SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
CSÍKSZEREDAI KAR
GAZDASÁGI INFORMATIKA SZAK

D I P L O M A D O L G O Z A T

D O L G O Z A T C Í M E

Végzős Hallgató:
Károlyi Krisztián

Témavezető(:
Dr. Madaras Szilárd, egyetemi adjunktus

2024

vázlat

Kivonat

Rezumat

vázlat

Abstract

Tartalomjegyzék

vázlat

1. Bevezetés

A munkanélküliség hosszú ideje jelentős gazdasági mutató és központi téma a közgazdaságtani kutatásokban. A munkanélküliség alakulása és változása az adott régió gazdasági egészségét tükrözi, és fontos információkat szolgáltathat a gazdasági kilátásokról. Az idősorok elemzése és az előrejelzés az egyik kulcsfontosságú eszköz lehet a munkanélküliség változásainak megértésében és a gazdasági intézkedések megalapozásában.

IDE MAJD kellene pár cikk a romániai munkanélküliségről, hogy miért aktuális téma…

Ebben az államvizsga dolgozatban Hargita, Kovászna és Maros megye havi munkanélküliségi rátáit vizsgálom 2010 január és 2022 szeptember között. Az adatokat Románia országos statisztikai hivatalának (Institutul Național de Statistică) hivatalos oldaláról töltöttem le. Célom, hogy statisztikai elemzést készítsek ezekről az idősorokról, valamint megvizsgáljam, hogy ezen idősorok esetében, tíz hónap távlatában (2022 október – 2023 július) a Box-Jenkins eljárással készült ARIMA modellek, vagy a gépi tanuláson alapuló MLP (többrétegű perceptron) neuronháló modellek nyújtanak pontosabb előrejelzéseket.

A kutatás során az adatok beolvasásához, feldolgozásához és az eredmények megjelenítéséhez egy Django webalkalmazás segítségével biztosítok felhasználói felületet, ezzel szemléletesebbé és egyszerűbbé téve a különböző statisztikai számításokat. A webalkalmazás lényegében bármennyi és bármilyen idősort képes elemezni a megfelelően előkészített adatforrásokból, tehát a jövőben még fel lehet használni más tematikájú elemzésekhez is.

vázlat

1.1 Szakirodalmi áttekintés

Ebben a részben röviden ismertetek néhány kutatást, ahol…

(Madaras, 2018) Hargita és Brassó megye esetében végzett hasonló regionális kutatást, és megállapította, hogy rövidtávon a mesterséges neuronháló-alapú NAR (nemlineáris autoregresszív) modell, középtávon az ARMA modell nyújtott pontosabb becsléseket.

(Davidescu, Apostu, & Paul, 2021) Románia országos munkanélküliségi rátáival végeztek kutatást, SARIMA, SETAR, Holt-Winters, ETS, és NNAR modelleket hasonlítottak össze, amelyek közül több szempont alapján is az NNAR (neuronhálós autoregresszív) modell mutatkozott a legjobbnak.

(Madaras, 2014) 2005 január és 2013 június közötti romániai munkanélküliségi ráta adatok alapján ARIMA (1, 1, 4) típusú autoregressziós modellel középtávú előrejelzést készített, és megjósolta a munkanélküliek számának növekedését 2013 július és 2014 február között Romániában.

(Ajoodha & Mulaudzi, 2020) Dél-Afrika országos munkanélküliségi rátáinak előrejelzéséhez hasonlítottál össze a gépi tanulást a hagyományos statisztikai módszerekkel, a három rejtett rétegű MLP modell jobban teljesített, mint a az ARIMA, a Ridge vagy a Holt-Winters modellek.

(Tufaner & Sözen, 2021) Törökország esetében hasonlítottak össze egy két rejtett rétegű MLP, és egy ARIMA (3, 1, 2) modellt, itt szintén az MLP volt a jobb.

vázlat

1.2 A munkanélküliségi ráta fogalmának meghatározása Romániában

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

A **munkanélküliek** a nemzetközi meghatározás (BIM[1]) 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** tulajdonképpen egy százalékos arányszám: a munkanélküliek arányát számolja ki a munkaerőhöz viszonyítva egy adott térségre. Tehát beszélhetünk országos, regionális vagy megyei munkanélküliségi rátáról.

A gazdaságilag aktív népesség a bázisidőszakban az áruk és szolgáltatások előállítására rendelkezésre álló munkaerőt biztosító valamennyi személyt magában foglalja, beleértve a foglalkoztatottakat és a munkanélkülieket is. (INSSE, 2016)
Tehát a munkanélküliségi ráta megmutatja, hogy a munkaképes lakosság hány százaléka nem rendelkezik a mérés pillanatában munkahellyel, viszont tudna és akarna dolgozni.

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

[1] Biroul Internaţional al Muncii

vázlat

1.3 Felhasznált statisztikai mutatók és fogalmak

Átlag: Jelöljünk n db megfigyelést x_1, x_2, ..., x_n -nel. A megfigyelések összegét elosztjuk a megfigyelések számával.

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

Szórás: A szórás azt méri, hogy a megfigyelések mennyire térnek el az átlagtól. A szórás az eltérések négyzetének átlagának a négyzetgyöke.

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

A variancia (szórásnégyzet):

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

$$s_x^2 = \frac{1}{n}\sum_{i=1}^{n} \left(x_i - \underline{x}\right)^2$$

Medián: Jelöljük x_1, x_2, ..., x_n –nel a megfigyeléseket, és jelöljük $x_{(1)}$, $x_{(2)}$, ... , $x_{(n)}$ –nel ugyanezeket a megfigyeléseket növekvő sorrendben. Tehát, $x_{(1)}$ a legkisebb, $x_{(2)}$ a következő, ... és $x_{(n)}$ legnagyobb:

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$$

A medián a sorrendbe állított x_1, x_2, ..., x_n megfigyelések középső megfigyelése. Ha n páratlan, akkor egészen egyszerű; a medián a (n + 1) / 2 sorrendű megfigyelés.

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

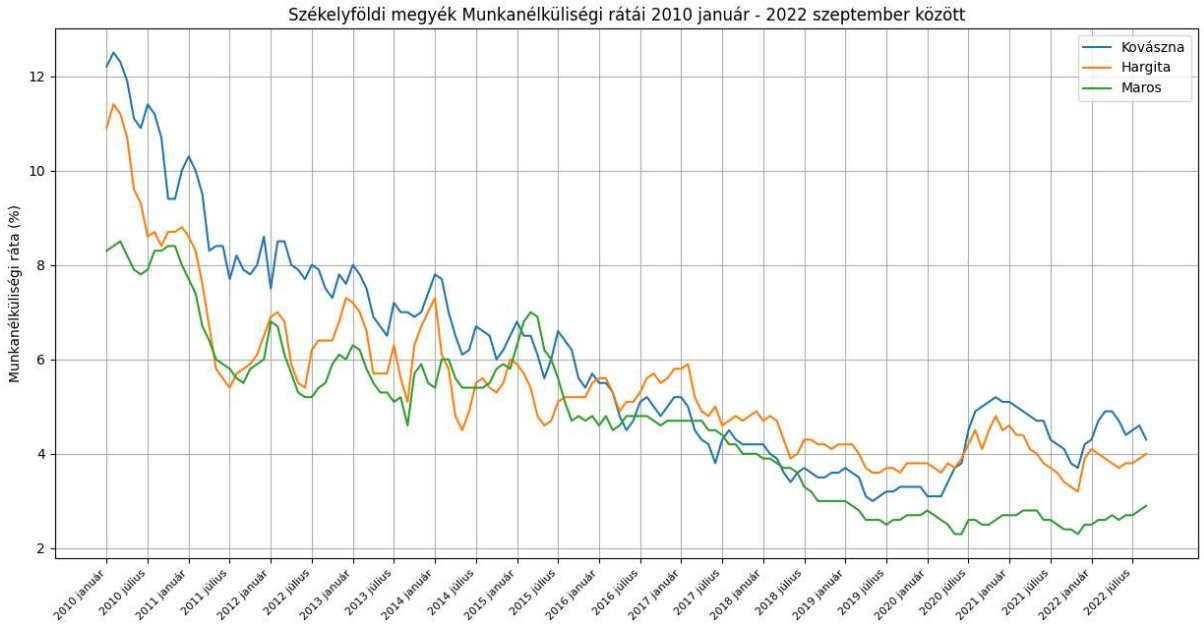
$$\frac{x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)}}{2}$$

(Sándor & Tánczos, Gazdasági statisztika jegyzet, 2019)

vázlat

2 Az idősorok elemzése

A webalkalmazásomba feltöltöttem a 2010 január – 2022 szeptember közötti munkanélküliségi ráta adatsorokat a három megyére. Az adatok Python-ben való beolvasását, tárolását és feldolgozását az X. fejezetben ismertetem részletesen. A feldolgozás után a program a következő grafikont készítette el:



A regionális munkanélküliségi ráták grafikonja
Forrás: saját ábra

Látszik, hogy Maros megyében szinte végig a legalacsonyabb a munkanélküliségi ráta, míg Kovásza megyében a legmagasabb. Sok periódusban megfigyelhető, hogy télen magasabb a mutató, mint nyáron. Szerencsére a 2008-as válság óta folyamatosan csökken az arány, viszont a Covid járvány idején sajnos egy nehezebb periódus jelei láthatóak.

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

vázlat

Azonban 2020 tavaszáig összeségében nézve folyamatosan csökkent a munkanélküliek száma, ez jól leolvasható a grafikonról, a gazdaság folyamatosan fejlődött, ehhez a technológiai fejlődés is hozzájárult. 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

A következő táblázat szemlélteti a három idősor átlagát, szórását, szórásnégyzetét (variancia), mediánját, minimum és maximum érétkeit. A táblázatban látható, hogy a téli hónapokban általában magasabb volt a munkanélküliség, mint a többi évszakban.

mutató	Kovászna	Hargita	Maros
Átlag	5.96%	5.41%	4.67%
Szórás	2.26%	1.65%	1.74%
Variancia	5.09%	2.71%	3.01%
Medián	5.2%	5.1%	4.7%
Minimum	*3.00%, 2019 május*	*3.20%, 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	5.86%	5.10%	4.47%
őszi átlag	5.72%	5.28%	4.55%

1. Táblázat: Az idősorok statisztikai mutatói

A következő lépésben röviden ismertetem a Box-Jenkins eljárást, majd Python-ben elkészítem az előrejelzésekhez a regressziós modelleket.

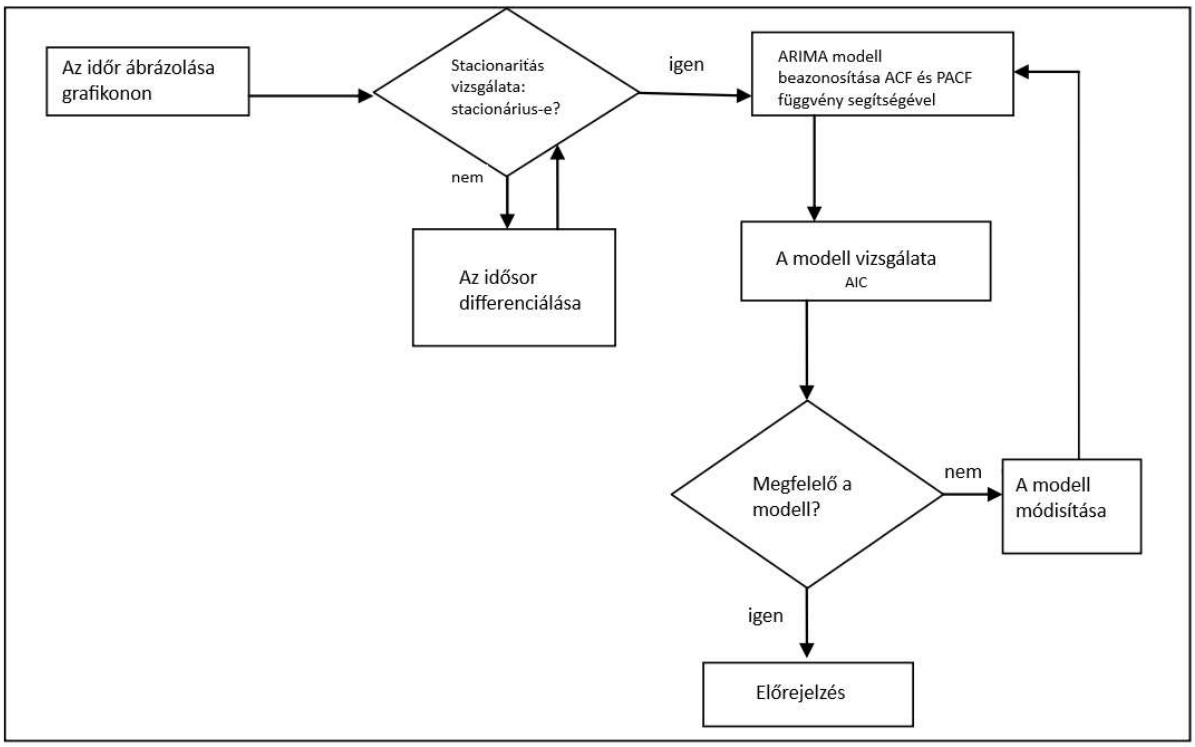
vázlat

3 Előrejelzés Box-Jenkins módszerrel

Az eljárás nevét két fő proponensétől, George Box-ról és Gwilym Jenkins-ről kaptam ők alkották meg az integrált autoregresszív és mozgóátlag modellt. Az alapgondolat az, hogy az idősorokat stacioner ARIMA (q, d, q) modellel írjuk le. A paramétereket a lehető legjobban kell behatárolni a modell pontosságának érdekében.

A főbb lépések a következők:

1) A stacionaritás vizsgálata (pl. ADF, KPSS segítségével). Ha nem stacioner az idősor, differenciálni kell.
2) AR(p) és MA(q) komponensek paramétereinek behatárolása PACF és ACF tesztek segítségével.
3) Modellminősítés: A megfelelő modellt (AR/MA/ARMA/ARIMA) az Akaike Information Criterion (AIC) együtthatók segítenek kiválasztani.
4) Előrejelzés készítése és annak pontosságának meghatározása (például MSE, RRMSE muatókal)
   Ha nem elég jók az eredmények, másféle modellt is ki kell próbálni.



A Box-Jenkins eljárás folyamatábrája
Forrás: (Lal & Jose, 2013), 30. oldal alapján

vázlat

3.1 A stacionaritás vizsgálata

A stacionaritás az idősorok statisztikai tulajdonságainak időbeni állandóságát vagy közelítő állandóságát jelenti. Egy stacionárius idősor esetén a várható értéke, varianciája és autokorrelációs függvénye állandó, vagy csak időben állandó konstans eltolódásokkal változik. Stacionárius idősorok könnyebben modellezhetők és előrejelzhetőek. A stacionaritás meglétét többféleképpen meg lehet állapítani, én a következő teszteket használtam:

- Augmented Dickey-Fuller (ADF) teszt: Az ADF teszt az egységgyökér jelenlétét vizsgálja az idősorban, és ezzel segít megérteni, hogy az idősor stacionárius-e vagy sem. A nullipotézis (H0) az, hogy az idősorban van egységgyökér, vagyis az idősor nem stacioner, míg az alternatív hipotézis (H1) az, hogy nincs egységgyökér, tehát az idősor stacionáriusnak mondható.
  Ha a p-érték (szignifikancia szint) kisebb, mint 0.05, akkor elutasítjuk a nullhipotézist, tehát az idősor stacioner, mert nincs kimutatható egységgyökér.
  A programomban a Python statsmodels.tsa.stattools csomagjából az adfuller függvényt használtam fel a teszt elvégzéséhez.

- Kwiatkowski-Phillips-Schmidt-Shin (KPSS) teszt: A nullhipotézis az, hogy az idősor szigorúan stacionárius (az ADF hipotézisével ellentétben), tehát nincs egységgyökér. Akkor fogadjuk el H0-t, ha a p-érték nagyobb, mint 0.05.
  A KPSS teszt lineáris regresszión alapul. Egy sorozatot három részből áll: determinisztikus trend (β_t), véletlenszerű séta (walk, r_t) és stacionárius hibára (ε_t).

$$x_t = r_t + \beta_t + \varepsilon_t$$

  A programomban a Python statsmodels.tsa.stattools csomagjából a kpss függvényt használtam fel, ami teljesen hasonlóan működik, mint az előbb említett függvény.

```
def Stationarity(self):
    adf_result = adfuller(self.adatok)
    self.adf["adf_stat"] = round(adf_result[0], 2)
    self.adf["p_value"] = round(adf_result[1], 2)
    self.adf["critical_values"]['5'] = round(adf_result[4]["5%"], 2)
    kpss_result = kpss(self.adatok)
    self.kpss["kpss_stat"] = round(kpss_result[0], 2)
    self.kpss["p_value"] = round(kpss_result[1], 2)
    self.kpss["critical_values"]['5'] = round(kpss_result[3]["5%"], 2)
```

vázlat

A Django webalkalamzásom segítségével elvégeztem a beolvasott adatsorokra ezeket a teszteket, és a következő eredményeket kaptam:

2.. táblázat: ADF és KPSS tesztek

megye	ADF			KPSS		
	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

ADF: Összességében, Kovászna megye esetében erős bizonyíték van a stacionaritásra, Hargita megye esetében bizonyos mértékben lehet, hogy stacionárius, míg Maros megye esetében nem találtunk erős bizonyítékot a stacionaritásra.

KPSS: Az értékek alapján minden megye esetében a teszt statisztikája nagyobb, mint a kritikus érték, és a p-érték kisebb, mint a hagyományos 0.05-ös szignifikanciaszint. Ez azt jelzi, hogy elutasítjuk a nullhipotézist, vagyis van bizonyíték arra, hogy az idősorok nem stacionáriusak. Összességében az eredmények azt sugallják, hogy Kovászna, Hargita és Maros megye esetében van egy olyan trend vagy kiegészítő komponens az idősorokban, ami miatt azok nem tekinthetők stacionáriusnak**.**

A bizonytalan eredmények miatt mindegyik megye esetében kipróbáltam a ARIMA (p, 1, q) modellekett is. A következő részben ismertetem az autoregresszív és mozgóátlag modelleket.

vázlat

3.2 Autoregresszív és mozgóátlag modellek (AR, MA, ARMA, ARIMA)

Az autoregresszió (AR) azt jelenti, hogy az aktuális időpontbeli értéket a korábbi időpontbeli értékek határozzák meg. Az AR komponens arra utal, hogy az aktuális érték korrelál az előző időpontbeli értékekkel. A "p" paraméter megadja az autoregressziós rendszámot, vagyis hány darab előző időpontbeli értéket használunk az aktuális érték becsléséhez.

Egy p-rendű autoregresszív modellt, amelynek jelzése $AR_{(p)}$, a következőképpen értelmezünk:

$$y_t = \alpha + \phi_1 y_{t-1} \pm \cdots + \phi_p y_{t-p} + \varepsilon_t, \quad t = p+1, \ldots, n$$

Az $\phi_1, \ldots, \phi_p$ 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.

Egy AR(p) folyamat akkor stacionárius, ha:

- Az autoregresszív $\phi_1, \ldots \phi_p$ paraméterek abszolútértékei kisebbek, mint 1
- Ezen paraméterek összegének abszolútértékei kisebb, mint 1

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)) jelölése:

$$y_t = \alpha + \theta_1 \varepsilon_t + \varepsilon_{t-1} + \cdots + \theta_1 \varepsilon_{t-q}, \quad t = q+1, \ldots, n$$

Ezek a modellek mindig **stacionáriusok**, vagyis a folyamat autokorrelációi nem változnak az idő függvényében, tehát az idősorban nincsenek trendek és szezonális mintázatok.

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

$$y_t = \alpha + \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$$

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

vázlat

Az ARIMA(p, d, q) modellekben az I (integrated) azt jelenti, hogy az idősort a szezonmentesítés érdekében d alkalommal differenciáljuk, vagyis először különbséget veszünk az aktuális értékek és az előző időpontok értékei között, így az idősorokat stacionáriussá tesszük.

(Sándor, 2019)

3.3 Autokorrelációs és parciális autokorrelációs tesztek

Ebben a részben megvizsgálom és elemzem az autokorrelációs (a továbbiakban ACF) és parciális autokorrelációs (a továbbiakban PACF) teszteket annak érdekében, hogy meg tudjam határozni a lehető legjobban illeszkedő idősor modelleket és azok paramétereit (AR(p), MA(q), ARMA(p, q), ARIMA(p, d, q)) az egyes idősorokra.

3.3.1 ACF

Egy idősor autokorrelációs függvénye (ACF) az autokorrelációk sorozata: $P_k = corr\,(y_t, y_{t-k}) = \frac{\gamma_k}{\gamma_0},\ k = 1, 2, \ldots,$ ahol $\gamma_k = E[(y_t - \mu)\,(y_{t-k} - \mu)]$ a k-ad rendű autokovariancia.

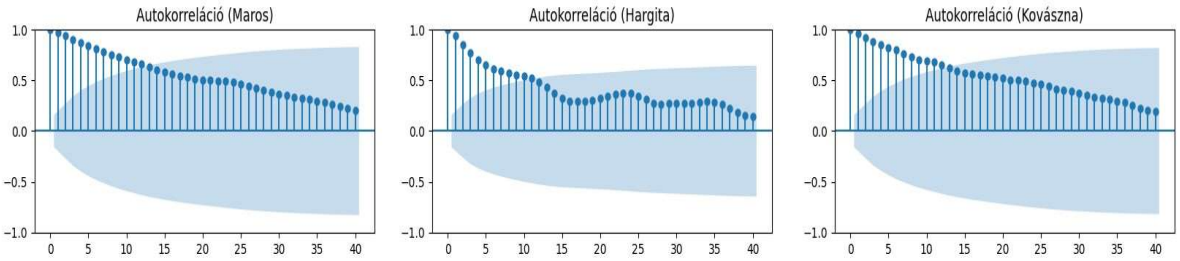
Az ACF segít azonosítani a mozóátlag (MA) folyamatot azáltal, hogy megmutatja, hány időegységnyi korreláció van az aktuális és az előző időpontok között, miközben figyelmen kívül hagyja a köztes időpontokat.

Ha egy stacionárius folyamat ACF-je teljesíti azt a feltételt, hogy

$$p_k = 0, \quad minden\ k > q\ esetén, és\ p_q \neq 0,$$ akkor a folyamat MA(q) folyamat.

(Sándor, 2019)

Az általam vizsgált idősorok autokorrelációs tesztjei a következőképpen néznek ki:



Elvégzett ACF tesztek a három megyére

Ha MA folyamat lenne, akkor az első néhány lépés után az autokorrelációk értékei hirtelen zuhannának, viszont a fent látható grafikonok nem ezt mutatják, hanem lineáris, fokozatos

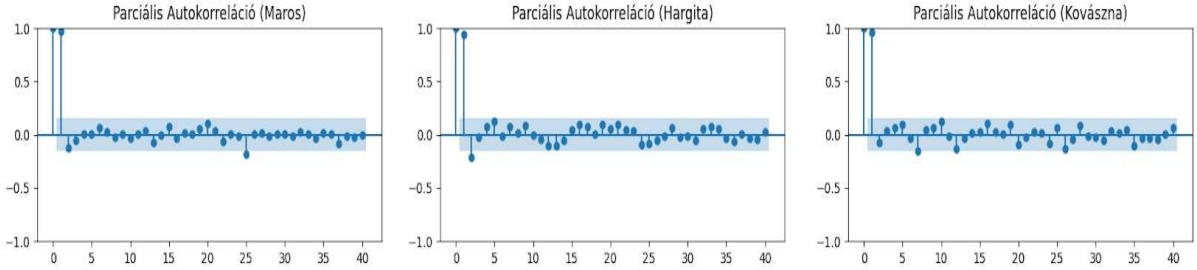
csökkenést, emiatt kizárható az, hogy MA(q) folyamatokról lenne szó bármelyik megye esetében is.

3.3.2 PACF

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

(Sándor, 2019)

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



Elvégzett PACF tesztek a három megyére

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

A grafikonok azt szemléltetik, hogy valószínűleg mindhárom megye esetében az AR(2) modellel érdemes próbálkozni az előrejelzéshez, viszont megnéztem még az ARMA(1, 1) és ARMA(1, 2) majd ezekből kiválasztottam a legkisebb AIC (Akaike Information Criterion) értékű modellt a Sándor (2019) alapján, mert valószínűleg ez a modell fog a legjobban illeszkedni az adott idősorra. Az AIC alapján a legjobban illeszkedő modell az, amely a lehető legnagyobb változatosságot magyarázza el a lehető legkevesebb független változó felhasználásával.

A következő táblázatban összefoglalom, hogy a különböző modellekre milyen AIC értékeket kaptam az egyes megyék esetében (a legkisebb AIC értékek celláit beszíneztem):

Modell	AIC		
	Maros	Hargita	Kovászna
AR (2)	1.59	114.34	124.09
ARMA (1,1)	2.15	116.51	123.70
ARMA (1,2)	3.81	113.73	124.86
ARIMA (2, 1, 0)	-4.31	108.78	115.03
ARIMA (1, 1, 1)	-4.31	108.91	116.46
ARIMA (1, 1, 2)	-2.42	108.55	116.49

A lefuttatott modell tesztek AIC eredményei

Úgy tűnik, Hargita megyénél az ARIMA(1, 1, 2) és Kovászna megyénél az ARIMA(2, 1, 0) modellt érdemes használni Maros megye esetében az AIC abszolútértékek alapján továbbra is az AR(2) modell a legjobb választás.

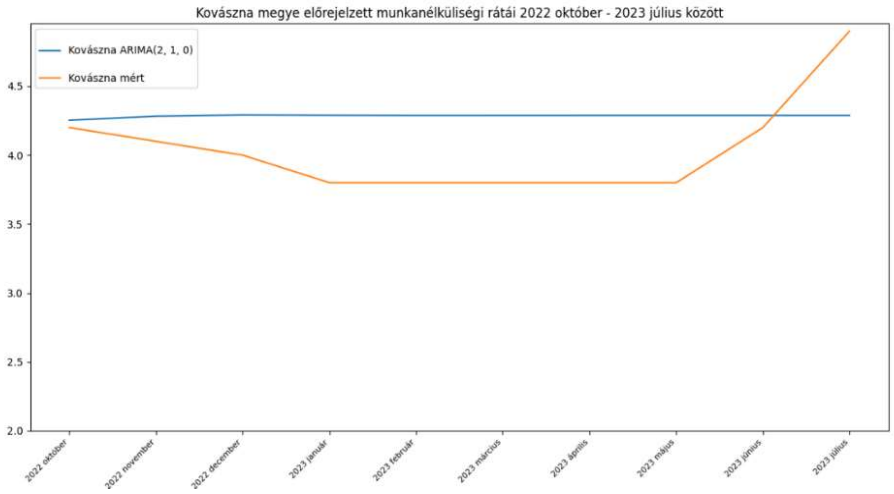
3.3 Előrejelzések és azok pontosságának meghatározása

A programomban a statsmodels.tsa.arima.model.ARIMA ingyenesen telepíthető csomag segítségével futtattam le a teszteket és az előrejelzéseket.

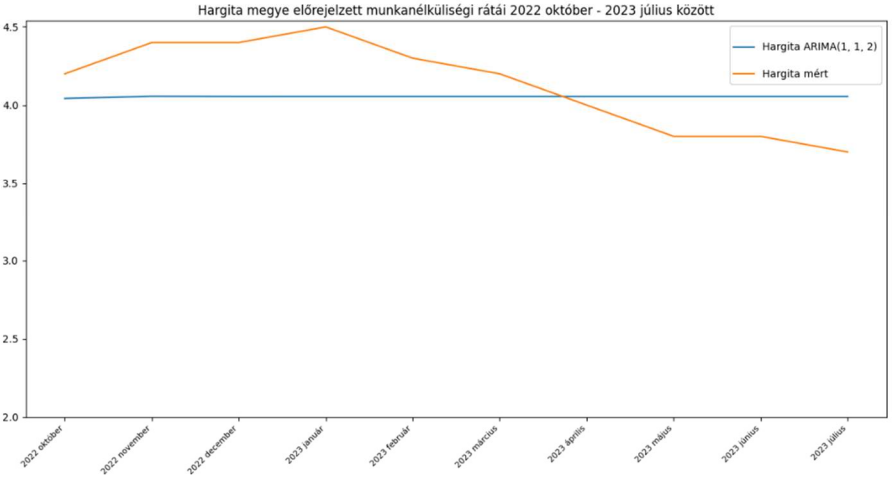
```
def ARIMA(self, p:int, d: int, q: int, t:int):
    p = int(p); q = int(q); d = int(d)
    idosor = self.adatok
    model = sm.tsa.ARIMA(idosor, order=(p, d, q))
    model_fit = model.fit()
    self.ARIMAbecslesek = model_fit.forecast(t)
    self.aic= model_fit.aic
    return ([model_fit.summary(), self.ARIMAbecslesek])
```

Az előrejelzéseket grafikonon ábrázoltam a matplotlib.pyplot csomag segítségével, amire egy általános függvényt írtam, ami több adatsort ábrázol és egy bájtfolyamban adja vissza a grafikont tartalmazó png formátumú képet (a html sablonban ezt könnyű megjeleníteni):

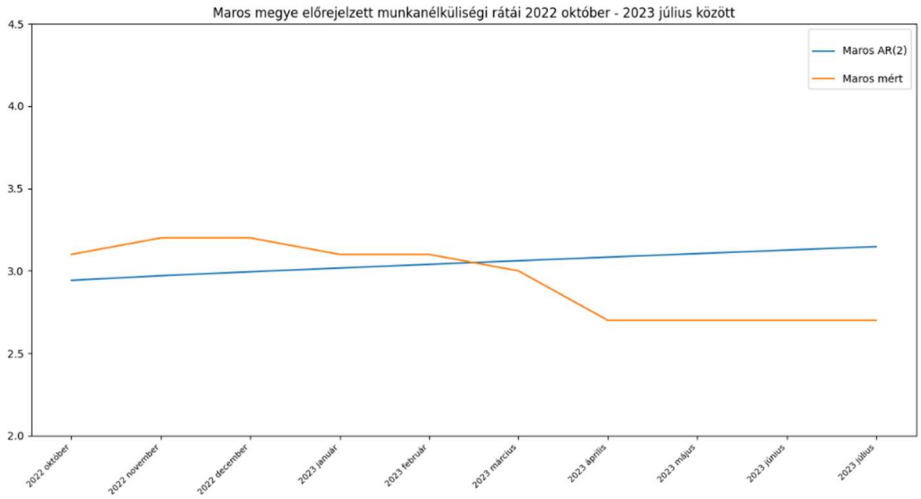
```
def AbrazolEgyben(adatsorok, idoszakok, megnevezesek, suruseg, Cim="",
yFelirat="", y_min=None, y_max=None, y_step=None, grid=False ):
    plt.figure(figsize=(15, 7))
    for i, megye in enumerate(megnevezesek):
        plt.plot(idoszakok, adatsorok[i], label=megye)
    plt.ylabel(yFelirat)
    plt.title(f"{Cim} {idoszakok[0]} - {idoszakok[-1]} között")
    plt.grid(grid)
    plt.xticks(idoszakok[::suruseg], rotation=45, ha="right", fontsize=8)
```



Kovászna megye ARIMA(2, 1, 0) modell előrejelzései (kékkek)



Hargita megye ARIMA(1, 1, 2) modell előrejelzései (kékkek)



Maros megye AR(2) modell előrejelzései (kékkek)

vázlat

vázlat

A becslések pontosságának megállapításához két mutatót használtam:

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

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

- **a relatív átlagos négyzetes** eltérések gyökét (Relative Root Mean Square Error, RRMSE), amely relatívan, normalizálva adja meg a hibákat, százalékos értékben. Az RRMSE az MSE-t normalizálja az aktuális értékek átlagával, és azok szórásával. Minél kisebb ez két mutató, a modell annál jobb becslést ad.

$$RRMSE = \frac{\sqrt{MSE}}{\underline{y}}$$

A képletekben n az adatok száma, y_i a valóságos, $\hat{y}_i$ a becsült érték az i-edik mintában. $\underline{y}$ a valós értékek átlaga. A következő táblázatban összefoglaltam az MSE és RRMSE értékeket az egyes modellekre:

Modell	MSE	RRMSE
Kovászna ARMA(2, 1, 0)	0.17	0.10
Hargita ARMA(1, 1, 2)	0.08	0.07
Maros AR(2)	0.08	0.07

MSE és RRMSE értékek

Mind a Kovászna, mind a Hargita megye modellje viszonylag alacsony RRMSE-t mutat, ami jó. Azonban a Maros megye modelljei az RRMSE alapján még jobban illeszkednek az idősorra.

A következő lépésben ismertetem a neurális hálózatok lényegét, majd megvizsgálom, hogy a mesterséges neuronhálón alapuló modellekkel milyen MSE és RRMSE mutatókat tudok elérni, vagyis melyik modell tud pontosabb becsléseket produkálni.

vázlat

4 Neurális hálózatok

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

Egy neuronháló általában háromféle rétegből áll:

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

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

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

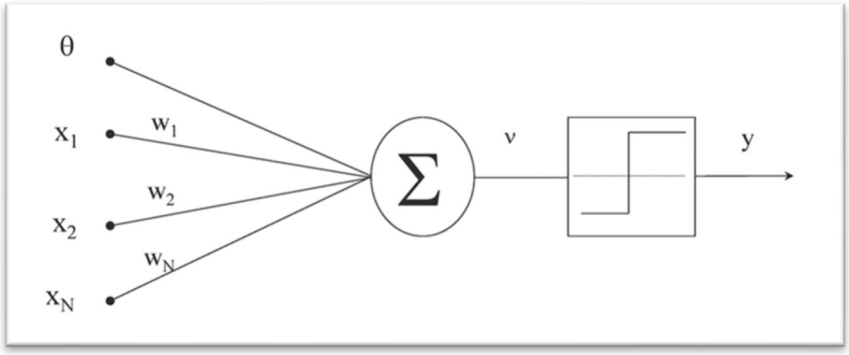
Tanulás során Az adatokból kapott visszacsatolás alapján a neuronhálók módosítják a paramétereiket (pl. eltolási értékek és súlyok). Rendszerint, a tanulási fázis lassú, több iterációt, sok sikertelen tanulási szakaszt hordozhat. A tanítás álltában korszakokra (epoch) van lebontva, és egy-egy korszak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

(Brassai, 2019)

vázlat

4.1 Neuronok

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



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

A neuronok a következő tényezőket használják:

- $x_1, x_2, \ldots x_i \ldots, x_N$: a neuron bemenetei, ezeket tartalmazza az $X = [x_1, x_2, \ldots x_i \ldots, x_N]$ vektor, ahol N a neuron bemeneteinek száma.
- A θ egy konstans bemenet, azaz az eltolási érték (bias), amely az érkező jelek súlyozott összegéhez hozzáadódik. Jellemzően a kimeneti rétegen kívül minden rétegnek van.
- w_i: az i-edik bemenethez tartozó súlytényező, ezeket a súlyokat tartalmazza a $W = [w_1, w_2 \ldots w_i \ldots w_N]$ vektor.

  A súlytényezők a lokális környezetben levő más neuronokkal való kapcsolatok irányát és erősségét reprezentálják. Ezen súlytényezőket kell finomhangolni a tanulás során.
- ϕ-vel (phi) jelöljük az aktivációs függvényt (általában nemlineáris transzferfüggvény)
- v-vel jelöljük a bemenetek súlyozott összegét, vagyis az ingert
- y neuron kimenete, más szóval válasz (activation).

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

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

(Brassai, 2019)

vázlat

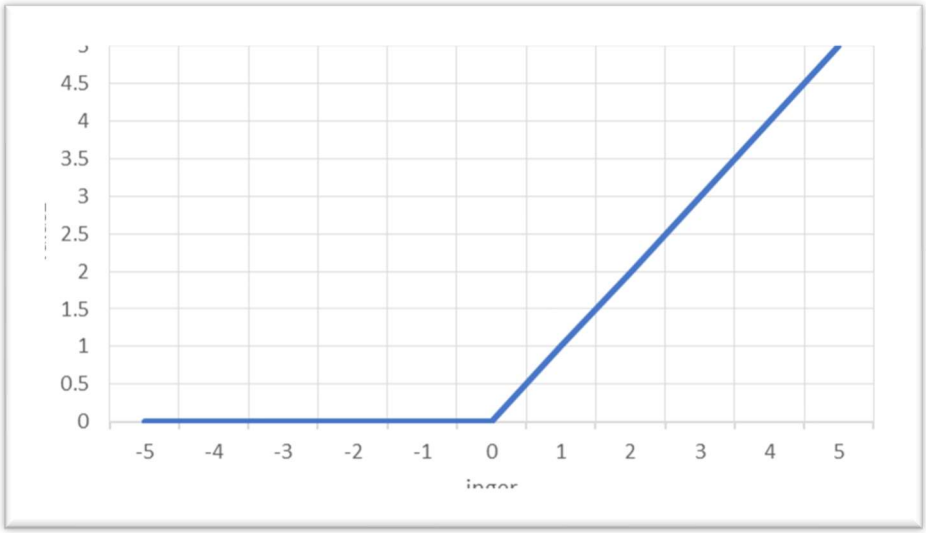
4.2 Aktivációs függvények

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

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

A ReLU (Rectified Linear Unit, azaz rektifikált lineáris egység): Egyszerűen a bemenetet adja vissza, ha az inger pozitív, és nullát, ha az inger negatív. Képlete:

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



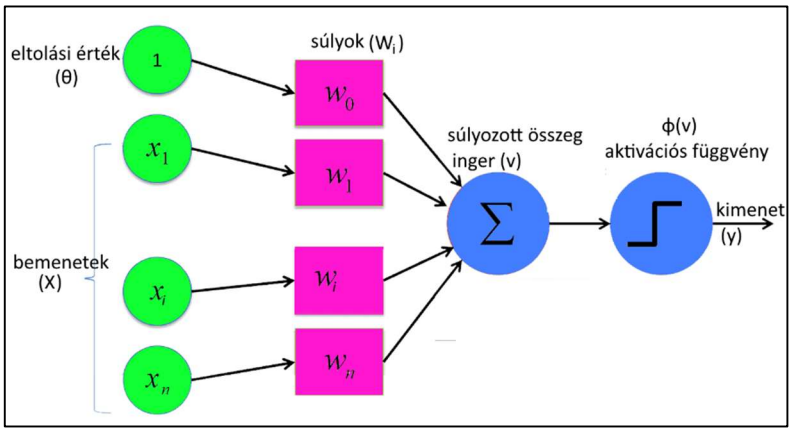
A ReLu (Rectified Linear Unit) aktivációs függvény grafikonja.
Forrás: saját ábra

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[2]. Sok rétegből álló (mély) neuronhálók esetében sokkal jobb, mint például a szigmoid vagy a tangens hiperbolikus függvény. (Brassai, 2019)

4.3 Perceptron, MLP

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



Egy perceptron felépítése
Forrás: saját ábra (Brassai, 2019) alapján, 46. oldal

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

(Brassai, 2019)

[2] A gradiens-elhalás probléma akkor fordul elő, amikor a gradiensek túl kicsik lesznek, és így a súlyokat nehéz vagy lassú frissíteni a tanulás során.

vázlat

4.4 Az MLP neurális hálózat tanítása

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

Én a felügyelt tanulás módszerét (supervised learning) alkalmaztam, amely során előre megadott és előkészített tanítóhalmazt adunk meg a modellnek, amely bemeneti adatokból (pl. korábbi megfigyelések), és az ahhoz tartozó elvárt kimeneti adatok (pl. megfigyelési értékek) párjaiból áll.

A tanítóhalmaz az MLPRegressor implementációjában két (akár többdimenziós) listából áll: x és y, ahol x a bemeneteket és y az elvárt kimeneteket tartalmazza. Összesen 163 megfigyelésem van mindegyik adatsorra, és ebből az utolsó 10-et jeleztem előre a hálókkal, tehát az utolsó 10 adat a teszthalmazban van, amit tanulás során nem lát a modell, csak a végső kimenetét értékeli vele. Bevált módszer, hogy választunk egy lag értéket, ami megadja azt, hogy egy kimenet hány korábbi bemenettől függ. Ilyenkor a neuronhálónak is ennyi bemeneti egysége lesz. Én 3 korábbi értéket választottam, mivel a tesztek alatt ez bizonyult a legjobbnak. Például a 2022 októberi megfigyelés, amit előre kell jeleznie a modellnek, azt a 2022 júliusi, augusztusi és szeptemberi értékből próbálja meghatározni. [2.7, 2.8, 2.9] → 3.1

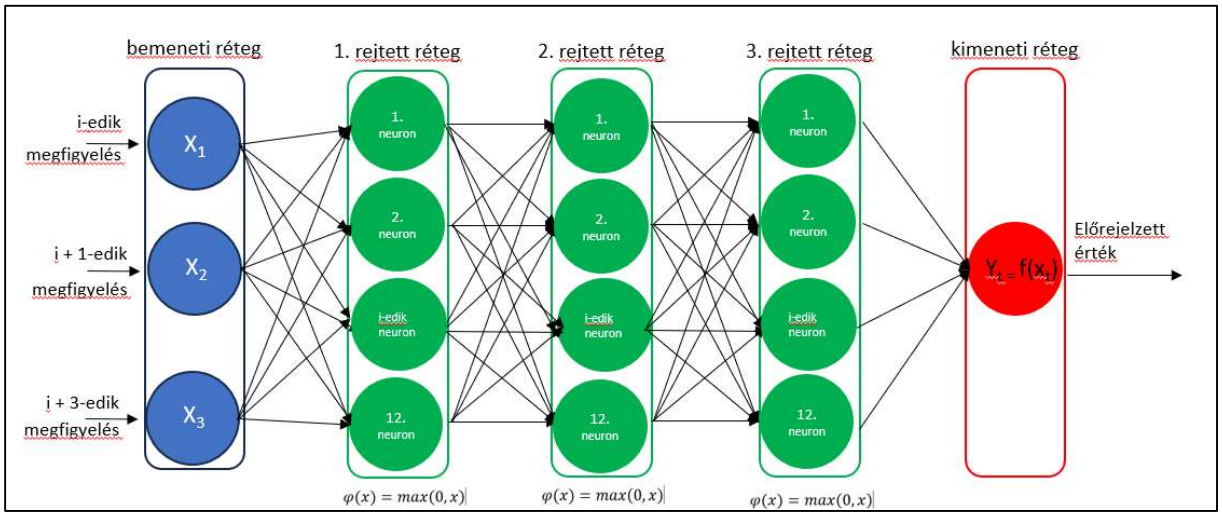
A háló az addigi ismeretei alapján kiszámítja, hogy az adott bemeneti adatból milyen kimeneti érték következik. Az úgynevezett hiba-visszaterjesztéses (back-propagation) algoritmus a tanító adatokon végig iterálva, a kapott kimeneti értékekből származó hiba alapján módosítja a hálózat súlyait úgy, hogy csökkentse a hibát. A végtelen ciklusok elkerülése érdekében korlátozott lépésszámban ismételjük a tanítást, ahelyett, hogy egy küszöbérték alá szeretnénk szorítani a hibát. Túl sok tanítási ciklus során könnyen előfordulhat, hogy a modellt túltanítjuk (overfitting). Ilyenkor a tanító halmazon a hiba csökken (egyre jobban illeszkedik rá), viszont a teszthalmaz elvárt eredményeitől egyre jobban távolodik. Ilyenkor le kell állítani a tanítási folyamatot, újra kell gondolni a háló paramétereit, tanítási algoritmusát. (Brassai, 2019)

vázlat

4.5 Az MLP modellek implementációja és előrejelzés

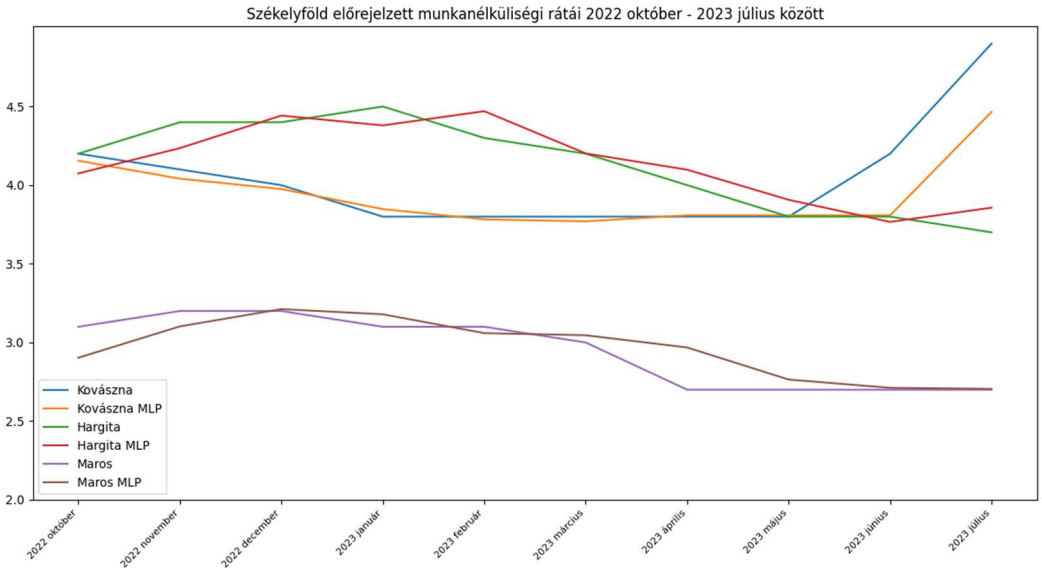
Az idősorokról tehát MLP hálókkal is készítettem előrejelzéset mindhárom megyére, azzal a céllal, hogy pontosabb értékeket kapjak, mint amit a Box-Jenkins módszer nyújtott. Az ARIMA modellek hátrányai közé tartozik, hogy egy csak egy bemeneti változóra lettek tervezve (univariate data), csak a lineáris kapcsolatokat tudják észlelni, valamint a feldolgozott idősor stacioner kell legyen. Ezzel szemben az MLP jóval ígéretesebbnek mutatkozik, a fentebb bemutatott tulajdonságainak köszönhetően. A nehézséget a megfelelő hálószerkezet és egyéb paramétereinek megtalálása jelenti. Ezt igyekeztem megkönnyíteni azzal, hogy a webalkalmazásomban egy külön felületet biztosítok az MLP modellek készítésére és teljesítményük szemléltetésére. A modellek szoftveres megvalósítására a Python kiegészítő csomagok között ingyenesen telepíthető sklearn.neural_network.MLPRegressor osztályt használtam.

Mindhárom megye esetében sok próbálgatás után 3 db, 12 neuronból álló rejtett réteget használtam, amelyek ReLU aktivációs függvénnyel dolgoznak. Mivel a tanítás során úgy jártam el, hogy 3 bemeneti adatból következik egy kimenet, ezért a bemeneti rétegnek 3 neuronja van. Az optimalizálási ciklus határa mindhárom idősor esetében 3000 lépés volt. A kimeneti réteg neuronja nem változtat az értéken (identitás függvény). Maros és Hargita megye esetében az SGD (Stochastic Gradient Descent) optimalizációs algoritmus volt a legjobb, míg Kovásznánál az LBFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno). A következő ábrán szemléltetem a neuronháló szerkezetét.



Az előrejelzésekhez használt MLP hálók szerkezete.
Forrás: saját ábra

vázlat



Az MLP neuronhálók által adott előrejelzések és a valós megfigyelések közös grafikonon ábrázolva
Forrás: saját ábra

Összehasonlításként következő táblázat tartalmazza az idősorok valódi megfigyeléseit, az ARIMA modellek és az MLP hálók becsléseit is az egyes időpontokra.

dátum	*Kovászna*			*Hargita*			Maros		
	mért	ARIMA	MLP	mért	ARIMA	MLP	mért	AR	MLP
2022 október	4.2	4.25	4.16	4.2	4.04	4.07	3.1	2.94	2.9
2022 november	4.1	4.28	4.04	4.4	4.06	4.24	3.2	2.97	3.1
2022 december	4	4.29	3.98	4.4	4.05	4.44	3.2	2.99	3.21
2023 január	3.8	4.29	3.85	4.5	4.06	4.38	3.1	3.02	3.18
2023 február	3.8	4.29	3.78	4.3	4.06	4.47	3.1	3.04	3.06
2023 március	3.8	4.29	3.77	4.2	4.06	4.2	3	3.06	3.05
2023 április	3.8	4.29	3.81	4	4.06	4.1	2.7	3.08	2.97
2023 május	3.8	4.29	3.81	3.8	4.06	3.91	2.7	3.1	2.76
2023 június	4.2	4.29	3.81	3.8	4.06	3.77	2.7	3.13	2.71
2023 július	4.9	4.29	4.47	3.7	4.06	3.86	2.7	3.15	2.71

Az ARIMA és az MLP neuronhálók teljesítményének összehasonlítása

Modell	MSE	RRMSE
Kovászna ARIMA(2, 1, 0)	0.17	0.1
Kovászna MLP (12, 12, 12, 12)	**0.04**	**0.05**
Hargita ARIMA(1, 1, 2)	0.08	0.07
Hargita MLP (12, 12, 12)	**0.01**	**0.03**
Maros AR(2)	0.08	0.1
Maros ARIMA(2,1,0)	0.05	0.07
Maros MLP (12, 12, 12)	**0.01**	**0.04**

A baloldalon levő táblázat pedig a modellek hibáit tartalmazza százalékos arányban.

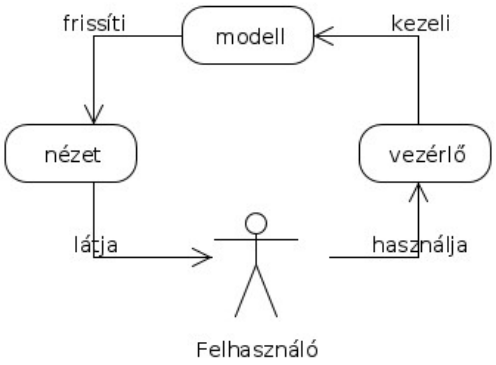
vázlat

vázlat

5 A Django webalkalamzás bemutatása

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

5.1 MVC



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

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

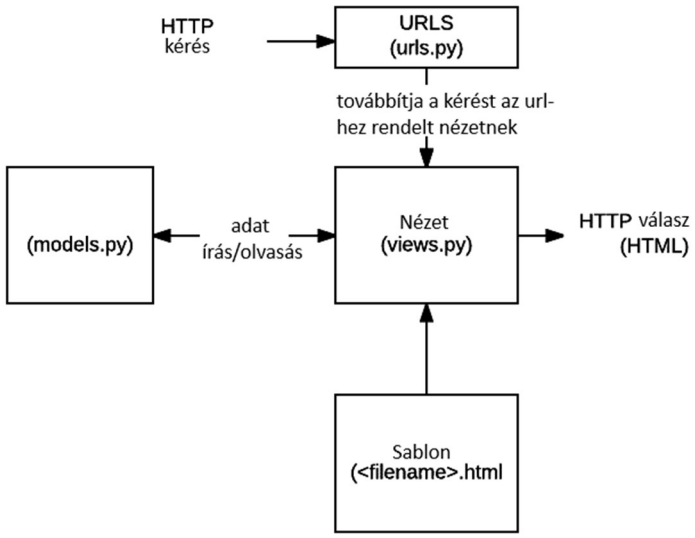
Az MVC minta alkalmazása segíthet javítani az alkalmazások karbantarthatóságát, kiterjeszthetőségét és tesztelhetőségét. Sok keretrendszer és fejlesztési környezet támogatja az MVC architektúrát, például a Ruby on Rails, a Django (Python), az ASP.NET, Laravel (PHP) és mások.

vázlat

5.2 Django

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

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



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

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

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

vázlat

Egy nézet dinamikusan létrehozhat egy HTML oldalt egy HTML sablon segítségével, adatokkal feltöltve azt a modellből.

(Foundation, developer.mozilla.org, 2024)

5.3 Beolvasás Excel-ből

A beolvasásra egy HTML űrlapot készítettem, ez a projekt kezdőlapja is egyben. Egyelőre csak Excel fájlokat fogad el (.xls/.xlsx), mivel az adatsorokat rendkívül egyszerűen lehet előkészíteni Excelben.

Fontos, hogy a táblázatok a következőképpen legyenek elkészítve:

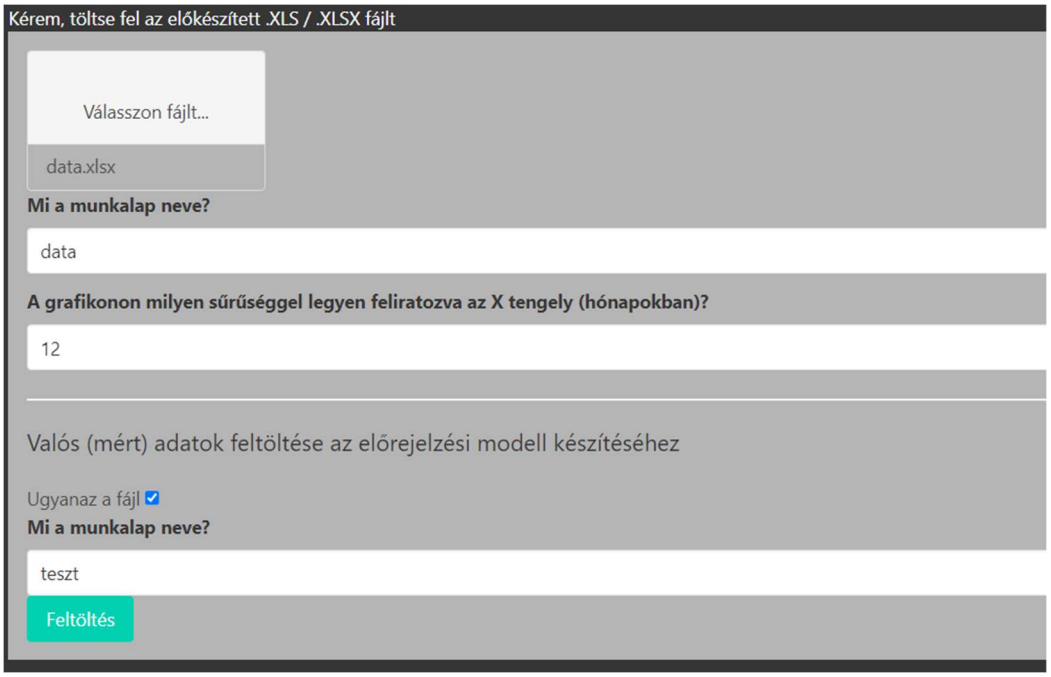
	A	B	C	D	E
1	idoszak	Kovászna	Hargita	Maros	
2	2010 január	12.2	10.9	8.3	
3	2010 február	12.5	11.4	8.4	
4	2010 március	12.3	11.2	8.5	
5	2010 április	11.9	10.7	8.2	

A z idpősorokat tartalmató Excel munkalap (példa)
Forrás: saját ábra

Az első sor fejléc kell legyen, ami tartalmazza az idősor(ok) megnevezését, az én esetemben a megyék nevét. Természetesen nem kötelező több idősor megléte. Az első oszlop a megfigyelésekhez tartozó időpontokat kell tartalmazza, ebből készülnek a grafikonok x tengelyen levő feliratai is. A többi sor oszlop pedig a megfigyelt értékeket tartalmazzák. Fontos persze az is, hogy időrendi sorrendben legyenek a megfigyelések. Az én esetemben például a „data.xslx„ fájl „data” nevű munkalapja tartalmazza a 12 év (2010 január – 2022 szeptember) megfigyeléseit a fenti formában, növekvő időrendi sorrendben. A „teszt” nevű munkalap pedig 2022 október és 2023 július közötti megfigyeléseket tartalmazza ugyanezekre a megyékre.

Lehetőség van két különböző fájlt is feltölteni, hogyha a teszt adatok - vagyis amelyeket nem vesz figyelembe a program az előrejelzési modellhez, hanem segítségével az előrejelzés pontosságát határozza meg – külön fájlban vannak. Ha ugyanabban a fájlban van a két munkalap, akkor be kell jelölni az „ugyanaz a fájl” feliratú jelölőnégyzetet, és akkor nem kell ugyanazt a fájlt kétszer feltölteni. Ebben az esetben JavaScript kóddal elrejtem a második fájlfeltöltő bemenetet.

vázlat



Mindkét esetben meg kell adni a munkalapok nevét, mivel a pandas modullal beolvasott többdimenziós adatszerkezet (DataFrame) asszociatív listákban tárolja az oszlopokat, kulcsai a munkalapok nevei. Rossz munkalap név esetén visszairányít minket a kezdőlapra. Ezenkívül lehetőség van megadni azt, hogy az adatsorból generált grafikon x tengelye milyen sűrűséggel legyen feliratozva. Például jelen esetben a több, mint 140 megfigyelésre bőven elég évente, vagy félévente megjeleníteni az időszakot.

A Feltöltésre kattintva létrejön egy POST kérés (ami tartalmazza a feltöltött fájlokat és a munkalapok neveit.) majd meghívódik az „upload” nevű URL, amihez az ugyanilyen nevű függvény van rendelve a views.py fájlban.

A függvény hiányzó paraméterek esetén hibaüzenettel visszatéríti a feltöltő oldalra a felhasználót. Különben eltárolom a megkapott adatokat a POST kérésből. Megnézem, hogy be lett-e jelölve a jelölőnégyzet, mert ha igen, akkor ugyanabból a fájlból vesszük a teszt periódus munkalapját, különben ott is leellenőrzöm, hogy fel lett-e töltve a másik fájl.

Ezután a pandas csomag read_excel függvényével eltárolom egy-egy DataFrame szerkezetbe a táblázatokat. Továbbá külön listákba eltárolom az időszakokat, az idősorok neveit (pl. megyék neveit), és persze a megfigyeléseket (többdimenziós lista, a kulcsok az idősorok nevei), azért, hogy könnyebben fel tudjam dolgozni az adatokat a diagram készítésekor és a weboldalon való megjelenítéskor.

vázlat

A nézet számára egy könnyen feldolgozható listába teszem a beolvasott adatokat, legenerálom az adatsorok diagramját, létrehozom a Stat objektumokat (amelyek egy-egy idősor mutatóit, adattagjait tartalmazzák), végül megcsinálom a nézetet az adatokkal és elküldöm a felhasználónak.

A render függvény jeleníti meg a html sablont, amely megkapja a szükséges változókat és adatszerkezeteket, tehát a beolvasás után egyből megjelennek az elemzések a felhasználónak.

```
def upload(request):
    if 'file' not in request.FILES or 'suruseg' not in request.POST or 'sheet'
not in request.POST:
        messages.error(request, 'Hiányzó paraméter(ek) (sűrűség/munkalap
nevek)!')
        return redirect('home')

    uploaded_file = request.FILES['file']
    suruseg = int(request.POST['suruseg'])
    sheetName = request.POST['sheet']

    adatsorok, adatsorNevek, idoPontok, teszt_adatok = [], [], [], None

    if 'sameFile' in request.POST:
        teszt_adatok = request.FILES['file']

    elif 'file_teszt' not in request.FILES:
        messages.error(request, 'Nem lett adatforrás fájl feltöltve az
előrjelzésekhez!')
        return redirect('home')

    else:
        teszt_adatok = request.FILES['file_teszt']

    tesztSheetName = request.POST['tesztSheetName']

    try:
        df_teszt = pd.read_excel(teszt_adatok, sheet_name=tesztSheetName)
        global beolvasott_teszt_idoszakok
        beolvasott_teszt_idoszakok = df_teszt[df_teszt.columns[0]].tolist()

        df = pd.read_excel(uploaded_file, sheet_name=sheetName)
        fejlec = df.columns.tolist()
        idoPontok = df[fejlec[0]].tolist()

        for i, col in enumerate(fejlec[1:]):
            adatsorNevek.append(col)
            adatsorok.append(df[col].tolist())
```

```
        data_rows = [{'idoPont': ido, 'adatsorok': [adatsor[i] for adatsor in
adatsorok]} for i, ido in enumerate(idoPontok)]

        diagram = AbrazolEgyben(adatsorok, idoPontok, adatsorNevek, suruseg,
"Székelyföld munkanélküliségi rátái", "")
        diagram = base64.b64encode(diagram.read()).decode('utf-8')

        global statisztikak
        statisztikak = createStatObjects(adatsorNevek, adatsorok, idoPontok)

        teszt_adatok_df = pd.read_excel(teszt_adatok,
sheet_name=tesztSheetName)
        for i in statisztikak:
            for j in fejlec:
                if i.megye_nev == j:
                    i.setTesztAdatok(teszt_adatok_df[j].tolist())
                    i.setTesztIdoszakok(beolvasott_teszt_idoszakok)

        return render(request, 'upload.html', {'data_rows': data_rows,
'adatsorNevek': adatsorNevek, 'statisztikak': statisztikak, 'diagram':
diagram})

    except pd.errors.ParserError:
        print(traceback.format_exc())
        return HttpResponse("Helytelen fájl!", status=400)
```

6 Következtetések

5. Irodalomjegyzék

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

Bamberger, S., Heckel, R., & Krahmer, F. (2023. 08 5). *Approximating Positive Homogeneous Functions with Scale Invariant Neural Networks.* doi:https://doi.org/10.48550/arXiv.2308.02836

Brassai, S. T. (2019). *Neurális hálózatok és Fuzzy logika.* Kolozsvár: Scientia Kiadó. Forrás: http://real.mtak.hu/id/eprint/122603

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

Foundation, P. S. (2024. 02 01). *developer.mozilla.org.* Forrás: Django introduction: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

Foundation, P. S. (2024. 02 01). *General Python FAQ.* Forrás: python.org: https://docs.python.org/3/faq/general.html#what-is-python-good-for

Georgeta, E. S. (2015). *The economic and social situation in Romania.* Brussel, Belgium: European Economic and Social Committee. doi:10.2864/484519

Josef, P., Skipper, S., Taylor, J., & statsmodels-developers. (2024). Forrás: statmodels.org: https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html

Lal, P., & Jose, J. (2013). Application of ARIMA(1,1,0) Model for Predicting Time Delay of Search Engine Crawlers. *Informatica Economică, 17*(4), 26-38. doi:DOI: 10.12948/issn14531305/17.4.2013.03

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 Brașov and Harghita counties. *Közgazdász fórum, 21*(135), 66-79. Letöltés dátuma: 2024. február

Sándor, Z. (2019). *Bevezetés az ökonometriába.* Sepsiszentgyörgy: T3 Kiadó.

Sándor, Z., & Tánczos, L. (2019). *Gazdasági statisztika jegyzet.* Sepsiszentgyörgy: T3 kiadó.

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