fedlap vázlat

**A SZÉKELYFÖLDI MEGYÉK MUNKANÉLKÜLISÉGI RÁTÁINAK ELŐREJELZÉSE BOX-JENKINS ELJÁRÁS ÉS A MESTERSÉGES NEURONHÁLÓ-ALAPÚ MODELLEK ALAPJÁN.   
  
DJANGO WEBALKALMAZÁS A KUTATÁS ELŐSEGíTÉSÉHEZ.**

Tartalom

Tartalom

[1. Bevezetés 2](#_Toc158478875)

[1.1 Szakirodalmi áttekintés 3](#_Toc158478876)

[1.2 A munkanélküliségi ráta fogalmának meghatározása Romániában 3](#_Toc158478877)

[1.3 Felhasznált statisztikai mutatók és fogalmak 4](#_Toc158478878)

[2 Az idősorok elemzése 6](#_Toc158478879)

[3 Előrejelzés ARMA módszerrel 8](#_Toc158478880)

[3.1 A stacionaritás vizsgálata 8](#_Toc158478881)

[3.2 Autoregresszív és mozgóátlag modellek (AR, MA, ARMA, ARIMA) 9](#_Toc158478882)

[3.3 Autokorrelációs és parciális autokorrelációs tesztek 10](#_Toc158478883)

[3.3.1 ACF 10](#_Toc158478884)

[3.3.2 PACF 11](#_Toc158478885)

[3.3 Előrejelzés pontosságának meghatározása a hibatesztek alapján 13](#_Toc158478886)

[4 Neurális hálózatok 16](#_Toc158478887)

[4.1 Neuronok 17](#_Toc158478888)

[5 A Django webalkalamzás bemutatása 18](#_Toc158478889)

[5.1 Python 18](#_Toc158478890)

[5.2 MVC 18](#_Toc158478891)

[5.3 Django 20](#_Toc158478892)

[5.4 Beolvasás Excel-ből 21](#_Toc158478893)

[5.5 Diagramok készítése 24](#_Toc158478894)

[5.6 Mutatók számítása, tesztek futtatása az idősorokon (modell) 26](#_Toc158478895)

[5.7 ARIMA előrejelzések 27](#_Toc158478896)

[6 Következtetések 32](#_Toc158478897)

[5. Irodalomjegyzék 33](#_Toc158478898)

# 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 keretrendszerben írt 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.

# Szakirodalmi áttekintés

(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 becséleseket.

(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.

## 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]](#footnote-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.

## Felhasznált statisztikai mutatók és fogalmak

**Átlag**: Jelöljünk n db megfigyelést x1, x2, ..., xn -nel. A megfigyelések összegét elosztjuk a megfigyelések számával.

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

**A variancia (szórásnégyzet):**

**Medián:** Jelöljük x1, x2, ..., xn –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) ≤ x(2)  ≤ ⋯ ≤ x(n)

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

Ha n páros, akkor 2 középső megfigyelés van, n/2 és (n / 2) + 1, tehát a medián:

(Sándor & Tánczos, 2019)

## 1.4 Idősorok előrejelzése

# Az idősorok elemzése

A webalkalmazásomba feltöltöttem a 2010 január – 2022 szeptember közötti munkanélküliségi ráta adatsorokat a három megyére. A feldolgozás után a program a következő grafikont készítette el:



*A regionális munkanélküliségi ráták grafikonja*

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)

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 | *3.0%, 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 | 5.86% | 5.10% | 4.47% |
| őszi átlag | 5.72% | 5.28% | 4.55% |

1. *Táblázat: Az idősorok statisztikai mutatói*

A következő lépésekben megvizsgálom ADF és KPSS teszt segítségével az idősorok stacionaritását, ismertetem és a program segítségével elvégeztem az autokorrelációs és parciális autokorrelációs teszteket, ezek segítségével megállapítom azokat az AR/MA/ARMA/ARIMA modelleket, amelyek a legjobban illeszkednek a megyék idősoraira, ezekkel pedig előrejelzéseket készítek.

# Előrejelzés ARMA módszerrel

## 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 előrejelzhetők. 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 (βt), véletlenszerű séta (walk, rt) és stacionárius hibára (εt).

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

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

*2.. táblázat: ADF és KPSS tesztek*

|  | **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 |

Látszólag az idősorok nem stacionáriusok, mindenképpen figyelembe kell venni az esetleges szezonalitás jelenlétét, ezért érdemes differenciálni az adatsorokat az előrejelzés során.  
Az idősorokban kimutatható szezonalitások véleményem szerint valószínűleg arra vezethetőek vissza, hogy ahogy az 1. táblázatban szemléltettem, a téli hónapokban összességében átlagosan magasabb munkanélküliségi rátákat mértek az egész régióban.

## 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:

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

Egy AR(p) folyamat akkor stacionárius, ha:

* Az autoregresszív 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:

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

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

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

Egy ARMA folyamat akkor stacionárius, ha az AR komponense stacionárius.

Az ARIMA(p, d, q) modellekben az I (integrated) azt jelenti, hogy az idősort a szezonmentesítés érdekében d alkalommal differenciáljuk, vagyis integráljuk.

(Sándor, 2019)

## 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.

## ACF

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

(Sándor, 2019)

Az általam vizsgált idősorok autokorrelációs tesztjei a következőképpen néznek ki:



*Elvégzett ACF tesztek a három megyére*

Ha AR 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.

## PACF

Egy AR(1) folyamat esetében a pk autokorrelációkfolyamatosan (exponenciálisan) csökkennek. Általában minden AR(p) folyamatnak megvan ez a tulajdonsága, viszont p > 1 rendű folyamatok esetében a csökkenés nem feltétlenül monoton. 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) eseteket is, majd ezekből kiválasztottam a legkisebb AIC[[2]](#footnote-2) (Akaike Information Criterion) értékű modellt a Sándor (2019) alapján, mert valószínűleg ez a modell fog a legjobban illeszkedni az adott idősorra. A következő táblázatban összefoglalom, hogy a különböző modellekre milyen AIC értékeket kaptunk az egyes megyék esetében (a legkisebb AIC-k celláit beszíneztem):

|  |  |  |  |
| --- | --- | --- | --- |
| Modell | AIC | | |
| Maros | Hargita | Kovászna |
| AR (1) | 9.65 | 128.96 | 123.91 |
| 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 |

*A lefuttatott modell tesztek eredményei*

Azt látjuk, hogy az előrejelzéseknél Maros megyénél AR(2), viszont Hargitánál ARMA(1, 2), míg Kovászna megyénél ARMA(1, 1) modellt érdemes használni, nem az AR(2)-t.

Viszont a stacionaritás vizsgálata azt sugallja, hogy érdemes differenciálni az idősorokat, mivel valamiféle trend lehet bennük. Ezért megnéztem ugyanezeket a modelleket, egyszer integrált, vagyis ARIMA(2, 1, 0), ARIMA(1, 1, 1), ARIMA (1, 1 ,2) formában is:

|  |  |  |  |
| --- | --- | --- | --- |
| Modell | AIC | | |
| Maros | Hargita | Kovászna |
| 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 |

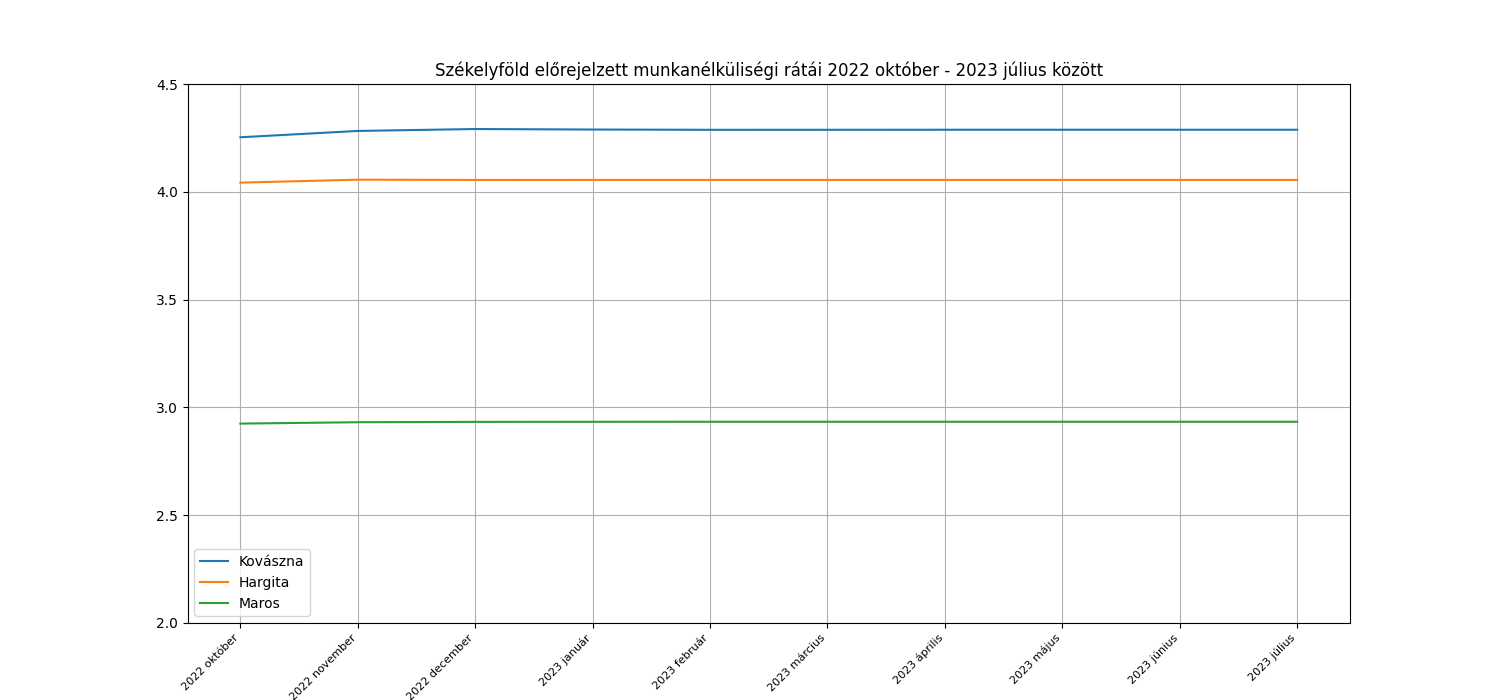
Úgy tűnik, Hargita megyénél az ARIMA(1, 1, 2) és Kovászna megyénél az ARIMA(2, 1, 0) modellt érdemes használni (ezeket az RRMSE értékek is igazolták). Maros megye esetében az AIC abszolútértékek alapján továbbra is az AR(2) modell a legjobb választás.

## 3.3 Előrejelzés pontosságának meghatározása a hibatesztek alapján

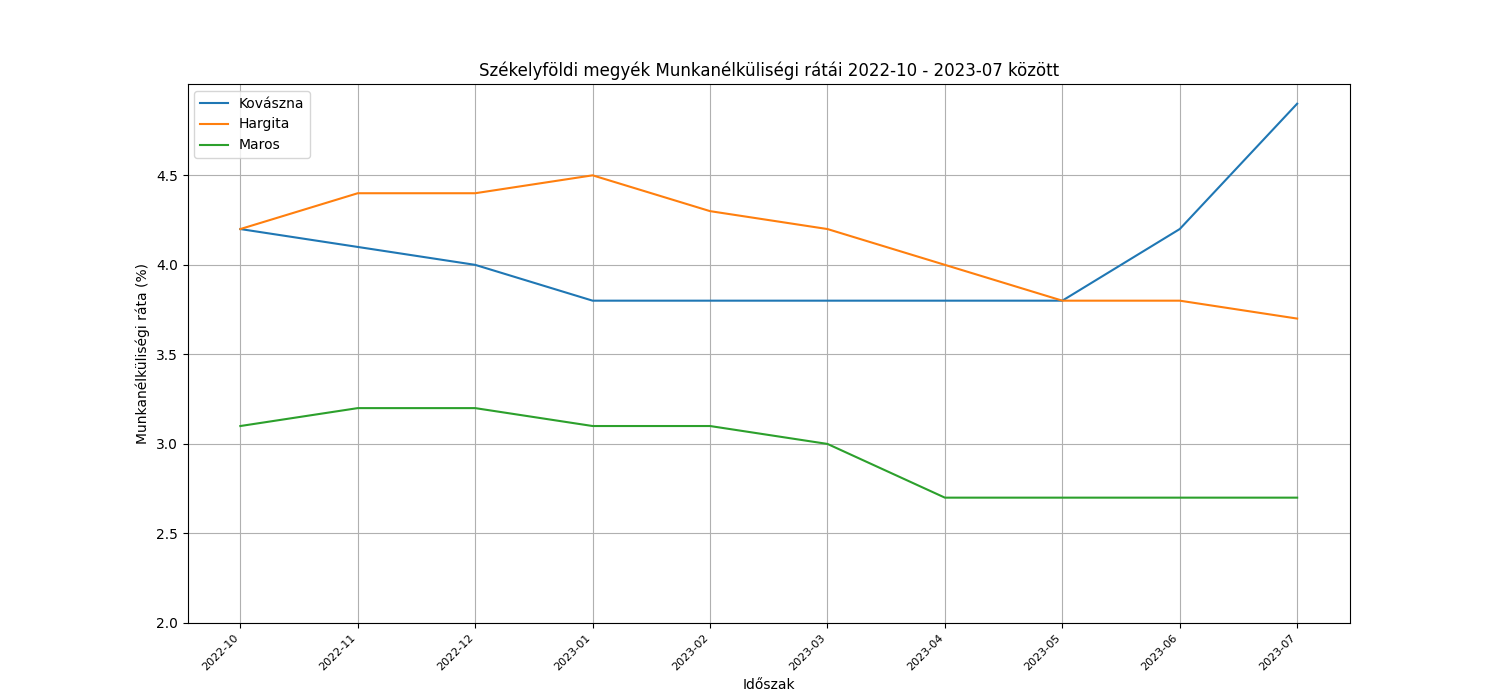
Az adatforrásból szándékosan meghagytam a legfrissebb 10 bejegyzést, hogy az egyes modellek becsléseinek hibáit meg tudjam állapítani a valódi adatokhoz képest.

Miután a programon keresztül kértem a kiválasztott modellekre a 10 hónapos előrejelzéseket, a következő eredményt kaptam:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| dátum | *Kovászna* | | *Hargita* | | Maros | |
| mért | ARIMA(2, 1, 0) | mért | ARIMA(1, 1, 2) | mért | AR(2) |
| 2022 október | *4.2* | 4.25 | 4.2 | 4.04 | 3.1 | 2.94 |
| 2022 november | *4.1* | 4.28 | 4.4 | 4.06 | 3.2 | 2.97 |
| 2022 december | 4 | 4.29 | 4.4 | 4.05 | 3.2 | 2.99 |
| 2023 január | 3.8 | 4.29 | 4.5 | 4.06 | 3.1 | 3.02 |
| 2023 február | 3.8 | 4.29 | 4.3 | 4.06 | 3.1 | 3.04 |
| 2023 március | 3.8 | 4.29 | 4.2 | 4.06 | 3 | 3.06 |
| 2023 április | 3.8 | 4.29 | 4 | 4.06 | 2.7 | 3.08 |
| 2023 május | 3.8 | 4.29 | 3.8 | 4.06 | 2.7 | 3.1 |
| 2023 június | 4.2 | 4.29 | 3.8 | 4.06 | 2.7 | 3.13 |
| 2023 július | 4.9 | 4.29 | 3.7 | 4.06 | 2.7 | 3.15 |



*13. ábra: ARMA előrejelzések grafikonon ábrázolva*



*14. ábra: A valós adatok grafikonon ábrázolva*

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:
* **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.

|  |  |  |
| --- | --- | --- |
| 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 |
| Maros AR(2, 1, 0) | 0.05 | 0.07 |

A képletekben n az adatok száma, yi a valóságos, a becsült érték az i-edik mintában. a valós értékek átlaga. A következő táblázatban összefoglaltam az MSE és RRMSE értékeket az egyes modellekre:

*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 megvizsgálom, a mesterséges neuronhálón alapuló modellekkel milyen MSE és RRMSE mutatókat tudok elérni, vagyis melyik modell tud pontosabb becsléseket produkálni.

# 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. Általában rendelkeznek egy tanulási algoritmussal, amely meghatározza az adatfeldolgozás módját, valamint egy előhívási (recall) algoritmussal, amely a megtanult információ felhasználását teszi lehetővé a modell számára.

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 réteg(ek): Ezek az a réteg(ek), 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 az adatokból kapott visszacsatolás alapján finomítják a paramétereiket. Ez a folyamat a "tanulás" része, és lehetővé teszi számukra, hogy javítsák a teljesítményüket az adott feladaton.

## 4.1 Neuronok

Egy neuron olyan feldolgozó elem, amely több bemenetet fogad és egy kimenetet generál. Rendelkezhet lokális memóriával, amelyben tárolhat bemeneti vagy kimeneti értékeket. Az aktuális kimeneti értéket általában nemlineáris transzferfüggvény segítségével hozza létre, amit aktiváló függvénynek is nevezünk. Ez az aktiváló függvény segít abban, hogy a neuron válaszoljon a bemenetekre a megfelelő módon, és lehetővé teszi a hálózat számára a komplexebb mintázatok és jelenségek felismerését és reprezentálását.



Egy általános neuron szerkezete

A neuronok a következő elemekből állnak:

* *x1, x2 ..xi.. xN*: a neuron bemenetei, az ezeket tartalmazza az X = [*x1, x2 ..xi.. xN*] vektor, ahol N a neuron bemeneteinek száma
* aegy konstans bemenet, vagyis az eltolási érték (bias)
* *wi*: az i-edik bemenethez tartozó súlytényező, ezeket a súlyokat tartalmazza a   
  W = [*w1, w2 ..wi.. wN*] vektor.

A súlytényezők a többi neuronnal való kapcsolatok irányát és erősségét reprezentálják.

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

Az aktivációs függvények nemlineárisak és differenciálhatóak kell legyenek. A kiemenetük (aktivációjuk) általában +1 körül van, ha az ingert helyesnek, 0 körül, ha rossznak tartják. A transzferfüggvényeknek jellemzően van egy küszöbértékük, amelyet az eltolási értékkel lehet állítani.  
A szigmoid (logisztikus) függvény például nagyon jó erre a célra.

# A Django webalkalamzás bemutatása

## Python

A Python egy magas szintű, objektum-orientált, általános célú programozási nyelv, amely számos különböző problémaosztályra alkalmazható.A nyelvhez tartozik egy nagy szabványos könyvtár, amely olyan területeket fed le, mint a karakterlánc-feldolgozás, az internetes protokollok (pl. HTTP, FTP), szoftvermérnöki csomagok (pl. egységtesztelés), naplózás, profilozás, Python-kód elemzése és operációs rendszer interfészei (rendszerhívások). (Foundation, General Python FAQ, 2024)

## 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.

## Django

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

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



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

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

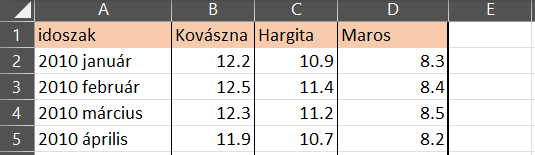
Modellek: A modellek olyan Python objektumok, amelyek meghatározzák egy alkalmazás adatstruktúráját, és mechanizmusokat biztosítanak a rekordok kezeléséhez (hozzáadás, módosítás, törlés) és lekérdezéséhez az adatbázisban.  
Sablonok: Egy sablon egy szöveges fájl, amely meghatározza egy fájl struktúráját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

(Foundation, developer.mozilla.org, 2024)

## 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:



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 sorrenben 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.

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 mutatja ki – külön fájlban vannak. Ha ugyanabban a fájlban van a két munkalap, akkor be kell jelölni az „ugyanaz a fájl” feliratú jelülőnégyzetet, és akkor nem kell ugyanazt a fájlt kétszer feltölteni. Ebben az esetben JavaScript kóddal elrejtem a második fájlfeltöltő bemenetet.



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. Ezenkívül lehetőség van megadni azt, hogy az adatsorból generált grafikon x tengelye milyen sűrűséggel legyen feliratozva.

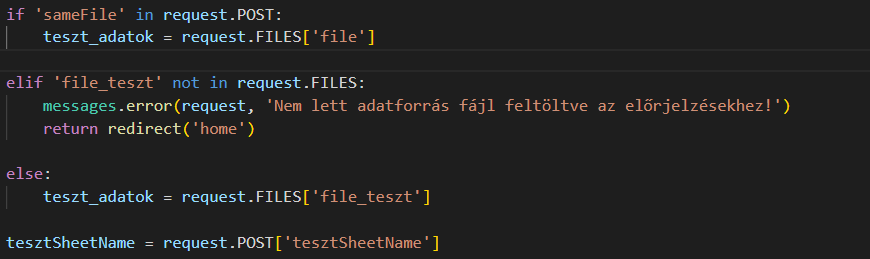
A Feltöltésre kattintva POST kéréssel (ami tartalmazza a feltöltött fájlt, stb.) meghívódik az „upload” nevű URL, amihez az ugyanilyen nevű függvény van rendelve a views.py fájlban.

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')

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.

try:

        df\_teszt = pd.read\_excel(teszt\_adatok, sheet\_name=tesztSheetName)

        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())

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.

statisztikak = createStatObjects(adatsorNevek, adatsorok, idoPontok)

return render(request, 'showData.html',   
{  
 'data\_rows': data\_rows,  
 'adatsorNevek': adatsorNevek,   
 'statisztikak': statisztikak,   
 'diagram': diagram  
})

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.

## 5.5 Diagramok készítése

Mivel többször is kellett grafikont előállítanom különböző adatsorokkal, készítettem egy függvényt (AbrazolEgyben), ami visszaad a bemeneti adatokból egy bájtfolyamot (IO.Bytes), ami a grafikon png formátumú képét tartalmazza. Feltöltés során az upload függvény tehát meghívja ezt a függvényt is, elmenti azt png képként és továbbadja a showData.html nézetnek. A függvény a következőképpen működik:

def AbrazolEgyben(adatsorok, idoszakok, megnevezesek, suruseg, Cim="", yFelirat="", y\_min=None, y\_max=None, y\_step=None):

    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(True)

    try:

        plt.xticks(idoszakok[::suruseg], rotation=45, ha="right", fontsize=8)

    except Exception as e:

        print(f"Error: {e}")

    if all((y\_min, y\_max, y\_step)):

        plt.yticks(np.arange(y\_min, y\_max, y\_step))

    plt.legend()

    buffer = io.BytesIO()

    plt.savefig(buffer, format="png")

    buffer.seek(0)

    return buffer

Az adatsorok nevű paraméter egy többdimenziós lista, a beolvasott adatsort vagy adatsorokat tartalmazza, az én esetemben a három megye megfigyeléseit. Az idoszakok egy lista, amely a megfigyelésekhez tartozó időpontokat tartalmazza (x tengely feliratai). A megnevezesek az idősorok neveit tartalamzó lista, az én esetemben például a három megye neve. Ezenkívül a grafikonnak lehet opcionális címet adni, y tengely feliratot, léptéket, minimum és maximum értéket. A beolvasást követően így ábrázolom az adatokat a függvény segítségével:

diagram = AbrazolEgyben(adatsorok, idoPontok, adatsorNevek, suruseg, "Székelyföld munkanélküliségi rátái")

diagram = base64.b64encode(diagram.read()).decode('utf-8')

Ezt a diagram nevű változót átadva a nézetnek, meg lehet jeleníteni egy img tag-ben:

 return render(request, 'showData.html', {'data\_rows': data\_rows, 'adatsorNevek': adatsorNevek, 'statisztikak': statisztikak, 'diagram': diagram})

A showData.html kódja:

 <div id="diagram" class="has-text-centered">

     <img src="data:image/png;base64,{{ diagram }}" alt="Diagram" style=" height: 30%;">

</div>

## Mutatók számítása, tesztek futtatása az idősorokon (modell)

Mivel a Python támogatja az osztályokat, ezért hasznosnak tartottam az adatsorokat objektumokként kezelni, saját adattagokkal és függvényekkel.  
Az osztályokat a models.py fájlban kell létrehozni.  
Például, minden megye egy „Stat” osztály példány, az idősor neve, a megfigyelések listája, azok átlaga, stb. mind-mind adattag. A számítások külön függvényekbe kerültek, hogy átláthatóbb legyen a kód. A beolvasás során már létre is jönnek a Stat példányok, és eltárolódnak egy listába.

class Stat :

    def \_\_init\_\_(self, megye\_nev, adatok, idoPontok):

        self.megye\_nev = megye\_nev

        self.adatok = adatok

        self.idoszakok = idoPontok

        self.atlag = round(np.mean(adatok), 2)

        self.szoras = round(np.std(adatok), 2)

        self.variancia = round(np.var(adatok), 2)

        self.median = round(np.median(adatok), 2)

        self.min = np.min(adatok)

        self.max = np.max(adatok)

        self.minDatum = idoPontok[list.index(adatok, self.min)]

        self.maxDatum = idoPontok[list.index(adatok, self.max)]

        self.adf = {}; self.kpss = {}

        self.aic = 0

        self.Stationarity()

        print(self.SeasonsAvg())

*az adatsorok osztályának konstruktora*

    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':0}

        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)

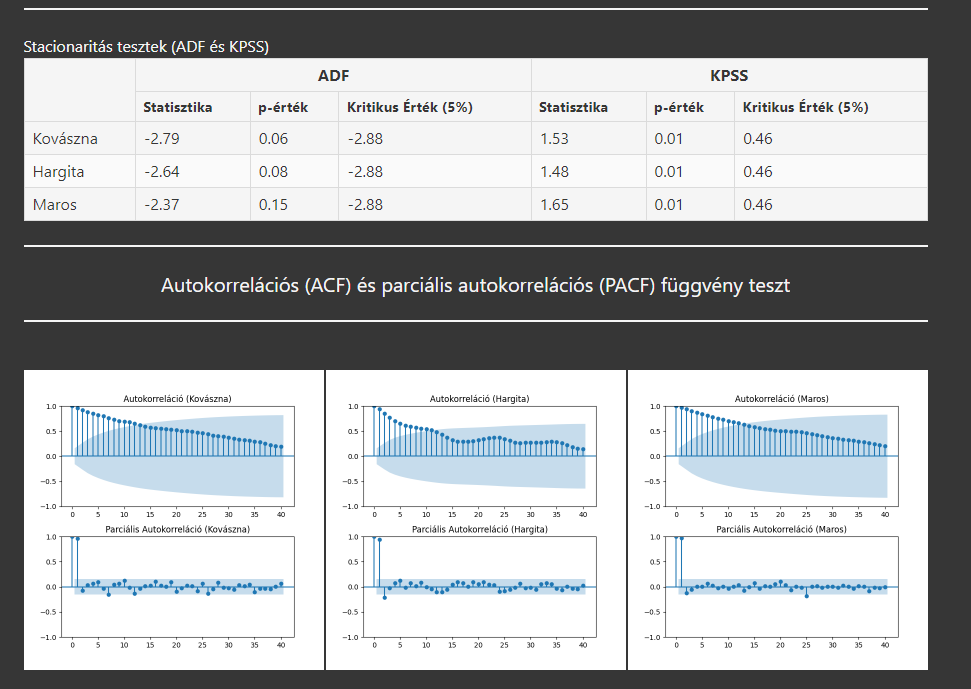
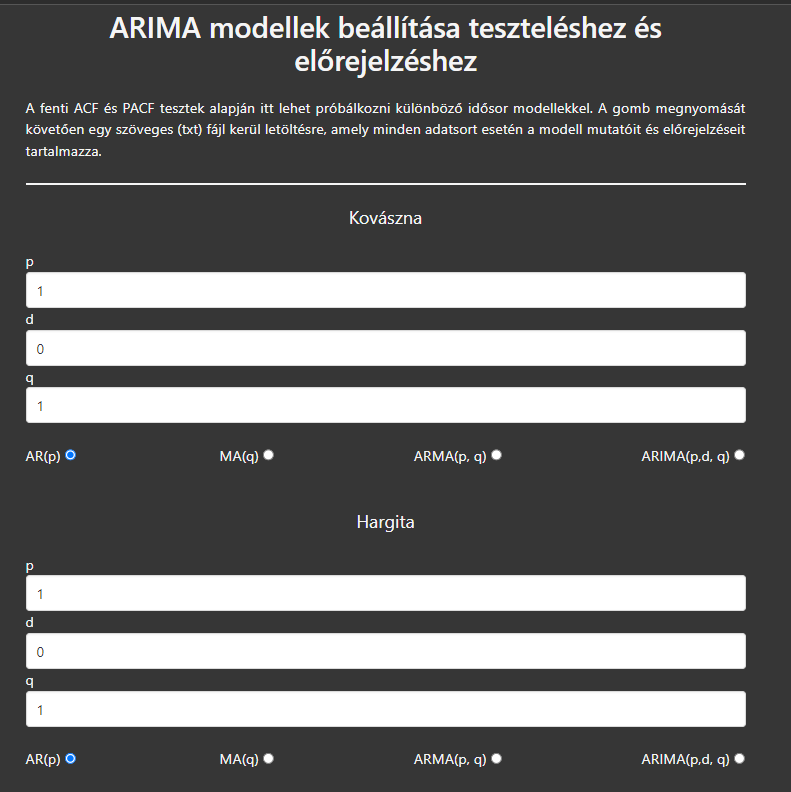
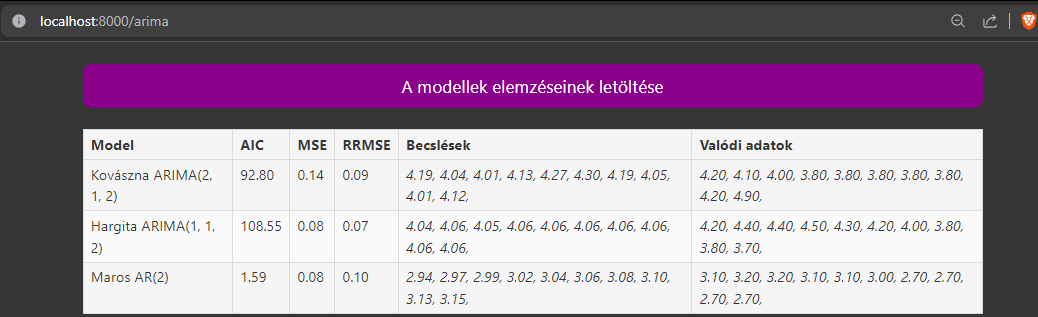
        self.kpss["p\_value"] = round(kpss\_result[1], 2)

        self.kpss["critical\_values"] = {'5':0}

        self.kpss["critical\_values"]['5'] = round(kpss\_result[3]["5%"], 2)

*az adatsorok stacionaritás teszteket futtató, és azok eredményeit eltároló függvény kódja*

## 5.7 ARIMA előrejelzések

Amikor a feltöltöttük az adatforrást, a sikeres feldolgozást követően minden lényeges adat megjelenik egy weboldalon. Először a beolvasott adatok HTML táblázatban megjelenítve, utána az adatsorokról készített grafikon, majd a statisztikai mutatókat tartalmazó táblázat.   
Ezalatt a stacionaritás tesztek (ADF és KPSS) eredményei egy táblázatban, majd a parciális autokorrelációs és autokorrelációs tesztek grafikonja egy képen, mindegyik beolvasott megye esetében.   
  
 Végül a fentieket figyelembe véve, lehetőségünk van mindegyik adatsorra külön-külön, AR/MA/ARMA/ARIMA modelleket beállítva és a szükséges (p, d, q) paramétereket megadva előrejelzéseket készíteni.   
  
A „Tesztek és előrejelzések futtatása” gombra kattintva egy újabb oldalon megjelennek az eredeti és az előrejelzett értékek, a modell AIC, MSE és RRMSE mutatói.  


Az előrejelzési modelleket a statsmodels.tsa.arima.model python kiegészítő csomag ARIMA nevű függvényével hozom létre, amely az idősor listáját, és a p, d, q paramétereket kéri. Mindegyik esetre (AR, MA, ARMA, ARIMA) készítettem egy függvényt az osztályban, amely az elkészített modell összegzését adja vissza. A views.py fájlban, az előrejelzések paraméterezésért felelős űrlapjának beküldésével létrejövő POST kérésből kiolvasom a kijelölt rádiógomb értékét, és attól függően hívom meg az idősor példány megfelelő függvényét és tárolom el azt egy tömbbe, amelynek az 1-es indexű eleme magát a becslések listáját tartalmazza, ezt adom át többek között az „arimaForecasts” nézetnek.

    def ARMA(self, p:int, q: int, t:int):

        try:

            p = int(p); q = int(q)

            idosor = self.adatok

            model = sm.tsa.ARIMA(idosor, order=(p, 0, q))

            model\_fit = model.fit()

            self.becslesek = [round(value, 2) for value in model\_fit.forecast(t)]

            self.aic = model\_fit.aic

            return ([model\_fit.summary(), self.becslesek])

        except Exception as e:

            print(e)

              if tipus == "ar":

                    test\_results = megye.AR(p, t)

                    title = f"\n{megye.megye\_nev} AR({p})\n"

                elif tipus == "ma":

                    test\_results = megye.MA(q, t)

                    title = f"\n{megye.megye\_nev} MA({q})\n"

# 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

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

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., & 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

1. Biroul Internaţional al Muncii [↑](#footnote-ref-1)
2. *Az AIC (Akaike Information Criterion) célja az, hogy a modellek közötti összehasonlításra szolgáljon, figyelembe véve a modell illeszkedését és a paraméterszámot. Az AIC egy olyan kompromisszumot keres, amely az illeszkedés minőségét és a modell bonyolultságát egyensúlyozza.* [↑](#footnote-ref-2)