősorok elemzéséhez és a különböző prediktív modellek készítéséhez szükségesek. és ezeket egy felhasználóbarát, egységes webes felületbe szerveztem. A program az eredményeket grafikonokkal vizualizálja és táblázatokkal foglalja össze, ezzel megkönnyítve a modellek megfelelő szerkezetének és paramétereinek megtalálását.

Az eredményekből azt a következtetést vontam le, hogy mindhárom idősor esetében az MLP modellek jobban illeszkedtek a tanító adatbázisra, és jobban általánosítottak, mint az ARIMA vagy az LSTM modellek.

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 szolgáltat 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 álláskeresői helyzetet. Kis- és nagyvállalkozások egyaránt bezártak, bocsátottak el dolgozókat vagy küldték őket kényszerszabadságra. A munkanélküliségi ráta havi idősorainak 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 és társai, 2021).

Ebben a dolgozatban Hargita, Kovászna és Maros megye 2010 január - 2022 július közötti havi munkanélküliségi rátáit vizsgáltam. Az adatokat romániai Nemzeti Statisztikai Hivatal (románul „Institutul Național de Statistică”) weboldaláról (TEMPO Online) töltöttem le.

A célom az volt, hogy statisztikai elemzést készítsek ezekről az idősorokról, valamint megvizsgáljam, hogy ezen idősorok esetében, további egy év távlatában (2022 augusztus – 2023 július) a Box-Jenkins eljárással készült ARIMA (autoregresszív integrált mozgóátlag) modellek, vagy a gépi tanuláson alapuló MLP (multilayer perceptron), vagy az LSTM (long short-term memory) mesterséges neuronhálók nyújtanak pontosabb előrejelzéseket. A legjobban illeszkedő és általánosító modellekkel pedig a 2023 augusztus – 2024 január közötti időszakra készítettem előrejelzéseket.

A számításokat Python-ben végeztem, amelyhez egy Django webalkalmazás segítségével biztosítok felhasználói felületet. 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 egyszerre több idősort is 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. A munkanélküliségi ráták előrejelzésére sok tudományos cikkben használták a Box-Jenkins módszertant.

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.

Tóth és társai (2021) a Visegrádi országok munkanélküliségének alakulását modellezték és jelezték előre ARIMA modellekkel, 2010 és 2021 közötti havi adatokkal. Az ARIMA modelleket az AIC mutató alapján választották ki.

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 hiba becslésekkel 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, a legtöbb alkalommal a gépi tanulás (*Machine Learning*) jobban teljesített. A modellek általánosító képességét, vagyis azt, hogy a tanító adatbázisra illesztett modell milyen pontossággal tudja becsülni a tanító adatbázison kívüli adatokat (teszt adatbázist), általában a következő mérőszámokkal hasonlítják össze: átlagos abszolút hiba (MAE), átlagos abszolút százalékos hiba (MAPE), az átlagos négyzetes hiba (MSE), ennek négyzetgyöke (RMSE), illetve az RRMSE, amely az RMSE-t normalizálja a valódi adatok átlagával.

Davidescu és társai (2021) eredményei alapján az 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. A modellek teljesítményét a MAE, MAPE és RMSE mutatókkal hasonlították össze.

Ajoodha és Mulaudzi (2020) Dél-Afrika éves munkanélküliségi rátáinak előrejelzéséhez 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 és R^2 mérőszámok 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 becseleket.

Tufaner és Sözen (2021) Törökországban végeztek hasonló kutatást, MSE hiba becslés 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 az RMSE hiba becslés alapján, hogy az ARMA (2, 1) modell a járvány időszakán kívül jobban teljesített, azonban a bizonytalan gazdasági időszakban a két rejtett rétegű neuronhálós modell jobban teljesített.

A munkanélküliségi rátákon kívül számos tudományos cikkben vizsgálták az MLP és az LSTM előrejelzési teljesítményét a gazdasági és pénzügyi idősorok terén. Az eredmények erősen függenek a vizsgált idősoroktól, azonban az MLP modellek sokszor jobbnak bizonyultak.

Derbentsev és társai (2020) négyféle ANN modell, köztük az MLP előrejelző képességét vizsgálták a Bitcoin, Ethereum, és az EUR-USD árfolyamokon. Az MLP és SGBM modellek 0.46-3.71%-os MAPE hiba becsléssel jeleztek előre a teszt időszakra, így megállapították, hogy a gépi tanulóval való megközelítés valóban alkalmas a pénzügyi előrejelzések készítésére.

Bareith és Vancsura (2023) Értékpapírok árfolyamok előrejelzése során vizsgálták, hogy az ARIMA, LSTM és a döntési fák milyen pontossággal tudnak előrejelezni (MAPE hibastatisztika alapján), különböző időtávok esetében. Emellett azt is nézték, hogy milyen módon befolyásolja a prediktív modellek teljesítményét a COVID-19 és az orosz-ukrán háború alatti árfolyamok. Eredményeik szerint az LSTM jobban teljesített, mint az ARIMA, és a COVID-19 járvány által érintett időszak alatt jobban tudtak teljesíteni a prediktív modellek, mint a háború ideje alatt.

Sasu (2013) összehasonlította az MLP és az ARIMA egyváltozós előrejelző-képességét, tízféle idősort felhasználva, például foglalkoztatottsági adatokat, munkanélküliségi ráta, és banki letétek összegét. Eredményeik szerint, a MAPE hiba becslés alapján az MLP modellek jobban teljesítettek.

Oukhouy és Himdi (2023) a marokkói tőzsdeindex (MSI 20) napi árának előrejelzésén tesztelték az XGBoost, SVR, LSTM, MLP neuronhálós modelleket. Megállapították, hogy a rácskereséses optimalizációs algoritmus (Grid Search) mellett az MLP és az SVG teljesítettek a legjobban. Az MAE, MSE, MAPE, RMSE, és R^2 mutatókat egyaránt figyelembe vették.

S. Siami-Namini és A. S. Namin (2018) tőzsdeindexek előrejelzésén tesztelték az ARIMA és az LSTM modelleket. Eredményeik szerint, az RMSE mutató alapján az LSTM modellek pontosabbak voltak.

Weytjens és társai (2021) cash flow előrejelzésekre használták az MLP, LSTM ARIMA és Prophet modelleket. Az előrejelzések pontosságának kifejezésére többek között a MAE és MSE mutatókat alkalmazták. Az MSE hibastatisztika szerint az LSTM modell volt a legjobb, míg az ARIMA teljesített a leggyengébben.

Muzaffar és Afshari (2018) óránkénti elektromos terhelései értékek előrejelzésén tesztelték az ARMA, SARIMA, ARMAX és az LSTM modellek teljesítményét egynapos, kétnapos, egyhetes és egy hónapos távlatban. A MAPE és RMSE hibamutatók alapján az LSTM pontosabb előrejelzéseket adott.

Nabipour és társai (2020) teheráni értéktőzsde különböző részvénytársaságait modellezték és jelezték előre, 10 évnyi adatot felhasználva, többféle neurális hálózattal, döntési fákkal, és egyéb módszerekkel. Kutatásuk során az derült ki, hogy az LSTM modellek teljesítettek a legjobban. Az előrejelzések pontosságát MAE, MAPE, RMSE és RRMSE mutatókkal mérték.

3. A kutatás módszertana és elméleti megalapozás

Ebben a fejezetben a kutatás során felhasznált elméletet i részt mutatom be, amelyek fontosak a kutatás gyakorlati részének megértéséhez.

3.1. A munkanélküliségi ráta meghatározása

A dolgozat során felhasznált adatok Románia Nemzeti Statisztikai Hivatalától (románul „Institutul Național de Statistică”) származnak.

A munkanélküliek a BIM (Biroul Internațional al Muncii) meghatározása 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. 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. Ezt a statisztikai mutatót havonta és éves időszakokban mérik.

3.2. Felhasznált statisztikai mutatók

Ebben a részben összefoglalom a fontosabb statisztikai mutatókat és összefüggéseket, amelyeket felhasználtam a dolgozatomban, (Sándor, 2019; Sándor és Tánczos, 2019) alapján.

Átlag: A megfigyelések összegét elosztjuk a megfigyelések számával.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad (1)$$

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

$$S_x = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n (x_i - \bar{x})^2} \quad (2)$$

A variancia a szórás négyzete, vagyis az átlagtól való eltérések négyzeteinek az átlaga.

$$s_x^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (3)$$

Medián: A sorrendbe állított $x_1, x_2, \dots, x_n$ megfigyelések középső megfigyelése. Ha n páratlan, akkor a medián a $(n + 1) / 2$. sorszámú megfigyelés. Ha n páros, akkor 2 középső megfigyelés van, ilyenkor a kettő szám átlaga adja a mediánt:

$$\tilde{x} = \frac{x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)}}{2} \quad (4)$$

Determinancia-együttható (R^2): azt mutatja, hogy a magyarázó változó milyen mértékben magyarázza a célváltozó változékonyságát. Az R^2 0 és 1 közötti értékeket vehet fel. Egy 1-hez közeli R^2 érték azt jelenti, hogy a modell jól illeszkedik az adatokhoz. Azonban nem biztosít elég információt a modell által becsült értékek pontosságáról, ezért önmagában nem elég a modellek általánosító képességének vizsgálatához.

$$R^2 = 1 - \frac{\sum e_i^2}{\sum (y_i - \bar{y})^2} \quad (5)$$

3.2.1. Előrejelzések pontosságát mérő mutatók

Az előrejelzések pontosságának megállapításához a szakirodalomból ismert MSE, MAPE és RRMSE mutatókat használtam, amelyek minél alacsonyabbak (a nullához minél közelebb vannak), a modellnek annál pontosabbak az előrejelzései. Mivel a MAPE-t a MAE-ből, és az RRMSE az RMSE-ből le lehet vezetni, ezért ezek képletét is leírtam, Nabipour és társai (2020) alapján.

Jelöljük n -nel a megfigyelések számát, t -vel az időpontot, az x_t -vel a megfigyelést (valódi értéket), és f_t -vel az előrejelzett értéket a t -edik időpontban.

Az átlagos abszolút hiba (MAE) a valódi adat és a becsült érték közötti eltérések abszolút értékeinek az átlaga. Az abszolút érték miatt nem veszi figyelembe a hiba irányát.

$$MAE = \frac{1}{n} \sum_{t=1}^n |f_t - x_t| \quad (6)$$

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

$$MAPE = \frac{1}{n} \sum_{t=1}^n \left| \frac{f_t - x_t}{x_t} \right| \quad (7)$$

Az átlagos négyzetes hiba (MSE) a tényleges és becsült adatok közötti különbségek négyzeteinek az átlaga.

$$MSE = \frac{1}{n} \sum_{t=1}^n (x_t - f_t)^2 \quad (8)$$

Az átlagos négyzetes hiba gyöke (RMSE) az MSE négyzetgyöke. A megfigyelések és a modell által becsült értékek közötti négyzetes eltérések átlagos négyzetgyökét mutatja.

$$RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^n (x_t - f_t)^2} \quad (9)$$

A relatív átlagos négyzetes eltérések gyöke (RRMSE) a becslési hibák szórása. Az RRMSE az RMSE-t normalizálja a megfigyelések átlagával.

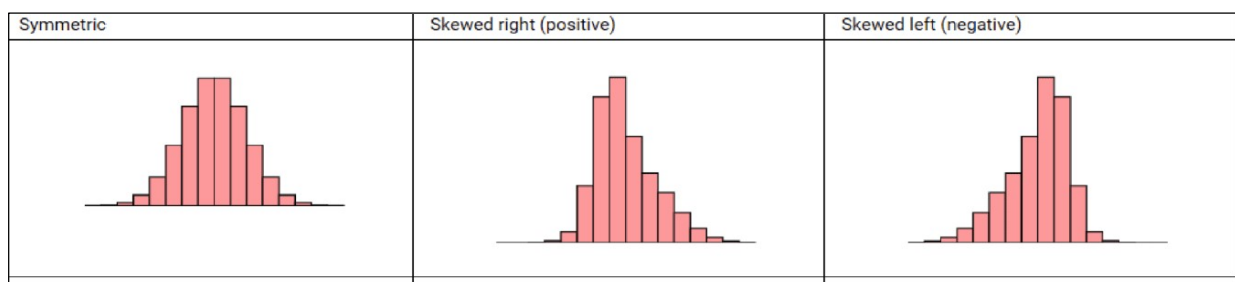
$$RRMSE = \sqrt{\frac{1}{n} \sum_{t=1}^n \left(\frac{x_t - f_t}{x_t} \right)^2} \quad (10)$$

3.3. Normalitásvizsgálat

A normális eloszlás az egyik legfontosabb statisztikai eloszlás, amelyet Python-ben is van lehetőség elemezni. A normál eloszlás szimmetrikus, vagyis az átlagtól ugyanakkora távolságra lévő értékek gyakorisága mindkét oldalon (megközelítőleg) egyenlő, és az átlagtól való távolság növekedésével az értékek gyakorisága csökken (Sándor és Tánczos, 2019; Lázár, 2022). A normális eloszlás jelölése $X \sim N(\mu, \sigma^2)$. és a sűrűségfüggvénye a (11) képlettel írható le.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right), \quad -\infty < x < \infty \quad (11)$$

A normalitás vizsgálatát érdemes hisztogram készítésével kezdeni. Ha a hisztogram alakja megközelítőleg illeszkedik a haranggörbére, az a normális eloszlás követésére utal (Lázár, 2022). Az első ábrán a szimmetrikus, és oldalra elnyúló eloszlásokra láthatunk példát.



1. ábra. Szimmetrikus és oldalra elnyúló eloszlásokat reprezentáló hisztogramok.

Forrás: algebra1.thinkport.org (2015)

A ferdeség (kurtosis) mutató azt jelzi, hogy az eloszlás mennyire csúcsos vagy lapos a normálhoz viszonyítva, azaz, hogy mennyire tömörülnek az adatok az eloszlási görbében. A ferdeség mutató pedig azt fejezi ki, hogy az eloszlás milyen mértékben és irányban tér el a szimmetrikustól. Általában a ferdeség okozza az eltérést a normális eloszlástól, és a kiugró értékek alkotják a hosszú, lapos, ritkább részt az eloszláson belül (Lázár, 2022).

A normális eloszlás követését egy adatsor esetében többféle statisztikai teszttel is meg lehet vizsgálni. A Kolmogorov-Smirnov tesztet egy adatsor eloszlásának összehasonlítására használják egy előre meghatározott eloszlással, például a normális eloszlással. A teszt nullhipotézise (H_0) az, hogy a vizsgált idősor empirikus eloszlása megegyezik a feltételezett eloszlással, míg az alternatív hipotézis (H_1) az, hogy a két vizsgált eloszlás nem egyforma. Ha a p-érték kisebb, mint 0.05, akkor elutasítjuk a nullhipotézist (Lázár, 2022).

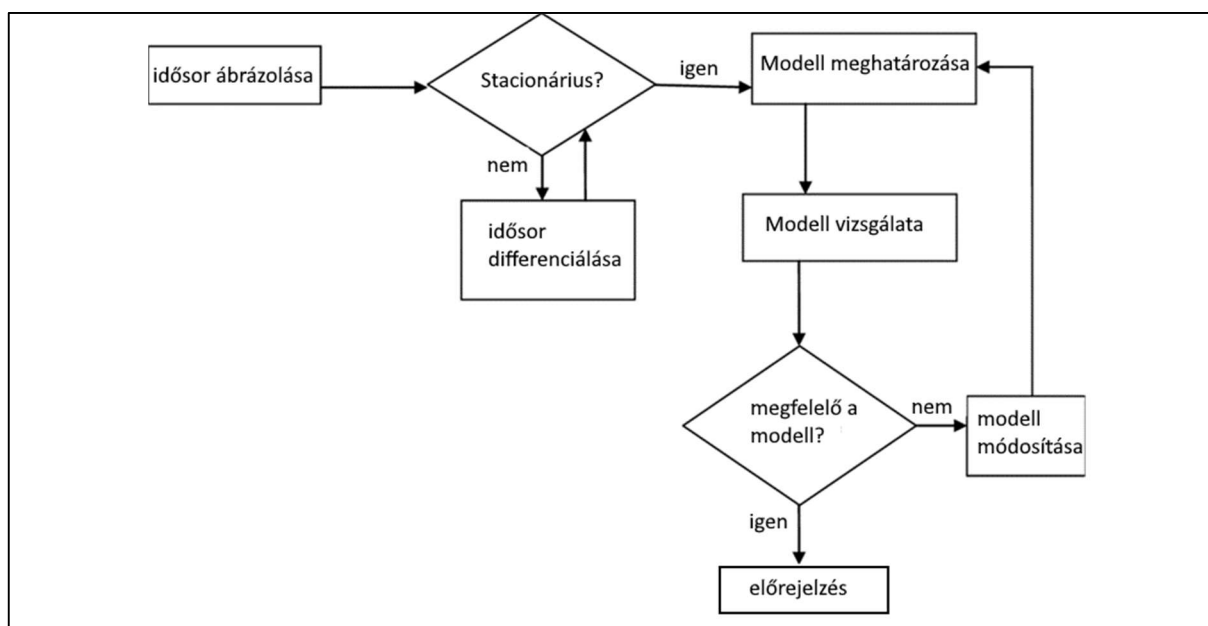
3.4. Box-Jenkins eljárás

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

Az eljárás főbb lépései a következők, amelyeket a második. ábrán is szemléltetnek, egy folyamatábra segítségével Sándor (2019) alapján.

- 1) A stacionaritás vizsgálata (pl. ADF és KPSS tesztek segítségével). Ha nem stacionárius az idősor, differenciálni kell.
- 2) AR(p) és MA(q) komponensek paramétereinek kezdeti behatárolása PACF és ACF tesztek segítségével. Modell illesztése a tanítóadatokkal.
- 3) A legjobban illeszkedő modell (AR/MA/ARMA/ARIMA) a legkisebb Akaike Information Criterion (AIC) értékkel rendelkező modell lesz. Általában itt modell becsléseinek reziduumaikat is vizsgálják, hogy fehérzajként viselkednek-e, vagyis egymástól függetlenek, mintázatot nem mutatnak és normális eloszlásúak.
- 4) A tanítóadatokon túli adatok becslése a modellel.

Ha a modell becsléseinek reziduumaik nem felelnek meg a modell diagnosztika során, új paraméterekkel kell próbálkozni (Sándor, 2019).



2. ábra. A Box-Jenkins eljárás folyamatábrája. (Saját szerkesztés)

A Box-Jenkins eljárás során a megfelelő AR/MA/ARMA/ARIMA modelleket az AIC mutató alapján választják ki. Az AIC a becsült modell illeszkedésének minőségét és a modell bonyolultságát összehasonlító mutató. Tehát az a modell illeszkedik a legjobban az eredeti értékekhez, amelynek az AIC kritérium értéke a legkisebb. Képlete:

$$AIC = 2k - 2 \ln(L), \quad (12)$$

ahol k a modell paramétereinek száma, L a modell valószínűségi értéke, ami azt méri, hogy mennyire valószínű, az, hogy a modell előállítja a megfelelő megfigyelt adatokat a paramétereinek ismeretében.

3.4.1. 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 előrejelezhetőek (Sándor, 2019). A stacionaritást az ADF és KPSS egységgyök teszttel vizsgáltam Madaras és Lehel (2016) alapján.

Az **Augmented Dickey-Fuller (ADF) teszt** nullhipotézise (H_0) az, hogy az idősorban van egységgyökér, vagyis az idősor nem stacionárius míg az alternatív hipotézis (H_1) 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 stacionárius, mert nincs kimutatható egységgyökér.

A **Kwiatkowski-Phillips-Schmidt-Shin (KPSS) teszt** nullhipotézise az, hogy az idősor szigorúan stacionárius (az ADF nullhipotézisével ellentétben), tehát nincs egységgyökér. Ha el a p-érték nagyobb, mint 0.05, nem utasítjuk el a nullhipotézist, vagyis az idősor stacionárius.

3.4.2. Autoregresszív integrált mozgóátlag modellek

Ebben a fejezetben az AR/MA/ARMA/ARIMA modelleket és komponenseit mutatom be Sándor (2019) alapján.

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 $AR(p)$ modellt, a következőképpen írható le:

$$y_t = \alpha + \phi_1 y_{t-1} + \dots + \phi_p y_{t-p} + \varepsilon_t, \quad t = p + 1, \dots, n \quad (13)$$

Az $\phi_1, \dots, \phi_p$ ismeretlen paraméterek (autoregresszív együtthatók) és az ε_t a hibaváltozó, amit **fehérzajnak** feltételezünk, vagyis olyan folyamat, amelynek várható értéke 0, varianciája konstans és autokorrelációja 0, valamint a hibaváltozó kovarianciája az y_t 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:

$$y_t = \alpha + \theta_1 \varepsilon_t + \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q}, \quad t = q + 1, \dots, n \quad (14)$$

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

$$y_t = \alpha + \phi_1 y_{t-1} + \dots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q} \quad (15)$$

P az autoregressziós folyamat rendje, q a mozgóátlag folyamat rendje, y_t az idősorozat aktuális értéke, α a konstans érték, $\phi_1, \dots, \phi_p$ az autoregresszív együtthatók, ε_t az adott időpontbeli fehérzaj, $\theta_1, \dots, \theta_q$ a mozgóátlag együtthatók.

Az **ARIMA** (p, d, q) modellekben az I (integrated) azt jelenti, hogy az idősort differenciáljuk. A differenciálás célja az, hogy az idősorok stacionáriusak legyenek, ehhez az szükséges, hogy a szezonális mintázatokat eltávolítsuk. Ilyenkor az idősor különbségeit számoljuk ki az eggyel korábbi megfigyelés alapján:

$$\Delta Y_t = Y_t - Y_{t-1} \quad (16)$$

3.4.3. AR és MA komponensek meghatározása ACF és PACF segítségével

Egy idősor autokorrelációs függvénye (ACF) az autokorrelációk sorozata:

$$P_k = \text{corr}(y_t, y_{t-k}) = \frac{\gamma_k}{\gamma_0}, \quad k = 1, 2, \dots \quad (17)$$

, ahol $\gamma_k = E[(y_t - \mu)(y_{t-k} - \mu)]$ a k-ad rendű autokovariancia (Sándor, 2019).

Az autokovariancia meghatározza, hogy egy adott időpontban két változó között mennyi a változás. Ha az autokovariancia pozitív, akkor, ha az egyik változó nő, a másik is hajlamos növekedni, és fordítva. Az autokovariancia függ a két változó közötti eltolódás mértékétől. Például, ha az idősorokban a két változó közötti eltolódás növekszik, akkor az autokovariancia csökken, mivel a kapcsolat gyengül (Sándor, 2019).

Az ACF segít azonosítani a mozgóátlag (MA) folyamatot azáltal, hogy megmutatja, hány időegységnyi korreláció van az aktuális és az előző időpontok között, miközben figyelmen kívül hagyja a köztes időpontokat. Egy stacionárius idősor MA(q) folyamatnak tekinthető, ha annak ACF-je teljesíti azt a feltételt, hogy

$$p_k = 0, \text{ minden } k > q \text{ esetén, és } p_q \neq 0. \quad (18)$$

Az utóbbi tulajdonságot felhasználhatjuk arra, hogy megkülönböztessünk egy MA folyamatot más (AR, ARMA), folyamatoktól, és hogy meghatározzuk a mozgóátlag folyamat rendjét (q).

A parciális autokorrelációs függvény (PACF - Partial Autocorrelation Function) az autokorrelációs függvény egy olyan változata, amely kizárja azokat az autokorrelációkat, amelyek a közvetlen korrelációs hatásoktól függenek. A PACF azt mutatja meg, hogy egy adott időpontbeli megfigyelés és az előző időpontok közötti autokorreláció mennyire marad meg, ha kizárjuk az idősor többi részéből származó közvetlen korrelációt. A p_k autokorrelációk becslőfüggvénye (a k -ad rendű autokorrelációs együttható) a (19) képlettel írható le:

$$\gamma_k = \frac{\sum_{t=k+1}^n (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^n (y_t - \bar{y})^2} \quad (19)$$

ϕ_{kk} -val jelöljük y_t és y_{t-k} megfigyelések közötti parciális autokorrelációs együtthatót. Ezen együtthatók $k = 1, 2, \dots$, sorozata jelenti a parciális autokorrelációs függvényt.

Ha egy idősor PACF-e teljesíti azt, hogy $\phi_{pp} = 0$ és $\phi_{kk} = 0$; $k > p$; akkor AR(p) folyamatról beszélünk. Egy AR (1) folyamat esetében a p_k autokorrelációk folyamatosan (exponenciálisan) csökkennek k növekedésével, 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, ahol az autoregresszív folyamat p rendje megegyezik a feltételt teljesítő p értékkel (Sándor, 2019).

3.5. Neurális hálózatok

A mesterséges neuronhálók (ANN) lényege az emberi agyban található neurális hálózat működésének utánzása. Ezek olyan szoftveresen vagy hardveresen megvalósított, elosztott működésű rendszerek, amelyeket jellemzően sok, hasonló vagy azonos típusú, lokális adatfeldolgozást végző műveleti elemek, 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, ahol a hálózati csomópontok a neuronok (Brassai, 2019).

Egy neuronháló általában háromféle rétegből áll:

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

A neurális hálózatok működése két fő fázisból áll:

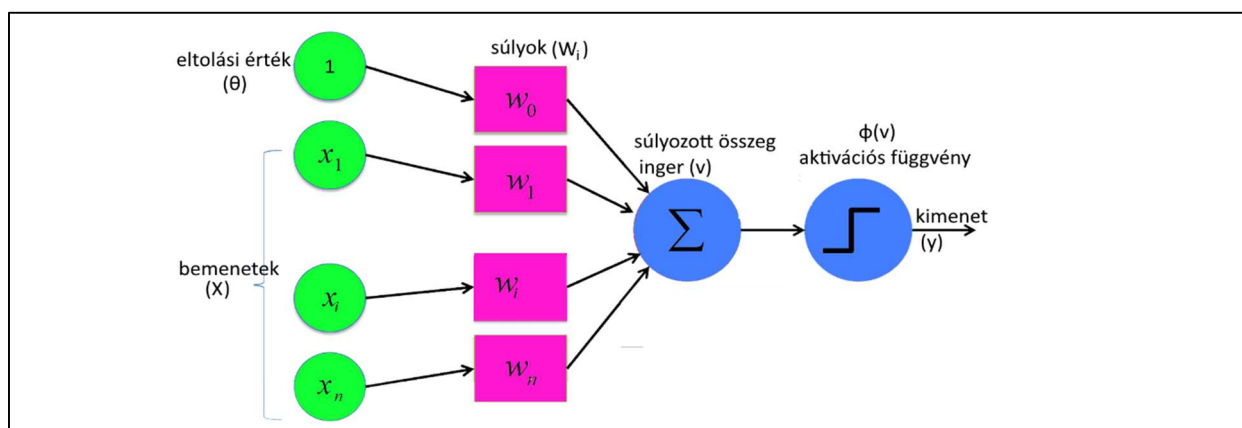
- **Tanulási fázis:** a hálózatban valamilyen módon eltároljuk a kívánt információfeldolgozó eljárást.
- **Előhívási fázis** (*recall*): a tárolt eljárás felhasználásával elvégezzük az információfeldolgozást.

A neuronhálók a tanulás során, az adatokból kapott visszacsatolás alapján módosítják a paramétereiket (eltolási értékek és súlyok). Általában, a tanulási fázis lassú, több iterációt, sok sikertelen tanulási szakaszt hordozhat. A tanítás általában korszakokra (*epoch*) van lebontva, és egy-egy korszak lefuttatása után a tanítási paramétereket újra lehet hangolni. Egy epoch során a háló teljesen végigmegegy a tanító adatokon. Az előhívási fázisban a pillanatnyi bemeneti értékek alapján meghatározzuk a neuronháló kimenetét (Brassai, 2019; Russell és Norvig, 2005).

3.5.1. Neuron, Perceptron, MLP

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) aktivációs függvényben kiértékeli. (Brassai, 2019; Russell és Norvig, 2005).

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 szeparálható) mintahalmazt. Ebből adódóan képes bemeneteket két osztályba sorolni, tehát egy lineáris osztályozó algoritmusnak tekintjük (Brassai, 2019; Russell és Norvig, 2005).



3. ábra. Egy perceptron felépítése.

Forrás: saját szerkesztés Brassai (2019), 80. oldal alapján

Egy perceptron felépítését a harmadik ábrán szemléltetem. Az egyszerű perceptron képtelen bonyolultabb feladatok megoldására, viszont a több perceptron rétegből álló hálók (multilayer perceptron - MLP) sokkal komplexebb feladatok elvégzésére is képesek, például szövegfelismerés, approximáció, regresszió és előrejelzés.

Ezek a hálózatok a ki- és bemeneti rétegen kívül egy vagy több rejtett réteget tartalmazhatnak. Az MLP egy előrecsatolt hálózat, tehát az információ egy irányba halad, nincsenek elemi visszacsatolások. Az egyes neuronok kimenete a vele összekapcsolt, következő rétegben levő neuron (egyik) bemenetét képezi. Egy perceptron kimenete a (20) képlettel írható le:

$$y = \varphi \left(\sum_{i=1}^N x_i w_i - \theta \right) \quad (20)$$

, ahol φ az aktivációs függvény, x_i az adott bemenet, w_i a bementhez tartozó súly, θ az eltolási érték (bias). Amikor minden szomszédos neuron kapcsolódik egymáshoz, azt teljesen összekötött topológiának nevezzük. Az ilyen összetett hálózatok képesek a mély tanulásra (*deep learning*), amely során összetettebb mintázatokat és hierarchikus jellemzőket tanulhatnak meg (Brassai, 2019; Russell és Norvig, 2005).

3.5.2. Az MLP modell optimális szerkezetének kiválasztása

Heaton (2005) (127 – 130. oldal) leírta, hogy az MLP neuronhálónál különböző problémák függvényében mennyi rejtett réteget érdemes használni, valamint kiindulópontokat fogalmazott meg arról, hogy az egyes rétegeknek hány darab neuront kellene tartalmazniuk. A bementi réteg annyi neuronból kell álljon, ahány (a kimenetet befolyásoló) független változónk van. A kimeneti rétegben pedig annyi neuron kell legyen, ahány kimenetet ad a neuronháló.

Heaton (2005) azt is leírja, hogy két rejtett réteg használata mellett bonyolult, nemlineáris összefüggéseket is képesek vagyunk reprezentálni, feltéve, hogy legalább az egyik rétegben nemlineáris aktivációs függvény van. Mivel pontosan nem határozható meg előre az, hogy az adott problémára mennyi rejtett réteg használata fog a legjobb eredményekkel szolgálni, kezdetben érdemes egy-két rejtett réteggel kezdeni, majd folyamatosan növelni azok számát, és letesztelni az eredményeket. A túl sok rejtett rétegek és neuronok száma könnyen vezethet a modell túltanításához. Általánosan elfogadott elvek szerint a rejtett rétegekben levő neuronok száma a ki- és bemeneti neuronok száma között kell legyen, vagy a bemeneti réteg neuronjainak számának kétharmad része plusz a kimeneti neuronok száma. A rejtett rétegek neuronjainak száma ne haladja meg a bemeneti neuronok számának kétszeresét.

3.5.3. ReLU aktivációs függvény

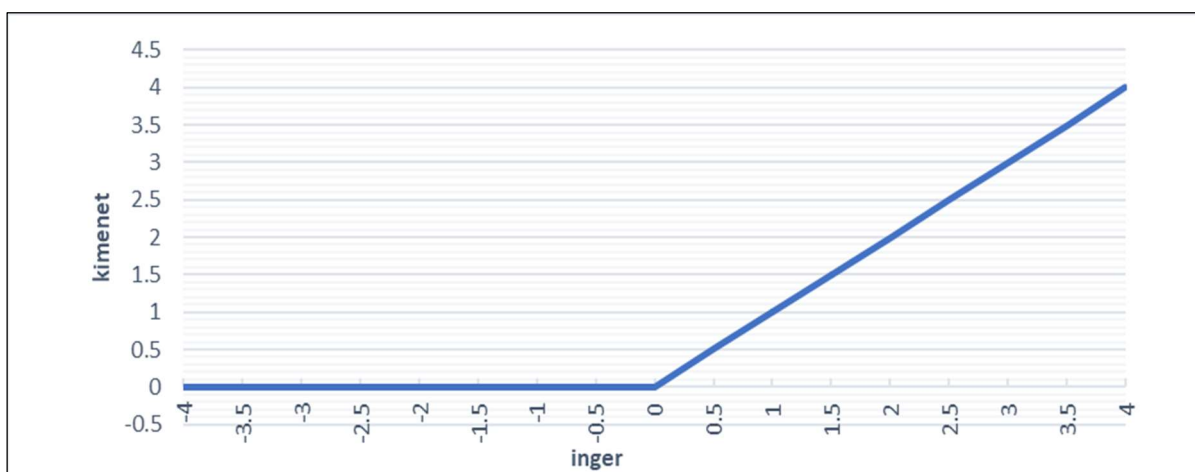
Az aktivációs függvények matematikai függvények, amelyek meghatározzák egy neurális hálózat rétegeinek kimenetét az adott bemeneti adatok alapján. Céljuk, hogy minden egyes neuronhoz egy aktiválási állapotot rendeljenek (aktív vagy inaktív). Ez az állapot jelzi, hogy a neuron milyen mértékben járuljon hozzá a réteg kimenetéhez. Általában ezen függvényeknek nemlineárisnak és/vagy differenciálhatónak kell lenniük, mivel számos optimalizálási algoritmus a hálók súlyait

gradiens segítségével hangolja. A gradiens a függvény legnagyobb növekedésének irányát megadó vektor, ami a függvény paramétereinek parciális deriváltjaiból áll. Megmutatja, hogy a hibafüggvény milyen irányba lejt, milyen irányba menjen tovább az optimalizációs algoritmus annak érdekében, hogy az egyre kisebb hiba felé haladjon (Brassai, 2019; Russell és Norvig, 2005).

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 a szakirodalomban ez a legelterjedtebb. Bamberger és társai (2023) kifejtették, hogy bizonyos jelenségek approximációja esetén milyen aktivációs függvényeket érdemes használni. A munkanélküliségi ráta függvénye a munkanélküliek számának és a munkaerő-állománynak a hányadosa, tehát a függvény nulladrendű homogén, vagyis skála-invariáns. Ebből adódóan Bamberger és társai (2023) (7. oldal, 2.5-ös tétel) szerint, az ilyen függvények esetében a ReLU aktivációs függvény megfelelő.

A ReLU (Rectified Linear Unit – rektifikált lineáris egység) az érkezett ingert adja vissza, ha nullánál nagyobb, különben nullát ad vissza (21), tehát a neuron inaktív lesz. A negyedik ábrán látható grafikon a ReLU aktivációs függvényt reprezentálja, egy példaként kiragadott bemeneti intervallumon (x tengely).

$$\varphi(x) = \max(0, x) \quad (21)$$



4. ábra. A ReLU (Rectified Linear Unit) aktivációs függvény grafikonja. (saját szerkesztés Excel-ben)

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 (Brassai, 2019).

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 (Russell & Norvig, 2005).

3.5.4. Ellenőrzött tanítási stratégia

A neuronhálók tanítása egy többváltozós optimalizációs folyamat, egy előre meghatározott célfüggvény (költségfüggvény, veszteségfüggvény, például 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 egy előre megadott tanító adatbázist adunk meg a neuronhálónak. A tanító adatbázis bemeneti és az ahhoz tartozó elvárt kimeneti adatok (*target*) párjaiból áll. A hiba-visszaterjesztéses (*backpropagation*) algoritmus célja az, hogy a modell a bemenetek és kimenetek közötti kapcsolatokat megtanulja úgy, hogy minimalizálja a modell kimenete és az elvárt kimenet közötti hibákat. Ilyenkor a kimeneti rétegtől visszafelé haladva, rétegről rétegre kiszámítja a költségfüggvény gradiensét, majd annak alapján, a kisebb hiba irányába való haladás érdekében frissíti a hálózat súlyait és eltolási értékeit (Brassai, 2019).

A modellek tanítása során gyakran előforduló probléma lehet a túltanítás (*overfitting*). Ilyenkor a tanítási ciklusok számának növelésével a hiba csökken a tanító adatok halmazán, viszont a tanító adatokon kívüli elvárt kimenetektől egyre jobban távolodnak a modell becslései. Ezt általában a nem megfelelő mennyiségű vagy minőségű tanítóadat, a túl sok tanítási ciklus, rossz modell szerkezet és paraméterek is okozhatják. A túltanítás ellen a teszt adatbázis is segíthet, amely a modell általánosító képességének ellenőrzésére szolgál (Brassai, 2019).

3.5.5. Optimalizációs algoritmusok

Ezek az algoritmusok felelősek azért, hogy a hiba-visszaterjesztés során számított gradiens információkat felhasználva frissítsék a háló paramétereit. Az optimalizációs algoritmusok döntenek a tanulási ráta, a momentum és más paraméterek beállításáról, valamint az egyes paraméterek módosításának mértékéről a gradiens alapján. Az algoritmusok célja általában az, hogy csökkentsék a modell hibáit azáltal, hogy megtalálják a lokális vagy globális minimumot a költségfüggvényben, és ezáltal optimalizálják a háló teljesítményét.

SGD (Stochastic Gradient Descent) az egyik legegyszerűbb optimalizációs algoritmus. Minden iterációban egy véletlenszerűen választott mintán számítja ki a gradienst, majd az adott tanulási ráta segítségével frissíti a paramétereket. A SGD könnyen alkalmazható és gyorsan konvergálhat, különösen nagy adatmennyiségek esetén, de az optimalizációs folyamat leragadhat a lokális minimumokban (Brassai, 2019).

Az ADAM (Adaptive Moment Estimation) a gradiens süllyedés algoritmus egy változata, amely adaptív módon szabályozza a tanulási ráta mértékét az egyes paraméterekhez. Ezáltal lehetővé teszi a gyorsabb tanulást és a hatékonyabb konvergenciát. Kevésbé érzékeny a tanulási ráta kiválasztására, és általában gyorsabban konvergál, mint az SGD (Brownlee, 2021).

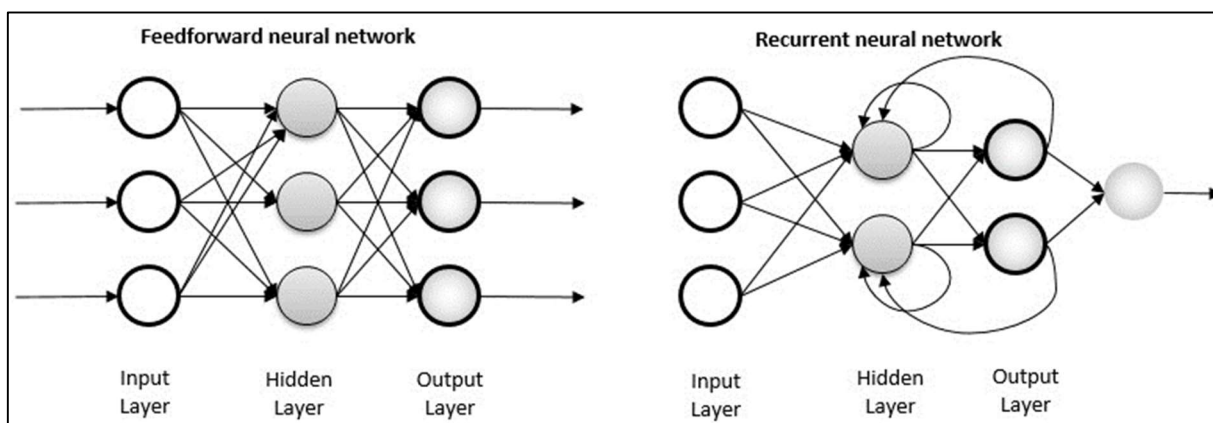
Az L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) egy hatékony, másodlagos optimalizációs algoritmus, amelyet leggyakrabban konvex és sima célfüggvények minimalizálására alkalmaznak. Ez a kvázi-Newton típusú módszer közelítőleg Newton-módszer alapján működik, de csak korlátozott memóriát használva, nem tárolja teljesen a Hess-mátrixot. Ennek köszönhetően alkalmas nagy méretű problémákhoz, és gyakran használják a gépi tanulásban és a mesterséges intelligencia területén modellek optimalizálására (Brownlee, 2021).

3.5.6. Visszacsatolt neuronhálók és az LSTM

Az LSTM (~hosszú rövidtávú memória) egy visszacsatolt neuronháló (RNN), amely az előrecsatolt hálózatokkal szemben visszajelző-kapcsolatokkal (hurokkal) is rendelkezik. Az előrecsatolt neuronhálók esetében a hálóelemek ki- és bemenetei függetlenek egymástól, és az adatok egy irányban haladnak, míg az előrecsatolt hálónál a feldolgozóegység kimenetét visszatápláljuk annak bemenetére is. Az ilyen szerkezetű neuronhálók memóriaként viselkednek, állapotokat tudunk iktatni a modellbe. Nem csak egy adatpontot (pl. egy számot) tud feldolgozni

egyszerre, hanem sorozatokat, szekvenciákat is képes értelmezni (pl. mondatok, beszéd, videók). Az efféle modellek célja általában a hosszútávú függőségek kezelése (Brownlee, 2021; Awan, 2022; Póda, 2021).

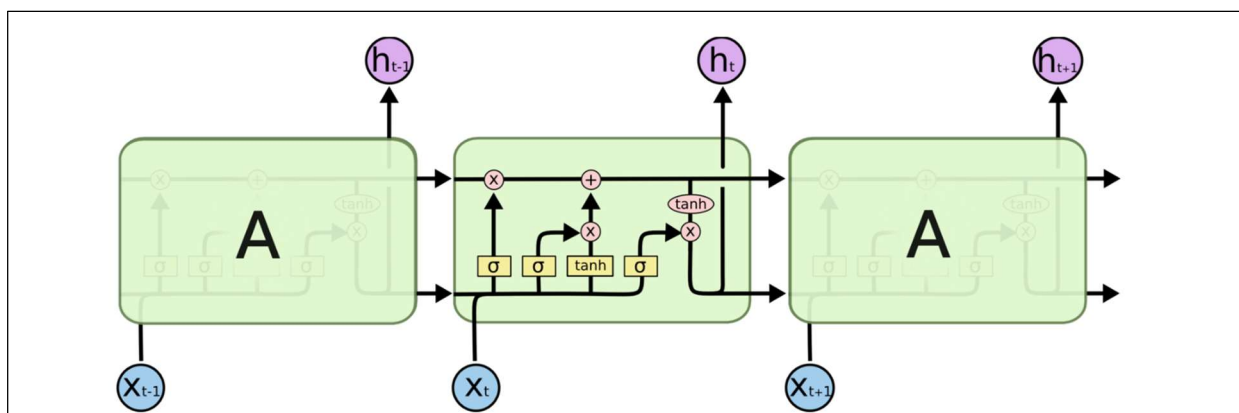
Az ötödik ábrán az előzőleg (MLP által) megismert előrecsatolt neuronháló struktúrát láthatjuk baloldalt, és jobboldalt egy visszacsatolt (rekurrens) szerkezet látható.



5. ábra. Egy előrecsatolt és egy visszacsatolt neuronháló szerkezete.

Forrás: (Pekel & Kara, 2017)

Az LSTM alapfelépítése hasonlít az RNN-hez, a rendszer itt is egymás után ismétlődő modulokból épül fel. Az LSTM modellek általában több rétegből állnak. Egy réteg több, egymáshoz kapcsolt állapotcellából épül fel. A celláknak több bemenete és kimenete van, valamint kapukat tartalmaznak. A felejtési kapu eldönti, hogy mennyire hagyjuk figyelmen kívül az előző időlépésen tárolt információkat, a bemeneti kapu meghatározza, hogy mennyi új információt engedünk be a rendszerbe. A kimeneti kapu dönti el, hogy mennyi információt adunk ki a hálózathoz (Póda, 2021).



6. ábra. Egy LSTM hálózat szerkezete.

Forrás: (Póda, 2021)

4. A kutatási gyakorlati része

Ebben a fejezetben a kutatás gyakorlati részét tárgyalom. Először a kutatáshoz készített programom működését ismertetem, majd a benne végzett számítások és tesztek eredményeit.

4.1. A Django webalkalmazás működése

Django egy kiváló Python-alapú webes keretrendszer, amely lehetővé teszi a gyors és biztonságos weboldalak fejlesztését. Az MVC (Model-View-Controller) szemléletet követi. Kontroller fájlok helyett a `views.py` fájlban található függvények segítségével dolgozzuk fel az adatokat és generáljuk a nézeteket, és küldjük vissza a választ a kliensnek. Az URL címeket az `urls.py` fájl tartalmazza, amelyek alapján a felhasználó által megadott címek alapján hívódnak meg a megfelelő nézetek. A modellek az alkalmazás adatstruktúráját és az adatbáziskezelést definiálják. A sablonok (template-ek) segítségével határozzuk meg az oldalak elrendezését és tartalmát, amelyek dinamikusan generálhatók és jeleníthetők meg az adatok alapján a weboldalon (Dhandi, 2024).

A Django széles körben használt és erőteljes eszköztárát a Pythonban alkalmazott számítások vizualizációjához is használják. Menon és társai (2021) például egy kórház számára készítettek adatvizualizációs eszközt. Punasya és társai (2021) eladási adatok vizualizálására készítettek Django webalkalmazást.

A letöltött és Excel táblázatban rendszerezett idősorokat a Pandas nevű Python modul segítségével olvastam be DataFrame adatszerkezetekbe, két részre osztva: tanítóhalmazra, amelyet megvizsgálók az előrejelzési modellek készítése előtt, és ami egyúttal az előrejelzési modellek illeszkedését, „tanulását” biztosítja majd. A második szakaszt arra használtam fel, hogy a „betanított” modellekkel ugyanerre az időtávra készített előrejelzések pontosságát meghatározzam.

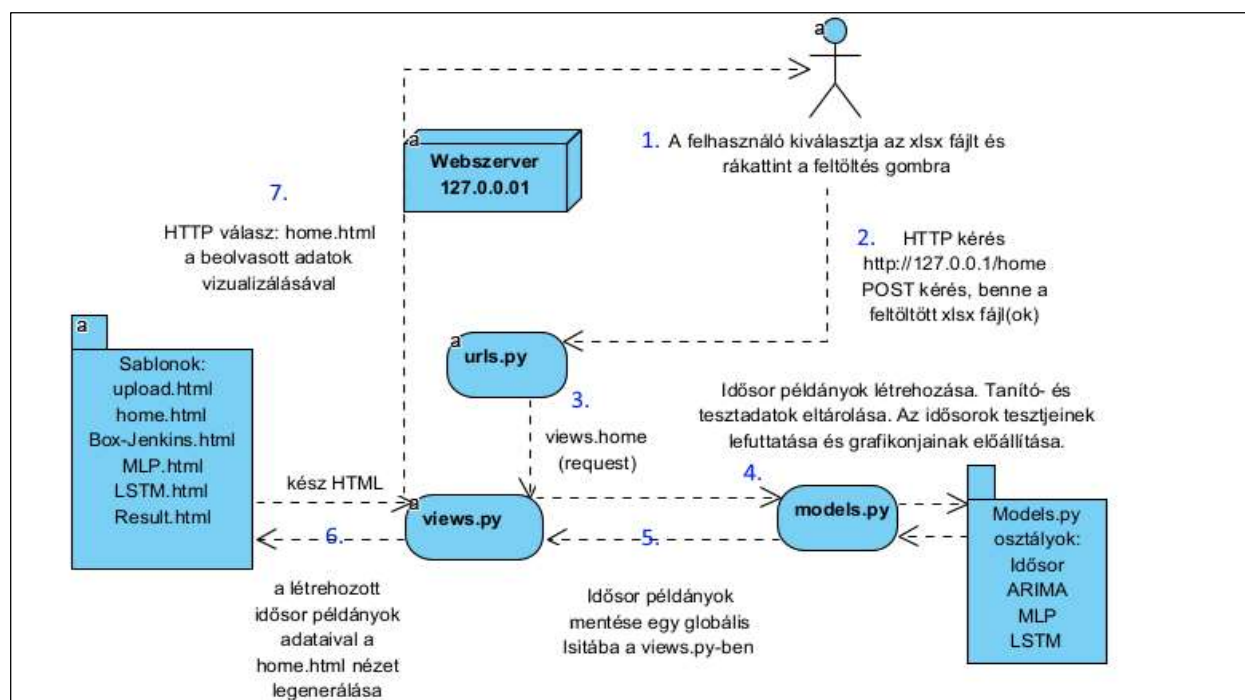
Az adatok beolvasása egy egyszerű HTML űrlap segítségével történik, ez a Django webalkalmazás kezdőlapja (`upload.html`). Az idősorok Excel fájlban kell legyenek, elkülönítve a tanító- és a tesztadatokat. Egyetlen fájl esetén két külön munkalapra kell helyezni az adatokat. Egyszerre több idősort is képes beolvasni a program, feltéve, hogy azok egymás mellett elhelyezett, egyenlő számú sorral rendelkező oszlopok. Az első oszlopot a megfigyelések független változójaként kezeli 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 fejlécként kezeli, tehát az idősorok megnevezéseit innen olvassa be. Az program számára megfelelően formázott táblázat mintája az első táblázatban látható.

1. táblázat. Példa az idősorok előkészítésére Excel-ben.

Időpont	Első idősor	Második idősor	i-edik idősor
1. dátum	1. megfigyelés	1. megfigyelés	1. megfigyelés
2. dátum	2. megfigyelés	2. megfigyelés	2. megfigyelés
t-edik dátum	t. megfigyelés	t. megfigyelés	t-edik megfigyelés

A Python programom az objektumorientált paradigmát követi, így minden beolvasott idősor egy osztály példánya lesz (Idősor nevű osztály a models.py fájlban), amelynek különböző adattagjai (tanító- és tesztadatok listája, mutatók, ARIMA, MLP, LSTM modellek példányai, diagramok) és függvényei vannak, így a Views.py könnyen át tudja adni a HTML sablonok számára az adatokat, amikor létrehozza a felhasználó számára az oldalt. A hatodik ábrán a webalkalmazás felépítése látható, amelyet az idősorokat tartalmazó Excel állomány feltöltésének folyamatán keresztül szemléltetek.



7. ábra. Az Excel fájl feltöltésének folyamata a webalkalmazásba.

Forrás: saját szerkesztés Visual Paradigm szoftverben

Felhasználva a Django adatvizualizációban használt lehetőségeit, 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. A program képes Excel fájlokból idősorokat beolvasni, amelyek tanító- és tesztadatokra vannak lebontva.

A sikeres beolvasást követően a program egy HTML táblázatban megmutatja a beolvasott adatokat, valamint azokat egy közös grafikonon is ábrázolja (10. ábra). Természetesen egyetlen idősort is fel lehet tölteni. Az idősorokról hisztogramokat készít és lefuttatja rajtuk a Kolmogorov–Smirnov tesztet a normális eloszlásra, amelyek eredményét egy táblázatban foglalja össze. Erről az oldalról van lehetőség előrejelzési módszert választani, valamelyik gombra kattintva: Box-Jenkins eljárás, LSTM vagy MLP neuronhálók.

A Box-Jenkins eljárást támogató oldalon megjelennek az automatikusan lefuttatott stacionaritás tesztek eredményei, valamint az idősorok ACF és PACF ábrái, amelyek segítenek az ARIMA (p, d, q) paraméterek kezdeti behatárolásában. Az oldalon lejjebb görgetve, megjelennek a beolvasott idősorok számára, külön-külön az ARIMA modellek paramétereit bekérő HTML űrlap elemek (9. ábra). Az „ARIMA modellek készítése” gomb előtt megadhatjuk, hogy a modellek mennyi előrejelzést adjanak a tanítóadatokon túlmenően.

A feldolgozást követően modellenként, táblázatokat kapunk arról, hogy a tesztadatok elvárt kimeneteihez képest mit becsült a modell, valamint a köztük levő eltéréseket. A táblázatok alatt grafikonon is láthatjuk a valódi és becsült adatokat, valamint a reziduumok hisztogramját, pontdiagramját és az autokorrelációs függvényéből készített ábrát. Végül egy összefoglaló táblázatban megjelennek a modellek hibastatisztikái (MSE, RRMSE, MAPE) és a determinisztikus együttthatók. Végül a program egy összesítő grafikonon ábrázolja a modellek becsléseit.

Innen lehetőség van természetesen visszanavigálni a kezdőlapra, és ki lehet próbálni az MLP vagy LSTM módszereket is. A neuronhálók esetében is szinte teljesen ugyanez a logika, mint az ARIMA modellek esetében, csak más paramétereket tudunk megadni a modelleknek. A neuronhálók paraméterezését szolgáló űrlapon minden paraméter mellé tettem egy információs ikont, amely egy leírást jelenít meg addig, amíg a kurzor az *i* ikon felett van. Ezek segítenek a felhasználónak a paraméterek céljának megértésében.

Az neuronhálós modelleknél lehetőség van megadni az idősorok normalizációjának a módját, a tanító- és teszt adatbázis elkészítése során a késleltetett értékeket, a hálók szerkezetét (rejtett rétegek számát, feldolgozó egységek számát, aktivációs függvényt), az epoch-ok számát, az optimalizációs algoritmust, és a kért előrejelzések számát.

Statistikai elemzés és előrejelzés

Kérem, tölts fel az előkészített .XLS / .XLSX fájlt
utolsó hiba: Nem lett adatbázis fájl feltöltve!

Válasszon fájlt...

Nincs kiválasztott fájl.

Mi a munkalap neve?

data

A grafikonon milyen sűrűséggel legyen feliratozva az X tengely (hónapokban)?

12

Valós (mért) adatok feltöltése az előrejelzési modell készítéséhez

Ugyanaz a fájl ☒

Kérem, tölts fel az előkészített .XLS / .XLSX fájlt, amely a becsült időszakkal megegyező elemszámú adatsort tartalmazza!

Válasszon fájlt...

Nincs kiválasztott fájl.

Mi a munkalap neve?

teszt

Feltöltés

Adatok feltöltve, kérem válasszon egy opciót az előrejelzéshez!

Box-Jenkins eljárás

LSTM neuronháló

MLP Neuronháló

Beolvasott adatok

dátum	Kovácsna	Hargita	Maros
2010 január	12.2	10.9	8.3
2010 február	12.5	11.4	8.4
2010 március	12.3	11.2	8.5
2010 április	11.9	10.7	8.2
2010 május	11.1	9.6	7.9
2010 június	10.9	9.3	7.8
2010 július	11.4	8.6	7.9
2010 augusztus	11.2	8.7	8.3
2010 szeptember	10.7	8.4	8.2

Adatok közös grafikonon ábrázolva

8. ábra. A fájlfeltöltés oldala, és azt követő elemző oldal.

ARIMA modellek beállítása teszteléshez és előrejelzéshez

A fenti ACF és PACF tesztek alapján itt lehet próbálkozni különböző idősor modellekkel. A gomb megnyomását követően egy szöveges (txt) fájl kerül letöltésre, amely minden adatsort esetén a modell mutatóit és előrejelzéseit tartalmazza.

Kovácsna

p:

d:

q:

AR(p) MA(q) ARMA(p, q) ARIMA(p, d, q)

MLP modell beállítása teszteléshez és előrejelzéshez

Az alábbi űrlapon adja meg az MLP modell paramétereit, majd kattintson az előrejelzések készítése gombra. A modell az összes beolvasott tesztadat érték száma szerint fogja előrejelzéseit elkészíteni.

Kovácsna

Rejtett rétegek neuronjainak száma (pl. 12,12,12):

Max iterációk száma:

Random állapot kezdőérték:

Random állapot utolsó érték:

Cél RRMSE %-ban:

Normalizálás: ☒ Elvárt kimenetek (y_train és y_test) normalizálása, majd visszaalakítás ☐ Standardizáció

(általában nem ajánlott)

Standardizáció

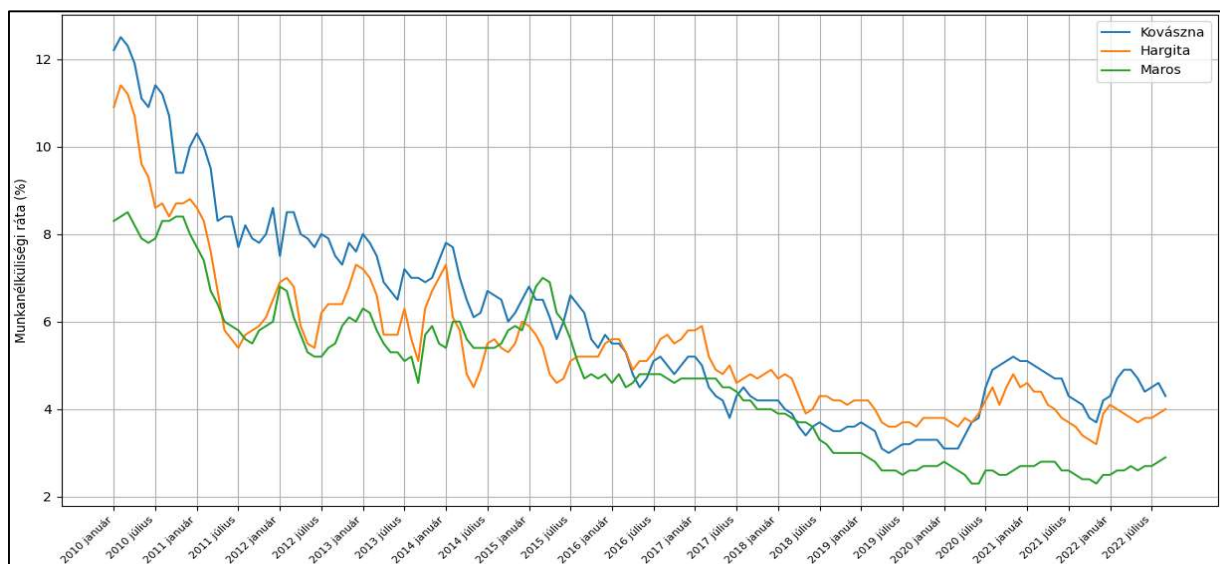
Aktivációs függvény:

Optimalizációs Algoritmus:

9. ábra. Az ARIMA modellek és az MLP modellek paraméterezését szolgáló űrlapok.

4.2. Az idősorok elemzése

Az előrejelzések készítése előtt fontos megvizsgálni az idősorok alapvető statisztikai tulajdonságait, mivel iránymutató szerepe lesz a prediktív modellek készítése során.



10. ábra. A Székelyföldi megyék 2010 január – 2022 július közötti munkanélküliségi rátáinak grafikonja.

Forrás: saját szerkesztés Python-ben, az INSSE adatai alapján)

Az idősorok grafikonján (10. ábra) látszik, hogy Maros megyében szinte végig a legalacsonyabb a munkanélküliségi ráta, amely többek között a megye ipari fejlettségének, ezáltal a több munkahelynek is tulajdonítható. Összességében 2020 tavaszáig csökkenő trend volt megfigyelhető. Sok periódusban megfigyelhető, hogy télen magasabb volt a mutató, mint a többi évszakban, ez például a szezonális munkákhoz (pl. mezőgazdaság és építkezések) is köthető, viszont a pontos okok tisztázása egy külön kutatást igényel és nem tartozik ezen dolgozat témájához.

A második táblázatban összefoglaltam a főbb statisztikai mutatókat. Kiszámítottam az átlagokat évszakonként összesítve is, hogy megnézzem, alátámasztja-e azt, hogy télen általában magasabb a munkanélküliségi ráta. A táblázatban a legalacsonyabb és legmagasabb megfigyelésekhez tartozó időpontokat is feltüntettem.

2. táblázat. A székelyföldi megyék munkanélküliségi rátáinak statisztikai adatai.

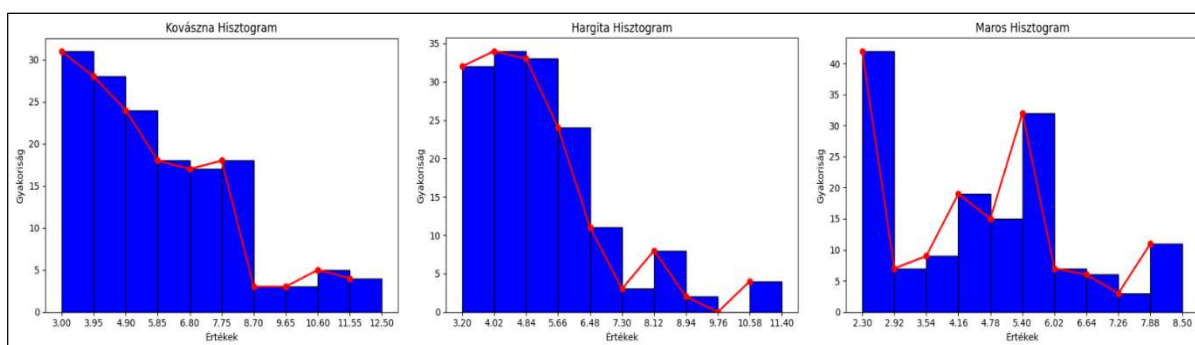
mutató	Kovászna megye		Hargita megye		Maros megye	
Átlag	5.98%		5.43%		4.69%	
Szórás	2.26%		1.65%		1.73%	
Variancia	5.13%		2.71%		3.01%	
Medián	5.20%		5.20%		4.70%	
Minimum	3.0%	2019 május	3.2%	2021 november	2.3%	2020 május
Maximum	12.5%	2010 február	11.4%	2010 február	8.5%	2010 március
Téli átlag	6.29%		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%	

Forrás: saját szerkesztés az INSSE adatai alapján

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 kialakult pénzügyi és gazdasági világválság hatása volt, amely kissé késve, de elérte Romániát is. 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).

Az 1. 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, torzítják a középértékeket a kiugró értékek (Lázár, 2022). Ez hatással lehet a neurális hálózatok tanulására és általánosító képességére is. A programom az idősorok hisztogramjait is automatikusan elkészíti, valamint lefuttatja rajtuk Kolmogorov-Smirnov tesztet a normál eloszlásra, a scipy.stats Python modul kstest függvényének segítségével. Az idősorokról készített hisztogramok a 11. ábrán láthatóak.



11. ábra. Az idősorok hisztogramjai.
Forrás: saját szerkesztés Python-ben, az INSSE adatai alapján

Látszik, hogy Kovászna és Hargita megye idősorainak eloszlásának az alakja jobbra elnyúló, tehát a kiugró értékek mind az átlagnál jóval magasabbak. Azonban a normalitás vizsgálatához Lázár (2022) alapján a Kolmogorov-Smirnov tesztet is elvégeztem.

3. táblázat. az idősorokon elvégzett Kolmogorov-Smirnov teszt eredményei.

Idősor	p-érték	Átlag	Ferdeség	Hegyesség
Kovászna	0.006	5.98	0.90	0.28
Hargita	0.004	5.43	1.45	2.36
Maros	0.003	4.69	0.35	-0.73

A harmadik táblázatban találhatóak a normalitást ellenőrző tesztek p-értékei a három idősor esetében, amelyek szerint a túl kicsi p-értékek miatt túl sok bizonyíték van arra, hogy az adatok nem követik a normális eloszlást. Ezt figyelembe vettem a neurális hálók tanítása során, és normalizáltam az adatokat.

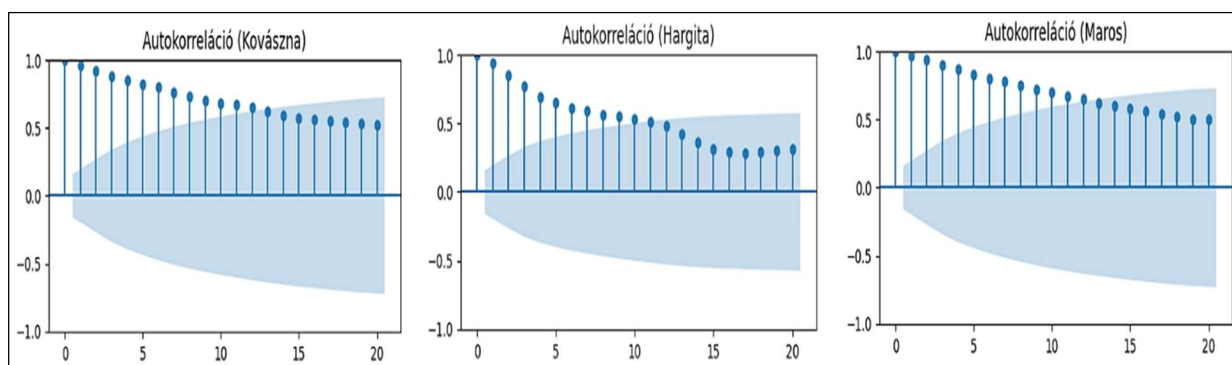
Ezt követően megvizsgáltam az idősorok stacionaritását az egységgyöker tesztekkel (Madaras & Lehel, 2016) alapján. Ehhez a Python *statsmodels.tsa.stattools* csomagjából az *adfuller* és *kpss* függvényeket használtam fel. Az eredmények a 4. táblázatban találhatóak.

4. táblázat. az ADF és KPSS tesztek eredményei. *(Saját szerkesztés az INSSE adatai alapján)*

Idősor	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	-2.88	1.48	0.01	0.46
Maros	-2.37	0.15	-2.88	1.65	0.01	0.46

Mindegyik megye idősorának 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 az ARIMA modellek jöhetnek szóba, ahol a d jelöli a differenciálást (*first differenced*: $d = 1$).

Az $MA(q)$, vagyis a mozgóátlag komponens rendjének meghatározása érdekében elkészítettem az idősorok autokorrelációs függvényeit a `statsmodels.tsa.stattools.acf` függvénnyel, és ábrázoltam azokat a `pyplot` modul segítségével. Az vizsgált idősorok autokorrelációs függvényei a 12. ábrán láthatóak.

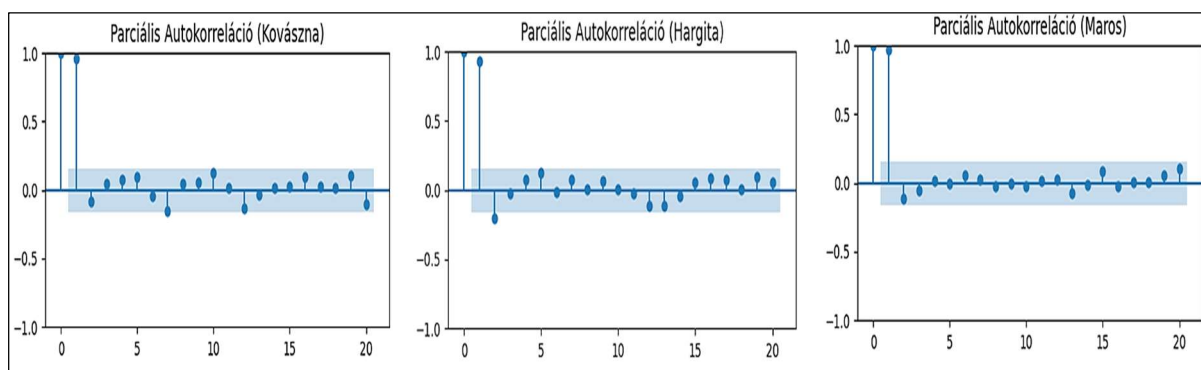


12. ábra. Az idősorok autokorrelációs függvényei. (Saját szerkesztés Python-ben)

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. Ha egy autokorrelációs együttható a kék színnel befestett területen belül helyezkedik el, akkor azt mondhatjuk, hogy nincs jelentős autokorreláció az adott késleltetett értékben (Drelczuk, 2020).

Ha az adott idősor egy $MA(q)$, vagyis mozgóátlag folyamat lenne, akkor az első néhány k lépés után a p_k autokorrelációk hirtelen csökkennének (Sándor, 2019), viszont a 12. ábrán levő ACF-ek nem ezt mutatják, hanem lineáris, fokozatos csökkenést. Ebből adódóan az autokorrelációs függvénnyel szerint az idősorok nem $MA(q)$ folyamatok. Azonban ez önmagában még jelenti azt, hogy az $ARMA(p, q)$, vagy $ARIMA(p, d, q)$ modellek ne jöhetnének szóba.

Az idősorok $AR(p)$ autoregresszív komponenseinek meghatározásához szükség van a parciális autokorrelációs függvényekre, amelyek a 13. ábrán láthatóak.



13. ábra. A parciális autokorrelációs függvények diagramjai. (Saját szerkesztés Python-ben)

Mindhárom megye esetében az látszik a parciális autokorrelációs függvényen, hogy az első kettő lépésben az autokorrelációs együtthatók egyhez közeli értékek, míg az összes többi lépésben elhanyagolhatóan kicsik. Ebből adódóan $AR(2)$ modellel érdemes kezdeni, de szóba jöhetnek az $ARMA$ vagy $ARIMA$ típusú modellek is, mivel a PACF önmagában nem tudja meghatározni vagy kizárni az $ARMA(p, q)$, $ARIMA(p, d, q)$ folyamatokat, így többféle modellt is leteszteltem (Sándor, 2019).

4.3. Az $ARIMA$ modellek implementációja

A 12. és 13. ábrán levő grafikonok azt szemléltetik, hogy valószínűleg mindhárom megye esetében az $AR(2)$ modellel érdemes próbálkozni elsőként a becslések készítéséhez, 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.

A Python programban a `statsmodels.tsa.arima.model.ARIMA` ingyenesen telepíthető csomag segítségével illesztettem a modelleket és készítettem el 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, míg az utóbbi a modell előrejelzési pontosságának meghatározására szolgál.

Az AIC értékeket a Python-ben létrehozott statsmodels.tsa.arima.model.ARIMA osztállyal készített modellek *model_fit.aic* attribútumából kértem le, a tanítóadatokra való illesztést követően, amelyek az ötödik táblázatban vannak összefoglalva modellenként.

5. táblázat. A letesztelt ARIMA modellek AIC értékei.

Idősor	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

A kutatás során több modell is ki lett próbálva: ARMA ($p = 1 \dots 3$, $q = 1 \dots 3$), ARIMA ($p = 1 \dots 3$, $i = 1 \dots 3$, $q = 1 \dots 3$), de ezek közül csak a legjobb ötöt tüntettem fel. Ebben a táblázaton az látható, hogy az AIC mutató szerint Hargita és Kovászna megye esetében az ARIMA (2, 1, 2), Maros megyénél az ARIMA (1, 1, 0) modell jelentősen jobban illeszkedett a tanító idősorra, mint az AR (2) modell, ezért ezeket a modelleket használtam a tesztadatok előrejelzésére és ezek teljesítményét hasonlítottam össze az MLP és LSTM regressziós modellekkel.

A modellek illesztését követően megnéztem, hogy hogyan általánosítanak, vagyis mennyire pontosan becsülik meg a tesztalmaz értékeit. Ehhez Brownlee (2023) alapján, az előre-lépő (*walk forward*) validációt alkalmaztam. Ilyenkor a modell az addigi ismeretei alapján egyszerre egy lépést jelez előre, majd az előrejelzett értékkel frissíti a tanítóadatoknak használt listát, amivel újra illeszti önmagát. Ez azért jobb megoldás, mint egy lépésben lekérni a 12 db előrejelzést a predict() függvénnyel, mivel abban az esetben csak egy egyenes vonalat kapnánk. Az ARIMA előrejelzéseket előállító Python alprogram a 14. ábrán látható.

```
def predict(self, test_data, adatok):
    history = [x for x in adatok]
    predictions = list()
    for t in range(len(test_data)):
        model = sm.tsa.ARIMA(history, order=(self.p, self.d, self.q))
        model_fit = model.fit()
        output = model_fit.forecast()
        predictions.append(output[0])
        obs = test_data[t]
        history.append(obs)

    self.predictions = predictions
```

14. ábra. előrejelzés ARIMA modellel, előre-lépő validációval Python-ben.

Forrás: saját szerkesztés, Brownlee (2023) alapján

Az ARIMA modellek becslései a és reziduuma a teszt időszakra vonatkozóan a 9. és a 10. összefoglaló táblázatokban találhatóak, a z MLP és LSTM modellek adataival együtt.

Az 6. táblázatban összefoglaltam a legjobban illeszkedő ARIMA modellek általánosító képességét a tesztalmazra tekintve, a különböző becslési hibákat mérő mutatók szerint:

6. táblázat. az ARIMA modellek teszt időszakra adott becsléseinek pontossága

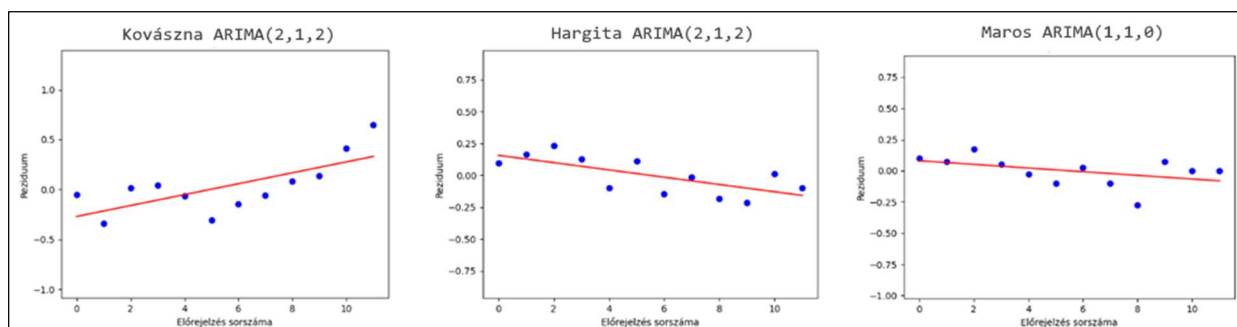
Idősor	Modell	MSE	RRMSE	MAPE	R ²
Hargita	ARIMA (2, 1, 2)	0.00198	0.00343	3.04	0.72
Kovácsna	ARIMA (2, 1, 2)	0.00713	0.00655	4.69	0.48
Maros	ARIMA (1, 1, 0)	0.00126	0.00383	2.84	0.71

4.3.1. A Reziduumok vizsgálata

A reziduumok a regressziós modellek becsléseinek és a valódi adatok közötti különbségeket jelentik (Lázár, 2022). A reziduumok segítségével megvizsgáltam, hogy az egy éves tesztelési időszakra tekintve mennyire megbízhatóak az ARIMA modellek becslései, Sándor (2019) alapján.

A becslések megbízhatóságának szempontjából fontos, hogy a reziduumok fehérzajként viselkedjenek, vagyis a hibáknak ne legyen szisztematikus mintázata és ne korreláljanak egymással. Továbbá a reziduumoknak lehetőleg homoszkedasztikusnak kell lenniük. Ez azt jelenti, hogy a hibák varianciája az időben állandó. Ellenkező esetben heteroszkedaszticitás van jelen, ami azt jelenti, hogy a modell becslései nem megbízhatóak (Sándor, 2019).

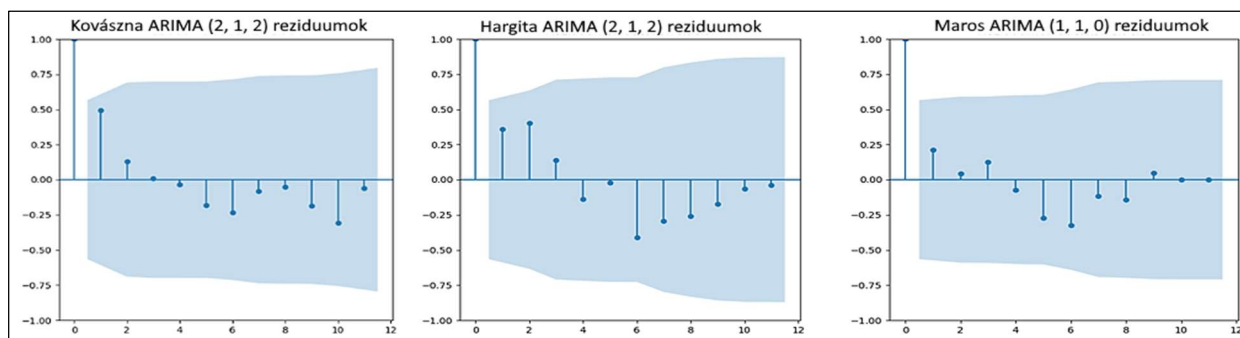
Az ARIMA modellek becsléseinek reziduumainak pontdiagramjai a 15. ábrán találhatóak, amelyeken az látszik, hogy az reziduumok inkább az egyenes mentén helyezkednek el, és nem tölcészerűen (Lázár, 2022). Ez azért jó, mert a modell feltételezhetően nem mutat heteroszkedaszticitást.



15. ábra. az ARIMA modellek reziduumainak pontdiagramjai.

Forrás: saját szerkesztés Python-ben

A reziduumok autokorrelációs függvényei a 16. ábrán láthatóak. Egyik modell esetében sincs autokorreláció a hibák között, mivel mindegyik érték a 95%-os intervallumon belül van, tehát a modellek ebből a szempontból is megfelelőek.



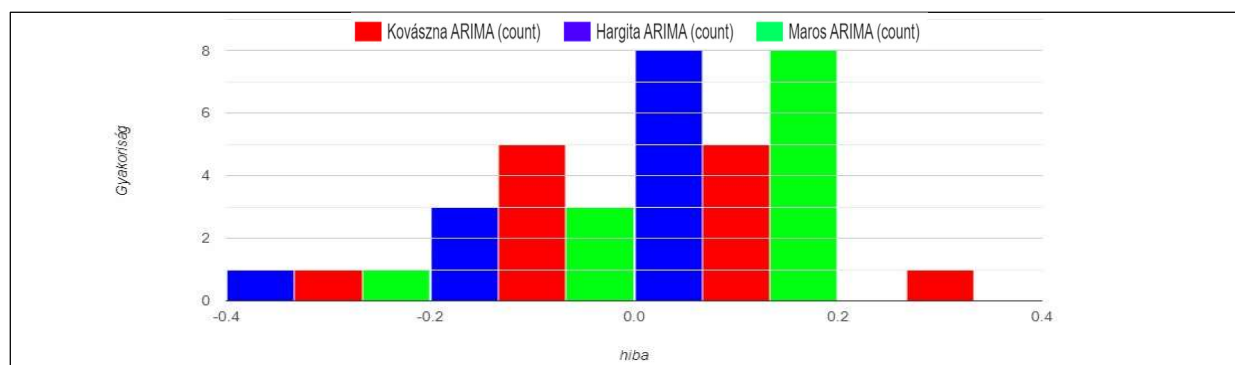
16. ábra. az ARIMA modellek általi becslések reziduumaik autokorrelációs függvényei. (saját szerkesztés Python-ben)

Ezen kívül még az is fontos, hogy a reziduumok normális eloszlást kövessenek, ezért elvégeztem a Kolmogorov-Smirnov tesztet mindegyik ARIMA modell hibáira Lázár (2022) alapján. A normál eloszlás alapján azt feltételezzük, a hibák átlaga megközelítőleg nulla.

7. táblázat. az ARIMA modellek reziduumaik elvégzett Kolmogorov-Smirnov tesztek eredményei.

Megye	Modell	p-érték	Átlag	Ferdeség	Hegyesség
Kovászna	ARIMA (2, 1, 2)	0.80	0.03	1.00	1.30
Hargita	ARIMA (2, 1, 2)	0.79	0.00	-1.39	0.07
Maros	ARIMA (1, 1, 0)	0.85	0.00	-1.04	1.81

A 7. táblázatban az látható, hogy mindegyik modell esetében a p-érték jóval nagyobb, mint 0.05, ezért azt mondhatjuk, hogy a hibák eloszlása nem különbözik szignifikánsan a normális eloszlástól. A reziduumokat egy közös hisztogramon is ábrázoltam, amely a 17. ábrán látható, amely megerősíti azt, hogy a reziduumok követik a normális eloszlást.



17. ábra. Az ARIMA modellek reziduumaik eloszlásai.

Forrás: www.statskingdom.com

4.4. A neuronhálók implementációja

Ebben a részben ismertetem, hogy milyen módszerrel valósítottam meg a neurális hálózatokat, hogyan tanítottam be őket, és hogyan jeleztem előre belük.

4.4.1. A tanító és tesztelő adatbázis előkészítése

Egy idősor megfigyeléseit át lehet alakítani felügyelt tanulási problémaként. Ezt úgy tehetjük meg, hogy az előző időpontot vagy időpontokat bemeneti változóként, és a következő időpontot 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}), a szakirodalomban sliding window néven ismert módszerrel. Ilyenkor az idősorból egy választott ablakméret alapján hozzuk létre a bemenetek és kimenetek listáját. Egy kimeneti értéket a választott ablakméretnek megfelelő számú korábbi megfigyelésből állítunk elő, vagyis egy kimenetet n darab korábbi értékből következik.

```
def split_sequence(sequence, n_steps):
    x, y = list(), list()
    for i in range(len(sequence)):
        end_ix = i + n_steps
        if end_ix > len(sequence)-1:
            break
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        x.append(seq_x)
        y.append(seq_y)
    return np.array(x), np.array(y)
```

18. ábra. Az idősorokból tanító párokat előállító Python függvény.
Forrás: Brownlee, (2018), 47. oldal

A bemeneti minták hosszának 3-at választottam, mert a szakirodalomban ez a legelterjedtebb, például (Brownlee, 2018) is ezt az ablakméretet használta. A 19. ábrán reprezentálom, hogy Maros megye idősorának első 5 megfigyeléséből hogyan fog kinézni az első kettő tanító pár.

```
time_series = [8.3, 8.4, 8.5, 8.2, 7.9]
lag = 3
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
```

19. ábra. Példa a mozgóablak módszerrel készített tanító párokra. (saját szerkesztés)

Ez a gyakorlatban azt jelenti, hogy a január, február és márciusi adatokból következik az áprilisi érték. A májusi megfigyelés pedig a februári, márciusi és áprilisi értékektől függ és így tovább.

4.4.2. Az adatok standardizálása

A neuronhálós modellek érzékenyek az adatok eloszlására és azok értéktartománya. A kiugró értékek lelassíthatják vagy rossz irányba vihetik el a tanulást, mivel a magasabb értékek dominálhatják a célfüggvényt (Brassai, 2019). Emiatt ajánlott az adatok skálázása, mielőtt azokat egy LSTM vagy MLP hálózatba tápláljuk. Az adatok skálázása biztosítja, hogy minden adatnak ugyanaz legyen az értéktartománya, ami hozzájárulhat a neuronháló hatékonyabb tanításához azáltal, hogy segít kiküszöbölni a lokális minimumokba való ragadást (Brassai, 2019). Chen és társai (2022) és kutatásukban megállapították, hogy az LSTM prediktív modellek sokkal jobban teljesítenek akkor, ha a bemeneti adatok normál eloszlást követnek.

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 (x_i) kivonjuk az átlagot ($\bar{x}$), majd elosztjuk a szórással (σ) (Brassai, 2019).

$$x' = \frac{x_i - \bar{x}}{\sigma} \quad (22)$$

A programomban a `sklearn.preprocessing` osztály segítségével többféle skálázásra is lehetőséget biztosítok az MLP és LSTM neurális hálók készítése során, Brownlee (2019) alapján: egy leghördülő listában választhatunk standardizáció, min-max normalizáció, robosztus skálázás és logaritmizálás között. Az is fontos kérdés, hogy csupán a bemeneteket (x), vagy az elvárt kimeneteket (y) is transzformáljuk-e? Mindkét megközelítés elfogadható, különböző idősorok és modellek esetében. Az elvárt kimenetek skálázása esetében viszont a konzisztencia szempontjából ugyanazzal a skálával kell dolgozni, amit a bemeneteknél használtunk.

A legjobb modellek kiválasztása során kipróbáltam mindegyik transzformációt a felsoroltak közül, és mindegyik megye esetében az idősorok standardizálása után teljesítettek a legjobban az MLP és az LSTM modellek.

A standardizálást a Scikit-learn modulban található `sklearn.preprocessing.StandardScaler` osztály segítségével végeztem el. A normalizálást végző függvény kódja a 20. ábrán látható:

```

def normalize(self, scalerMode: str = "-", normOut: bool = False):
    scaler = None
    if scalerMode == "robust":
        scaler = RobustScaler()
    if scalerMode == "minmax":
        scaler = MinMaxScaler()
    if scalerMode == "standard":
        scaler = StandardScaler()

    if scalerMode != "-":
        if scalerMode == "log":
            self.X_train = np.log(self.X_train)
            self.X_test = np.log(self.X_test)

            if normOut:
                self.y_test = np.log(self.y_test)
                self.y_train = np.log(self.y_train)
        else:
            self.X_train = scaler.fit_transform(np.array(self.X_train))
            self.X_test = scaler.transform(np.array(self.X_test))

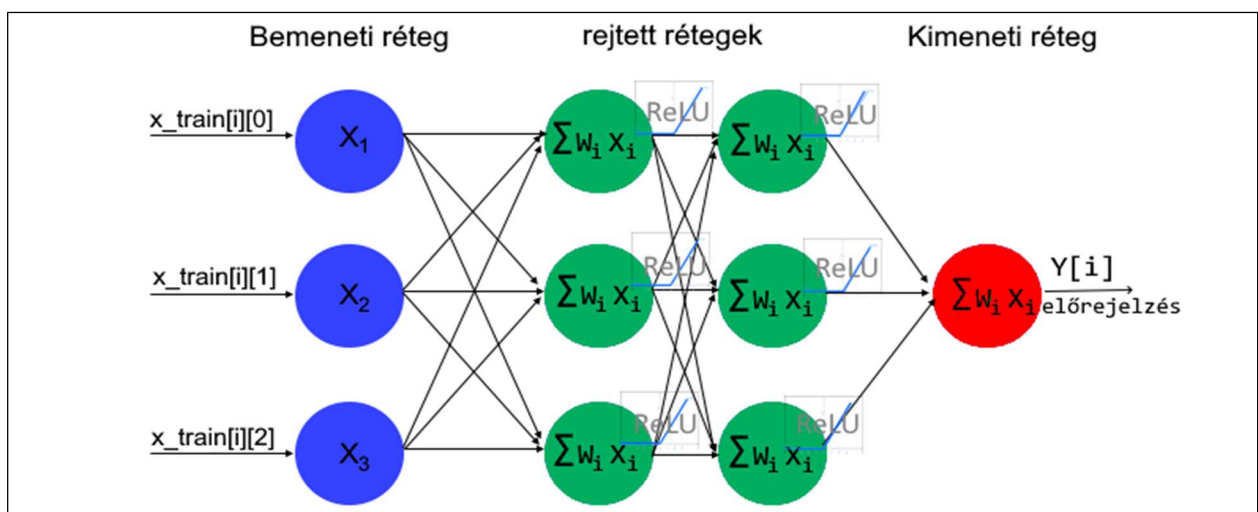
            if normOut:
                self.y_test = scaler.fit_transform(self.y_test.reshape(-1, 1))
                self.y_train = scaler.transform(self.y_train.reshape(-1, 1))

```

20. ábra. az idősorok normalizálását végző Python függvény (saját szerkesztés Python-ben)

4.4.3. Az MLP modellek implementációja

(Heaton, 2005) alapján és próbálgatások után választottam ki a megfelelő hálószerkezetet: A mozgó ablak mérete 3 volt (3 korábbi adatból következik egy negyedik), ebből adódóan mindegyik MLP modellnél 3 bemeneti neuront használtam. Mivel egy előrejelzést ad a modell a bemeneti mintából, a kimeneti rétegben egyetlen neuron van. Az MLP modellbe végül két rejtett réteget, bennük 3-3 neuront tettem, így elérve a legjobb általánosító képességet.



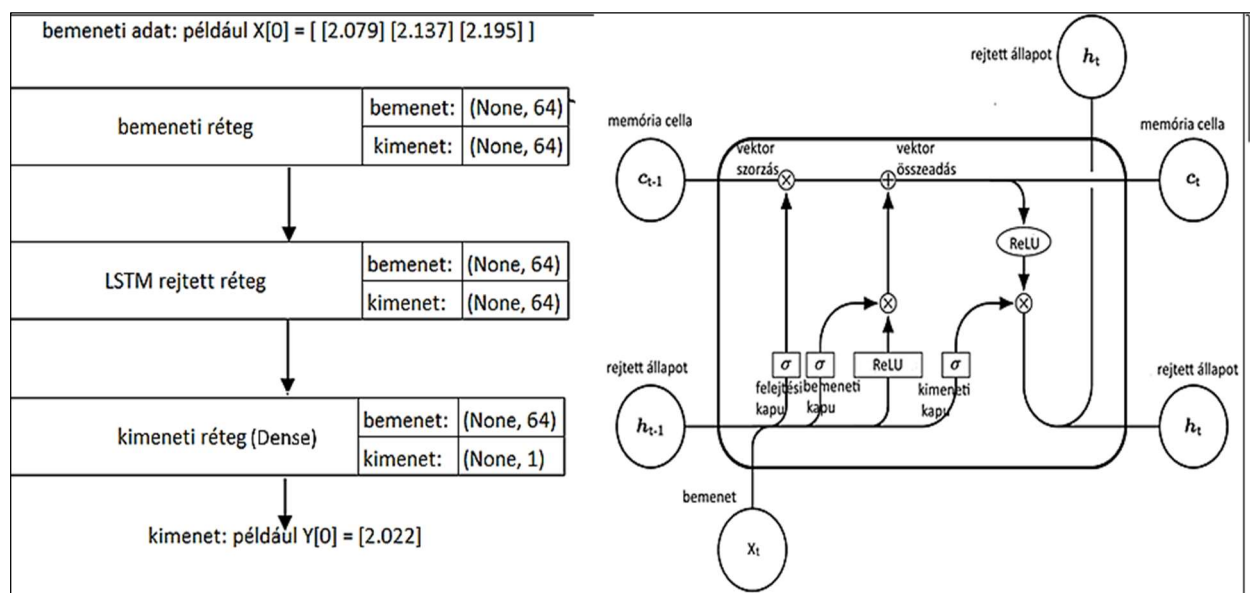
21. ábra. Az előrejelzésekhez használt MLP hálók szerkezete.
Forrás: Saját szerkesztés (Brassai, 2019), 80. oldal alapján

A legjobb MLP modell keresése során az MLP és LSTM modelleknél egyaránt összehasonlítottam, hogy az egyes idősorok esetében melyik optimalizációs algoritmus (ADAM/LBFGS/SGD), illetve az adatok milyen transzformációja (standardizálás, min-max normalizáció, robosztus skálázás, logaritmizálás, vagy egyik sem) mellett illeszkedtek, illetve általánosítottak legjobban a modellek. A bemeneti adatok standardizálása segített a legjobban a modellnek, az elvárt kimenetei értékek skálázása elrontotta az előrejelzéseket, azokat nem alakítottam át. Az LBFGS optimalizációs algoritmus által rövidebb idő alatt konvergáltak a modellek, és ezen állapotban jóval kisebb MSE/RRMSE/MAPE mutatókat értek el a teszt adatokon, mint az ADAM vagy SGD esetében. Az maximális epoch szám minden esetben 2000 volt. A modellek már legfeljebb 364 epoch után megfelelően konvergáltak, ezért a tanítás leállt. A konvergencia elérésére abból következtet a tanítást kezelő algoritmus, hogy megadott (alapból 10) súlyfrissítési művelet után sem javul a veszteség a megadott (alapból 10^{-4}) értékkel.

4.4.4 Az LSTM modellek implementációja

Az LSTM regressziós modelleket (Brownlee, 2022); (Brownlee, 2018) alapján készítettem el, a TensorFlow projekt részévé vált Keras modullal (Keras: Deep Learning for humans, 2024). Az úgynevezett Vanilla LSTM-et választottam, amely a be- és kimeneti rétegen kívül egyetlen rejtett rétegből áll. Ez a réteg is ReLU aktivációs függvényt használ, akárcsak az MLP esetében. Próbáltam 2 és 3 réteget is (Stacked LSTM), de nem adtak jobb eredményt, viszont jóval több ideig tartott a tanulásuk. A rejtett réteg után egy kimeneti réteget kapcsoltam, amely egyetlen LSTM cellából áll, és teljes összeköttetésben áll a rejtett réteggel. A teljesen összekapcsolt rétegeket sűrű rétegnek (Dense) is nevezik. Mivel nincs pontos útmutatás arról, hogy a rejtett réteg mennyi LSTM cellát kell tartalmazzon, kipróbáltam először 3 cellával, ami nagyon kevésnek bizonyult. A szakirodalomban gyakran használt 64 db LSTM cella már sokkal kielégítőbb eredményeket adott. Batch méretnek 1-et használtam. A batch méret azt határozza meg, hogy a háló egyszerre hány bemeneti mintát kap meg súlyfrissítés előtt. Mivel az MLP modellnél az ADAM/LBFGS/SGD optimalizációs algoritmusokat használtam, viszont a TensorFlow Keras modul LSTM osztályában nincs LBFGS, ezért az ADAM és SGD teljesítményét hasonlítottam össze. Az ADAM mellett jobban általánosítottak a modellek. Az epochs értéknek 300-at állítottam be. A tanítóadatok ugyanolyan szerkezetben voltak, mint az MLP esetében (19. ábra) és standardizálva voltak. Viszont, amíg az MLP esetében az elvárt kimenetek standardizálása nem javította jelentősen az általánosító képességet, (a teszt adatbázison kívül adott előrejelzéseket el is

elrontotta), addig az LSTM esetében éppen fordítva volt, tehát az y_{train} és y_{test} halmazokat is skáláznom kellett.



22. ábra. A felhasznált LSTM modell diagramja.
 Forrás: saját szerkesztés (Póda, 2021) alapján

4.4.5. Rekurzív előrejelzés a betanított neuronhálókkal

A tesztadatokon túl nem rendelkezésre több bemeneti minta, amiből a modellek tudnának jósolni, ezért a legutolsó bemeneti mintákból kell kiindulni. Az MLP és LSTM modelleknél Brownlee (2018) alapján egyaránt úgy jártam el, hogy a modell számára elkészített teszt adatbázis legutolsó bemeneti mintája alapján először egy előrejelzést készítettem, majd az előrejelzett értékkel frissítettem a bemeneti mintát úgy, hogy eggyel balra toltam az értékeket, így bemeneti minta legelső (legrégebbi) eleme kiesik, és helyére az előrejelzett érték jön.

Amennyiben a modell normalizált ki- és bemeneti adatokkal lett tanítva, az előrejelzéseket a felhasznált skálázási módszer szerint vissza kell transzformálni. Például standardizáció esetén a tanítóadatok átlagával és szórásával alakítjuk vissza az előrejelzett értéket a (22) képlet átrendezésével. Ha a modell számára csak a bemeneti adatokat normalizáltuk, de a kimeneteket

nem, akkor az előrejelzést is normalizálni kell, mielőtt felhasználnánk a frissített bemeneti mintában.

```
def recursiveForecast(model, n):
    forecasts = []
    input_data = model.x_test[-1].reshape(1, -1)

    for _ in range(n):
        forecast = model.predict(input_data)[0]

        if not normOut and scalerMode == "log":
            forecast = np.log(np.longdouble(forecast))

        forecasts.append(forecast)
        input_data = np.hstack((input_data[:, 1:], forecast.reshape(1, -1)))

        if model.scaler is not None and not model.normOut:
            last_value_normalized = model.scaler.transform(np.array(input_data).reshape(1, -1))[0][-1]
            input_data[-1][-1] = last_value_normalized

    if model.scalerMode == "log":
        forecasts = np.exp(forecasts).tolist()

    elif model.scaler is not None and model.normOut:
        forecasts = model.scaler.inverse_transform(np.array(forecasts).reshape(1, -1)).flatten().tolist()

    return forecasts
```

23. ábra. Rekurzív előrejelzéseket készítő Python függvény.

Forrás: saját szerkesztés, (Brownlee, 2018) alapján.

5. Eredmények

Ebben a fejezetben a modellek által (a teszt adatbázis bemeneteire) adott becslések pontosságát hasonlítom össze.

5.1. modellek általánosító képességeinek összehasonlítása

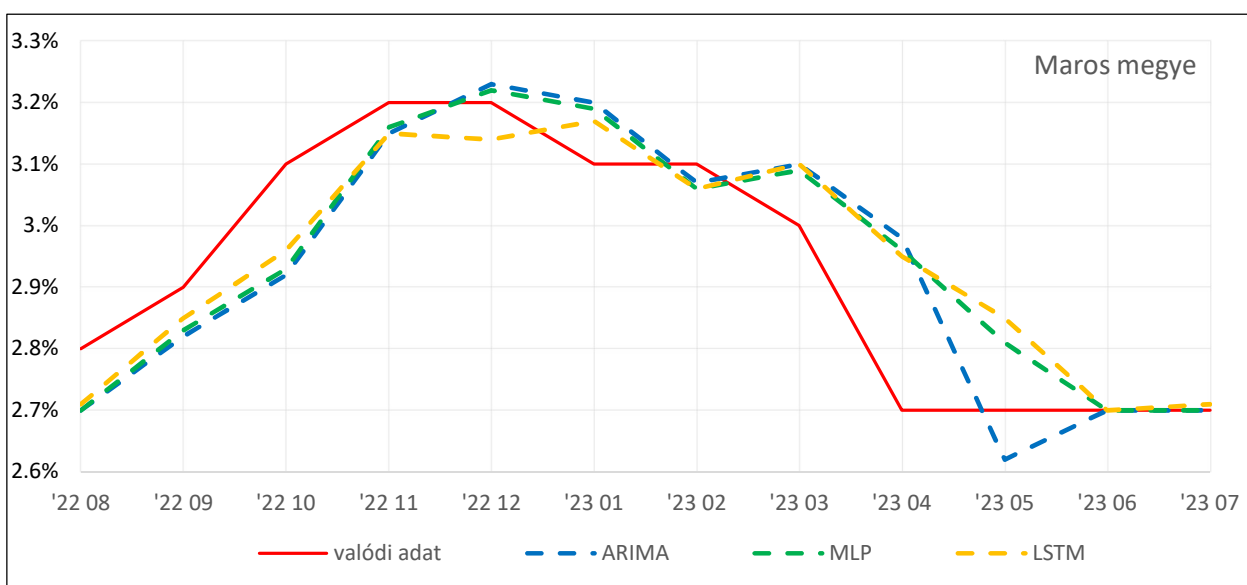
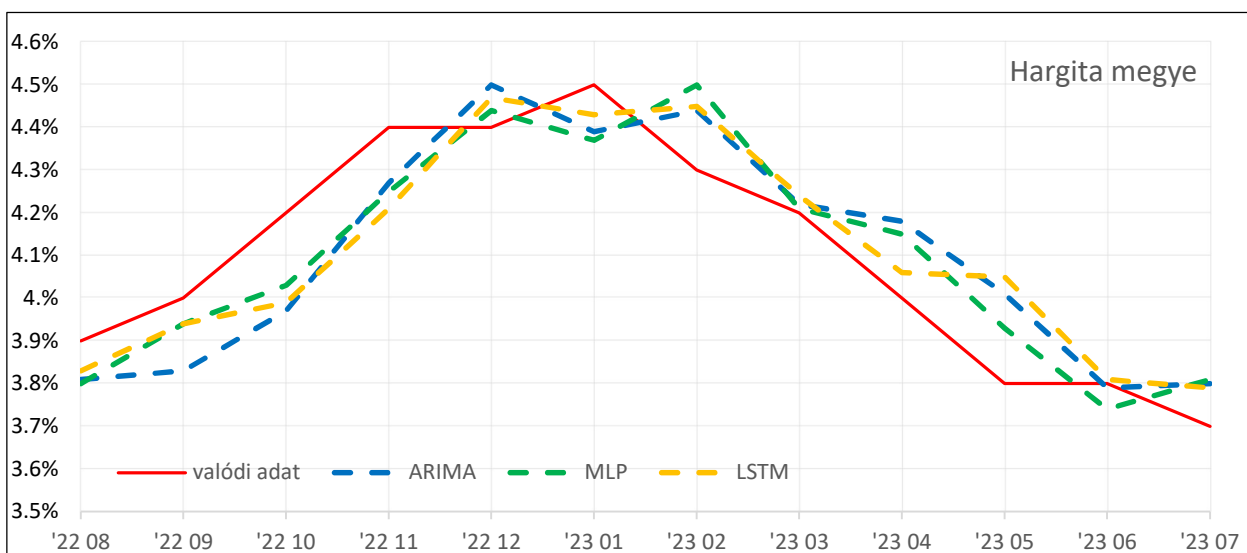
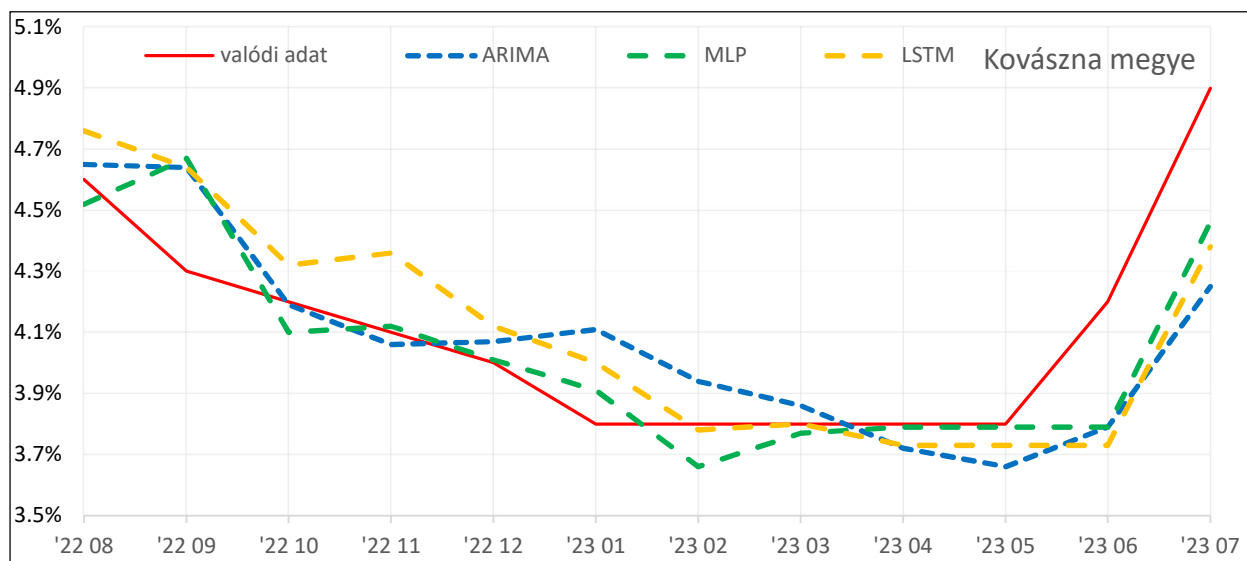
A webalkalmazásom a modellek illesztését és betanítását követően automatikusan előre jelez a tesztadatokra is, valamint kiszámítja a reziduumokat és a hibastatisztikákat. Ezeket idősoronként egy táblázatban összesíti, ahol az elvárt érték, a becslt érték és a közöttük levő eltérést mutatja. Ezenkívül a reziduumokat pontdiagramon ábrázolja, egy hisztogram és egy autokorrelációs diagram mellett, amelyek a modellek diagnosztikájára szolgálnak.

A 8. táblázatban összesítettem, hogy 2022 augusztusa és 2023 júliusa között hogyan alakultak a munkanélküliségi ráták a székelyföldi megyékben, és hogy a kiválasztott ARIMA, ANN modellek ehhez képest milyen értékeket jeleztek előre. A reziduumokat külön rendszereztem, amelyek a 9. táblázatban találhatóak.

8. táblázat. A székelyföldi megyék valódi munkanélküliségi rátái a teszt időszak alatt, és a különböző modellek becslései. Az értékek százalékban értendők.

modell időszak	Kovácsna				Hargita				Maros			
	valódi adat	ARIMA	MLP	LSTM	valódi adat	ARIMA	MLP	LSTM	valódi adat	ARIMA	MLP	LSTM
'22 08	4.6	4.65	4.52	4.76	3.9	3.81	3.8	3.83	2.8	2.7	2.7	2.71
'22 09	4.3	4.64	4.67	4.64	4	3.83	3.94	3.94	2.9	2.82	2.83	2.85
'22 10	4.2	4.19	4.1	4.32	4.2	3.97	4.03	3.99	3.1	2.92	2.93	2.96
'22 11	4.1	4.06	4.12	4.36	4.4	4.27	4.25	4.21	3.2	3.15	3.16	3.15
'22 12	4.0	4.07	4.01	4.12	4.4	4.5	4.44	4.47	3.2	3.23	3.22	3.14
'23 01	3.8	4.11	3.91	4.0	4.5	4.39	4.37	4.43	3.1	3.2	3.19	3.17
'23 02	3.8	3.94	3.66	3.78	4.3	4.44	4.5	4.45	3.1	3.07	3.06	3.06
'23 03	3.8	3.86	3.77	3.8	4.2	4.22	4.21	4.24	3.0	3.1	3.09	3.1
'23 04	3.8	3.72	3.79	3.73	4.0	4.18	4.15	4.06	2.7	2.98	2.96	2.95
'23 05	3.8	3.66	3.79	3.73	3.8	4.01	3.93	4.05	2.7	2.62	2.81	2.85
'23 06	4.2	3.79	3.79	3.73	3.8	3.79	3.74	3.81	2.7	2.7	2.7	2.7
'23 07	4.9	4.25	4.46	4.38	3.7	3.8	3.81	3.79	2.7	2.7	2.7	2.71
átlag:	4.11	4.08	4.05	4.11	4.10	4.10	4.10	4.11	2.93	2.93	2.95	2.95

Grafikonokon (24. ábra) ábrázoltam minhárom idősor 2022 augusztus és 2023 július közötti megfigyeléseit, valamint a modellek becsléseit. Ezeken az látszik, hogy a zöld színű, szaggatott vonallal jelölt MLP modellek becslései állnak a legközelebb a valódi adatokhoz.



24. ábra. Az ARIMA, MLP és LSTM modellek illeszkedése a tesztadatokra.
(saját szerkesztés Excel-ben)

A 9. táblázatban a modellek becsléseinek hibáit rendszereztem megyénként, valamint feltüntettem a rajtuk elvégzett Kolmogorov-Smirnov tesztek p-értékeit az utolsó sorban, amelyek szerint mindegyik modell reziduumaik sorozata követi a normális eloszlást.

9. táblázat. Az előrejelzési modellek becsléseinek reziduumaik a teszt időszakra vonatkozóan. Az adatok százalékban értendők.

időszak	modell	Kovászna			Hargita			Maros		
		ARIMA	MLP	LSTM	ARIMA	MLP	LSTM	ARIMA	MLP	LSTM
2022 08		-0.05	0.08	-0.16	0.09	0.1	0.07	0.10	0.10	0.09
2022 09		-0.34	-0.37	-0.34	0.17	0.06	0.06	0.08	0.07	0.05
2022 10		0.01	0.1	-0.12	0.23	0.17	0.21	0.18	0.17	0.14
2022 11		0.04	-0.02	-0.26	0.13	0.15	0.19	0.05	0.04	0.05
2022 12		-0.07	-0.01	-0.12	-0.1	-0.04	-0.07	-0.03	-0.02	0.06
2023 01		-0.31	-0.11	-0.2	0.11	0.13	0.07	-0.10	-0.09	-0.07
2023 02		-0.14	0.14	0.02	-0.14	-0.20	-0.15	0.03	0.04	0.04
2023 03		-0.06	0.03	0.00	-0.02	-0.01	-0.04	-0.10	-0.09	-0.10
2023 04		0.08	0.01	0.07	-0.18	-0.15	-0.06	-0.28	-0.26	-0.25
2023 05		0.14	0.01	0.07	-0.21	-0.13	-0.25	0.08	-0.11	-0.15
2023 06		0.41	0.41	0.47	0.01	0.06	-0.01	0.00	0.00	0.00
2023 07		0.65	0.44	0.52	-0.10	-0.11	-0.09	0.00	0.00	-0.01
hiba átlag		0.03	0.13	0.00	0.00	0.03	-0.01	0.00	0.00	-0.01
KS teszt p-értéke		0.70	0.62	0.54	0.79	0.76	0.99	0.85	0.95	0.72

A tesztadatokon mért hiba becslések összefoglalója a 10. táblázatban található, amely szemlélteti a többretegű perceptron modellek egyértelmű fölényét. Mindhárom megye esetében az MLP modell becsült a legkevesebb hibával az MSE, RRMSE és a MAPE mutatók alapján. Maros megyénél az ARIMA (1, 1, 0) minimálisan pontosabb volt, mint az LSTM modell, viszont a Hargita és Kovászna megyénél az ARIMA (2, 1, 2) teljesítménye volt a leggyengébb.

10. táblázat. A kiválasztott modellek becsléseinek pontosságát összehasonlító mutatók.

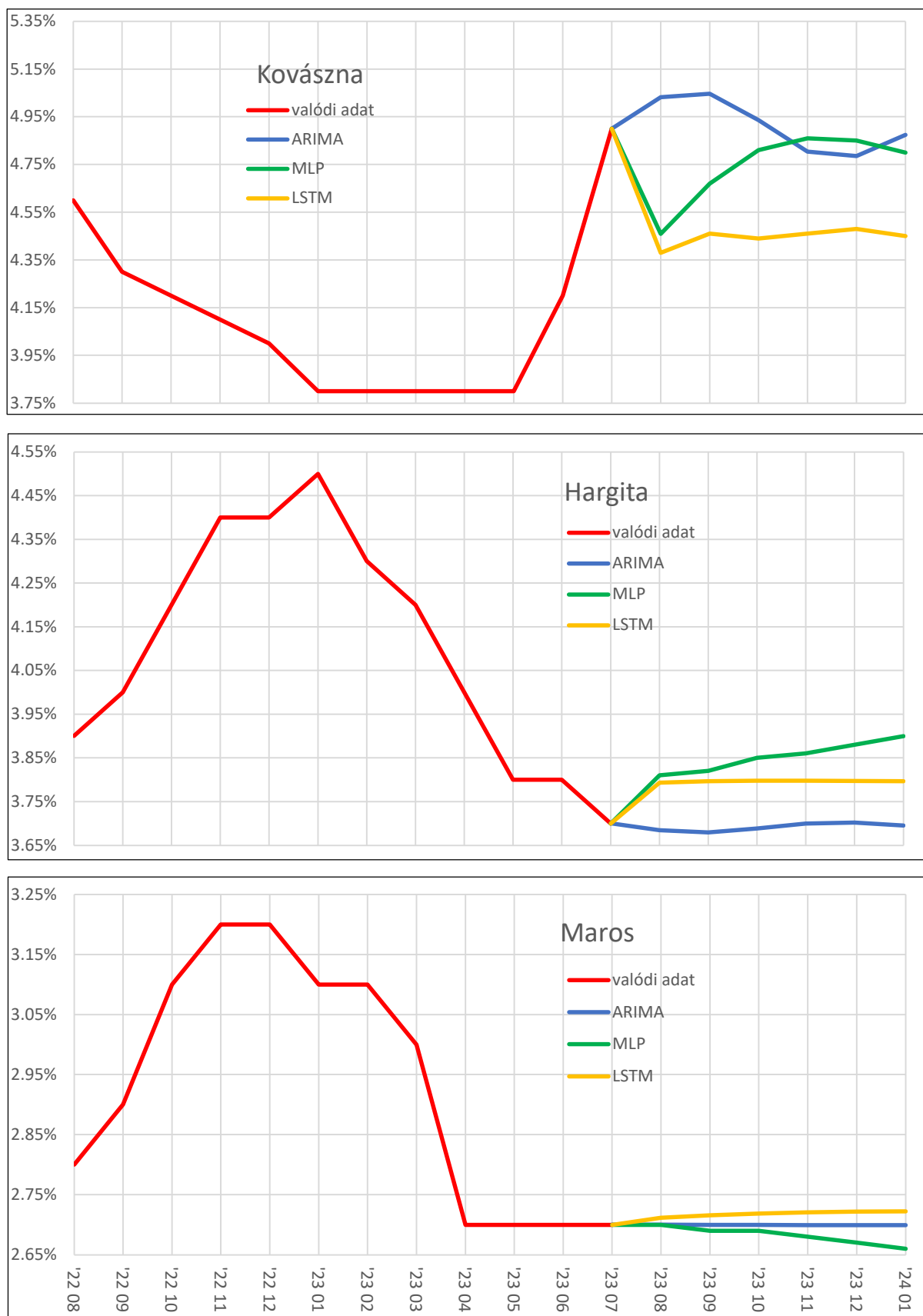
Megye	Modell	MSE	RMSE	RRMSE	MAPE	R ²
Hargita	MLP	0.015	0.084	0.029	2.65	0.78
	LSTM	0.016	0.126	0.031	2.75	0.76
	ARIMA (2,1,2)	0.020	0.141	0.034	3.04	0.72
Kovászna	MLP	0.045	0.200	0.053	3.45	0.61
	LSTM	0.065	0.255	0.062	4.57	0.54
	ARIMA (2,1,2)	0.071	0.266	0.066	4.69	0.48
Maros	MLP	0.012	0.114	0.037	2.84	0.69
	ARIMA	0.013	0.114	0.038	2.84	0.71
	LSTM	0.014	0.118	0.040	2.98	0.70

5.2. Előrejelzések

A 10. táblázatban felsorolt modellekkel további hat hónapra készítettem előrejelzéseket, 2023 augusztus és 2024 januári időszakra vonatkozóan.

Az ARIMA modellek előrejelzéseit ugyanazzal a módszerrel készítettem, mint a tesztadatok becslésénél (14. ábra). Az LSTM modellek előrejelzéseit pedig rekurzívan, a 18. ábrán látható eljárás szerint kértem le.

A 25. ábrán, grafikonok segítségével ábrázoltam a székelyföldi megyék 2023 augusztus és 2024 január között (ARIMA, MLP és LSTM modellek által) előrejelzett munkanélküliségi rátáit.



25. ábra. Féléves előrejelzések az ARIMA, MLP és LSTM modellekkel. (saját szerkesztés Excel-ben)

6. Következtetések

A kutatás során Hargita, Kovászna és Maros megye munkanélküliségi rátáit modelleztem ARIMA, MLP és LSTM prediktív modellekkel. Minden számítást Python-ben végeztem. Tanító adatbázisnak a 2010 január és 2022 július közötti havi adatokat használtam fel és elemeztem. A modellek általánosító képességének vizsgálatához a 2022 augusztus – 2023 júliusi megfigyeléseket használtam.

A kutatás során kiválasztott ARIMA és neurális háló modellek MAPE értéke összesítve 2.65 és 4.69 között alakultak a teszt adatbázisra adott becslésekre tekintve. Mindegyik modell megfelelt a reziduumokhoz kapcsolódó elvárásoknak.

A legjobb becsléseket az MSE, RRMSE és MAPE hibamutatók alapján a két rejtett réteggel rendelkező MLP modellek adták, LBFGS tanuló algoritmust használva. Az adatok standardizálva lettek a neurális hálók számára. A tanító- és tesztelő adatbázisok mozgó ablak (*sliding window*) módszerrel lettek elkészítve. A maximális epoch érték 2000 volt.

Eredményeink összhangban vannak Ajoodha és Mulaudzi (2020); Tufaner és Sözen (2021); eredményeivel, abból a szempontból, hogy az MLP neurális háló átlagosan jobban teljesített a munkanélküliségi ráták előrejelzésében, mindhárom idősor esetén, mint az LSTM vagy ARIMA modellek. Sasu (2013) kutatásának eredménye is összhangban van azzal, hogy az MLP modellek jobban jelezték előre a munkanélküliségi rátákat, mint az ARIMA modellek.

Továbbá a különböző gazdasági folyamatok előrejelzése során az LSTM prediktív modellek becslései megbízhatóbbnak bizonyultak, mint az ARIMA Bareith és Vancsura (2023); Muzaffar és Afshari (2018); S. Siami-Namini és A. S. Namin (2018); kutatásaiban. Ez összhangban van az én eredményemmel, miszerint Hargita és Kovászna megye LSTM modelljei pontosabban jelezték előre, mint az ARIMA modellek. Maros megye esetében is minimális volt a különbség.

Oukhouy és Himdi (2023) kutatásában, a tőzsdeindexek előrejelzésében az MLP modell jobban teljesített, mint az LSTM. Ez összhangban van az én kutatásommal, abban a tekintetben, hogy mindhárom megyére készített ARIMA, MLP és LSTM modellek közül az MLP modellek általánosítottak a legjobban. Az MLP modellek tanulási folyamata egyébként lényegesen rövidebb volt, hamarabb konvergáltak, mint az LSTM modellek, kevesebb erőforrást igényelt a tanítási folyamatuk.

(Paszkiewicz, et al., 2023) eredményei szerint, a vágóeszközök élettartamának előrejelzésében az LBFGS optimalizációs algoritmus mellett pontosabb becsléseket adott az MLP modell, mint az ADAM használata esetén. A kutatásom során én is azt tapasztaltam, hogy a többrétegű perceptron modellek jobban általánosítottak az LBFGS mellett.

Chen és társai (2022) eredményeik alapján az LSTM prediktív modellek jobban teljesítenek akkor, ha a bemeneti adatok normál eloszlást követnek. Én is ezt tapasztaltam a kutatásom során.

A Django keretrendszer, a Python alprogramokkal összekötve lehetővé teszi, hogy a prediktív neuronháló modellek, és velük az előrejelzések készítése felhasználóbarát legyen. A programom nehezen elérhető Python elemzési modulokat, akár csak az adat vizualizációt, könnyen elérhetővé teszi, akár csak bármelyik mai korszerű statisztikai program.

7. Irodalomjegyzék

- Adenomon, M. O. (2017). Modelling and forecasting unemployment rates in Nigeria using ARIMA model. *FUW Trends in Science and Technology Journal*, 2(1B), 525-531.
- Ajoodha, R., & Mulaudzi, R. (2020. November 25-27). An Exploration of Machine Learning Models to Forecast the Unemployment Rate of South Africa: A Univariate Approach. *An Exploration of Machine Learning Models to Forecast the Unemployment Rate of South Africa: A Univariate Approach*. Kimberley, Dél-Afrika: IEEE.
doi:10.1109/IMITEC50163.2020.9334090
- Awan, A. (2022. március). Recurrent Neural Network Tutorial (RNN). Forrás:
<https://www.datacamp.com/tutorial/tutorial-for-recurrent-neural-network>
- Bamberger, S., Heckel, R., & Krahmer, F. (2023. 08 5). *Approximating Positive Homogeneous Functions with Scale Invariant Neural Networks*. doi:10.48550/arXiv.2308.02836
- Bareith, T., & Vancsura, L. (2023). Prediktív modellek teljesítményének vizsgálata Covid-19 és az orosz-ukrán háború időszakában. *Pénzügyi Szemle*, 69(2), 123–138.
doi:10.35551/PFQ_2023_2_7
- Brassai, S. T. (2019). *Neurális hálózatok és Fuzzy logika*. Kolozsvár: Scientia Kiadó. Forrás:
<http://real.mtak.hu/id/eprint/122603>
- Brownlee, J. (2018). *Predict the Future with MLPs, CNNs and LSTMs in Python*. Forrás:
<https://dokumen.pub/deep-learning-for-time-series-forecasting-predict-the-future-with-mlps-cnns-and-lstms-in-python.html>
- Brownlee, J. (2019). How to Scale Data for Long Short-Term Memory Networks in Python. Forrás: <https://machinelearningmastery.com/how-to-scale-data-for-long-short-term-memory-networks-in-python/>
- Brownlee, J. (2021). A Gentle Introduction to Long Short-Term Memory Networks by the Experts. Letöltés dátuma: 2024. március, forrás:
<https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/>

- Brownlee, J. (2021. október 12). A Gentle Introduction to the BFGS Optimization Algorithm. *Machine Learning Mastery*. Forrás: <https://machinelearningmastery.com/bfgs-optimization-in-python/>
- Brownlee, J. (2021. január 13). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. *Machine Learning Mastery*. Forrás: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>
- Brownlee, J. (2022. 08 07). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. Forrás: <https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/>
- Brownlee, J. (2023. november 18). *How to Create an ARIMA Model for Time Series Forecasting in Python*. Letöltés dátuma: 2024. március, forrás: machinelearningmastery.com: <https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>
- Chen, H., Bu, Y., Zong, K., Huang, L., & Hao, W. (2022). The Effect of Data Skewness on the LSTM-Based Mooring Load Prediction Model. *Journal of Marine Science and Engineering*, 10(12). doi:10.3390/jmse10121931
- Davidescu, A. A., Apostu, S.-A., & Paul, A. (2021). Comparative Analysis of Different Univariate Forecasting Methods in Modelling and Predicting the Romanian Unemployment Rate for the Period 2021–2022. *Entropy*, 23(325), 324. doi:10.3390/e23030325
- Derbentsev, V., Matviychuk, A., Datsenko, N., Bezkorovainyi, V., & Azaryan, A. (2020). Machine learning approaches for financial time series. *M3E2-MLPEED 2020* (old.: 434-450). Odessa: CEUR Workshop Proceedings. doi:<https://doi.org/10.31812/123456789/4478>
- Dhandi, A. (2024). *Mastering Django: Projects for Beginners*.
- Didiharyono, D., & Syukri, M. (2020. 03 03). Forecasting with ARIMA Model in Anticipating Open Unemployment rates in South Sulawesi. *International journal of scientific and technology research*, 9, 3838-3841. Forrás: <https://osf.io/f5zh6/>

- Drelczuk, K. (2020. 05 07). *ACF (autocorrelation function) — simple explanation with Python example*. Forrás: Medium: <https://medium.com/@krzysztofdrelczuk/acf-autocorrelation-function-simple-explanation-with-python-example-492484c32711>
- Georgeta, E. S. (2015). *The economic and social situation in Romania*. Brussel, Belgium: European Economic and Social Committee. doi:10.2864/484519
- Heaton, J. (2005). *Introduction to Neural Networks with Java*. St. Louis: Heaton Research.
- INSSE. (2016). Metodologie Somaj BIM lunar. Forrás: https://insse.ro/cms/files/statistici/comunicate/somaj%20BIM/a16/precizari_metodologie_ian2016.pdf
- Keras: Deep Learning for humans*. (2024). Forrás: Keras documentation: <https://keras.io/about/>
- Lázár, E. (2022). *Közgazdasági kutatómódszertan*. Kolozsvár: Risoprint kiadó.
- Madaras, S. (2014). A gazdasági válság hatása a munkanélküliség alakulására országos és megyei szinten Romániában. *Közgazdász Fórum, 17 (1-2)*, 136–149. Forrás: <https://epa.oszk.hu/00300/00315/00108/pdf/>
- Madaras, S. (2018). Forecasting the regional unemployment rate based on the Box-Jenkins methodology vs. the Artificial Neural Network approach. Case study of Braşov and Harghita counties. *Közgazdász fórum, 21(135)*, 66-79. Letöltés dátuma: 2024. február
- Madaras, S., & Lehel, G. (2016. december). Non-Linearity and Non-Stationarity of Exchange Rate Time Series in Three Central-Eastern European Countries Regarding the CHF Currency in 2014 and 2015. *Acta Universitatis Sapientiae Economics and Business, 4(1)*, 33-41. doi:10.1515/auseb-2016-0002
- Menon, A., Aishwarya, M., Joykutty, A., Av, A., & Av, A. (2021). Data Visualization and Predictive Analysis for Smart Healthcare: Tool for a Hospital. *2021 IEEE Region 10 Symposium (TENSYP)*. Grand Hyatt Jeju: IEEE. doi:10.1109/TENSYP52854.2021.9550822
- Muzaffar, S., & Afshari, A. (2018). Short-Term Load Forecasts Using LSTM Networks. *10th International Conference on Applied Energy (ICAE2018)*. Hong Kong. doi:10.1016/j.egypro.2019.01.952

- Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., & S., S. (2020). Deep Learning for Stock Market Prediction. *Entropy*, 22(8). doi:10.3390/e22080840
- Oukhouya, H., & El Himdi, K. (2023). Comparing Machine Learning Methods—SVR, XGBoost, LSTM, and MLP— For Forecasting the Moroccan Stock Market. *Computer Sciences & Mathematics Forum*, 7(39). doi:10.3390/IOCMA2023-14409
- Paszkievicz, A., Piecuch, G., Żabiński, T., Bolanowski, M., Salach, M., & Rączka, D. (2023). Estimation of Tool Life in the Milling Process—Testing Regression Models. *Sensors*, 23. doi:10.3390/s23239346
- Póda, C. (2021. január 24). Hogyan működnek az LSTM-modellek? Forrás: <https://www.mlbudapest.com/blog/lstm-modellek>
- Punasya, D., Kushwah, H., Jain, H., & Sheikh, R. (2021). An application for sales data analysis and visualization using Python and Django. *International Research Journal of Modernization in Engineering Technology and Science*, 3(6), 1757-1672.
- Ramli, S. F., Fidaus, M., Uzair, H., Khairi, M., & Zharif, A. (2018). Prediction of the Unemployment Rate in Malaysia. *International Journal of Modern Trends*, 1(4), 38-44. Forrás: <http://www.ijmtss.com/PDF/IJMTSS-2018-04-12-04.pdf>
- Russell, S., & Norvig, P. (2005). *Mesterséges intelligencia modern megközelítésben (második kiadás)*. Budapest: Hungarian Translation Panem Könyvkiadó.
- Sándor, Z. (2019). *Bevezetés az ökonometriába*. Sepsiszentgyörgy: T3 Kiadó.
- Sándor, Z., & Tánczos, L. (2019). *Gazdasági statisztika jegyzet*. Sepsiszentgyörgy: T3 kiadó.
- Sasu, A. (2013). A quantitative comparison of models for univariate time series forecasting. *Bulletin of the Transilvania University of Brasov. Series III: Mathematics and Computer Science*, 6(55), 117-124. Forrás: https://webbut.unitbv.ro/index.php/Series_III/article/view/2005
- Siami-Namini, S., & Siami Namin, A. (2018). Forecasting Economics and Financial Time Series: ARIMA vs. LSTM. *arXiv:1803.06386*. doi:10.48550/arXiv.1803.06386
- Tóth, A., Kálmán, B., & Juhász, T. (2021). Unemployment development in the Visegrád Group countries as a result of the COVID-19 pandemic. *Reproduction of Human Capital –*

Mutual Links and Connections (RELIK 2021), 780-787. Forrás:

<https://real.mtak.hu/id/eprint/191199>

Tufaner, M. B., & Sözen, İ. (2021). Forecasting Unemployment Rate in the Aftermath of the Covid-19 Pandemic: The Turkish Case. *İzmir Journal of Economics*, 36, 685 - 693.
doi:10.24988/ije.202136312

Vamsi, K., Lokesh, P., Reddy, K., & Swetha1, P. (2019). Visualization of Real World Enterprise Data using Python Django Framework. *IOP Conf. Series: Materials Science and Engineering*. doi:10.1088/1757-899X/1042/1/012019

Weytjens, H., Lohmann, E., & Kleinstauber, M. (2021). Cash flow prediction: MLP and LSTM compared to ARIMA and Prophet. *Electronic Commerce Research*(21), 371-391.
doi:10.1007/s10660-019-09362-7

Yamacli, D. S., & Yamacli, S. (2023). Estimation of the unemployment rate in Turkey: A comparison of the ARIMA and machine learning models including Covid-19 pandemic periods. *Helyion*, 9(1). doi:10.1016/j.heliyon.2023.e12796