**Sapientia Erdélyi Magyar Tudományegyetem**

**Csíkszeredai Kar**

**Gazdasági Informatika Szak**

**Diplomadolgozat**

**Dolgozat címe**

**Végzős Hallgató:**

**Károlyi Krisztián**

**Témavezető:**

**Dr. Madaras Szilárd, egyetemi adjunktus**

**2024**

**TDK fedlap ide**

**Kivonat**

**Székelyföldi megyék regionális munkanélküliségi rátáinak vizsgálata és előrejelzése Box-Jenkins eljárással és mesterséges neurális hálózatokkal, Django webalkalmazás segítségével**

***Károlyi Krisztián1***

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

**Témavezető:**

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

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

A dolgozatom fő témája, hogy Hargita, Kovászna és Maros megye 2010 január és 2022 augusztus közötti havi munkanélküliségi rátáit elemezve, megkeressem és összehasonlítsam a legjobban illeszkedő ARIMA (autoregresszív mozgóátlag), illetve az MLP (többrétegű perceptron) és LSTM (long short-term memory) modellek mélytanuláson alapuló előrejelzési teljesítményét, 1 éves távlatban.

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

**Tartalomjegyzék**

[1. Bevezetés 4](#_Toc159850578)

[1.1 Szakirodalmi áttekintés 5](#_Toc159850579)

[1.2 A munkanélküliségi ráta fogalmának meghatározása Romániában 6](#_Toc159850580)

[1.3 Felhasznált statisztikai mutatók és fogalmak 7](#_Toc159850581)

[2 Az idősorok elemzése 8](#_Toc159850582)

[3 Előrejelzés Box-Jenkins módszerrel 13](#_Toc159850583)

[3.1 A stacionaritás vizsgálata 14](#_Toc159850584)

[3.2 Autoregresszív és mozgóátlag modellek (AR, MA, ARMA, ARIMA) 15](#_Toc159850585)

[3.3 Autokorrelációs és parciális autokorrelációs tesztek **Error! Bookmark not defined.**](#_Toc159850586)

[3.3.1 ACF 16](#_Toc159850587)

[3.3.2 PACF 17](#_Toc159850588)

[3.3 Előrejelzések és azok pontosságának meghatározása 18](#_Toc159850589)

[4 Neurális hálózatok 20](#_Toc159850590)

[4.1 Neuronok 21](#_Toc159850591)

[4.2 Aktivációs függvények 22](#_Toc159850592)

[4.3 Perceptron, MLP 23](#_Toc159850593)

[4.4 Az MLP neurális hálózat tanítása 24](#_Toc159850594)

[4.5 Az MLP modellek implementációja és előrejelzés 25](#_Toc159850595)

[5 A Django webalkalamzás bemutatása 27](#_Toc159850596)

[5.1 MVC 27](#_Toc159850597)

[5.2 Django 28](#_Toc159850598)

[5.3 Beolvasás Excel-ből **Error! Bookmark not defined.**](#_Toc159850599)

[6 Következtetések 29](#_Toc159850600)

[5. Irodalomjegyzék 30](#_Toc159850601)

# 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 július között. Az adatokat Románia országos statisztikai hivatalának (Institutul Național de Statistică) hivatalos oldaláról töltöttem le.   
Célom, hogy statisztikai elemzést készítsek ezekről az idősorokról, valamint megvizsgáljam, hogy ezen idősorok esetében, 1 év távlatában (2022 augusztus – 2023 július) a Box-Jenkins eljárással készült ARIMA modellek, vagy a gépi tanuláson alapuló MLP (többrétegű perceptron) neuronháló modellek nyújtanak pontosabb előrejelzéseket. A legjobban teljesítő modellekkel pedig a 2023 augusztus – 2024 január közötti időszakkal készítek 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.

# Szakirodalmi áttekintés

Ebben a részben röviden ismertetek néhány korábbi 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 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

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

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

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

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

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

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

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

## Felhasznált statisztikai mutatók és fogalmak

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

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

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

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

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

Az előrejelzések pontosságának megállapításához három mutatót használtam, amelyek minél alacsonyabbak, a modellnek annál pontosabbak az előrejelzései.

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

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

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

*MAPE* =

# Adatbeolvasás

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

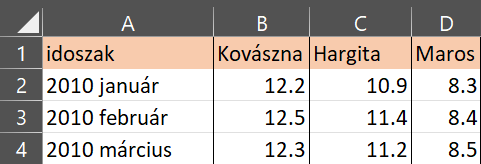
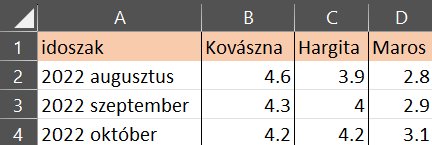
Az adatok beolvasása egy egyszerű HTML űrlap segítségével történik, ez a Django webalkalmazás kezdőlapja, amely a 1. ábrán látható:



**1. Ábra**: a webalkalamzás kezdőlapja

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

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



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



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

# 3. Az idősorok elemzése

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

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

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

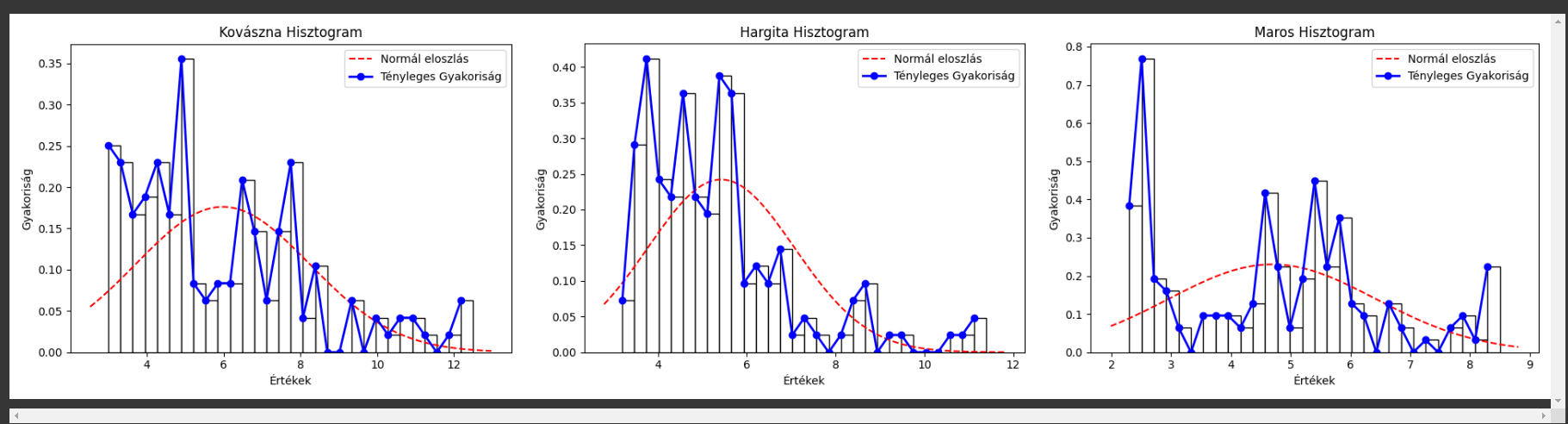
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **mutató** | **Kovászna megye** | | **Hargita megye** | | **Maros megye** | |
| Átlag | 5.98% | | 5.43% | | 4.69% | |
| Szórás | 2.26% | | 1.65% | | 1.73% | |
| Variancia | 5.13% | | 2.71% | | 3.01% | |
| Medián | 5.20% | | 5.20% |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

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

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

## 3.2 Az eloszlások vizsgálata, standardizáció

Egy másik fontos szempont, amivel érdemes tisztában lenni az előrejelzési modellek készítése előtt, az idősor eloszlása. A normál eloszlást nem követő idősorok torzíthatják az előrejelzéseket, ilyenkor érdemes az adatokat a modell számára transzformálni, például logaritmizálással (majd visszaalakítás exponenciális függvénnyel), vagy standardizálással.



**4. ábra**: az idősorok eloszlása

A negyedik ábrán az látszik, hogy egyik megye idősora sem igazán követi a normál eloszlást (Maros megyénél majdnem), viszont egy Kolmogorov-Smirnov próbának is alávetettem az adatsorokat, amely szintén alátámasztotta az előbbi feltevést. Ez a teszt azt vizsgálja, hogy egy minta mennyire tér el egy feltételezett eloszlástól (esetünkben a normális eloszlásra).

Az eloszlások közötti különbség mértékét a tesztstatisztika értéke mutatja, minél kisebb ez a szám, a minta a annál jobban illeszkedik a feltételezett eloszlásra. A nullhipotézis (H0) az, hogy a minta megfelel az elméleti eloszlásnak, az alternatív (H1) pedig az, hogy nem. Akkor utasítjuk el a nullhipotézist, ha a statisztika értéke nagy és a szignifikancia-szint (p-érték) alacsony. A webalkalamzásomba beépítettem ezt a tesztet is a scipy.stats csomag ks függvénye segítségével, amely szintén alátámasztotta azt, hogy nem normál eloszlásúak az idősorok.

**3. táblázat**: Kolmogorov-Smirnov teszt normál eloszlásra

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

Emiatt a neurális hálózatokon alapuló MLP és LSTM modelleknél az adatokat standardizálni fogom a Scikit-Learn StandardScaler osztály segítségével, mert úgy sokkal gyorsabban konvergálnak a tanulás alatt és jobban teljesítenek az előrejelzéseknél. A standardizáció megpróbálja az adatok átlagát nullára és szórását egyre állítani N(0,1), így közelebb hozva a mintát a normál eloszlásra. Ilyenkor a megfigyelésekből kivonjuk az átlagot, majd elosztjuk a szórással.

Az idősorokat kellőképpen megvizsgáltuk, a következő lépés a Box-Jenkins eljárás bemutatása lesz.

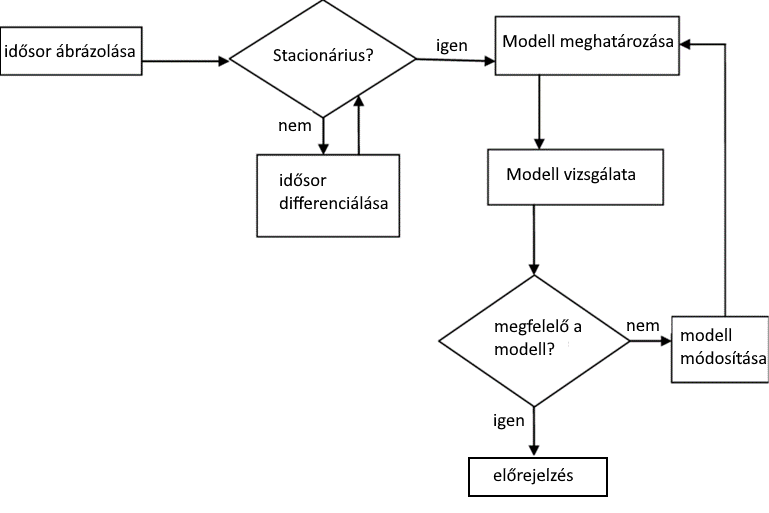
# 4 Előrejelzés Box-Jenkins módszerrel

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

Az eljárás főbb lépései a következők:

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

Ha nem elég jók az eredmények, másféle modelleket is ki kell próbálni, tehát a harmadik, a negyedik és az ötödik lépés iteratív folyamatot jelent.



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

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

## 4.1 A stacionaritás vizsgálata

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(Sándor, 2019)

## 4.3 Autokorrelációs függvény használata mozgóátlag komponens meghatározására

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

Az általam vizsgált idősorok autokorrelációs tesztjei a következőképpen néznek ki:



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

Ha MA 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.

## 4.4 Parciális autokorrelációs függvény használata autoregresszív komponens meghatározására

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

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



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

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

## 4.5 A megfelelő ARIMA modell kiválasztása

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

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

Az AIC értékeket a python-ben létrehozott ARIMA modell példány model\_fit,aic attribútumából kértem le. a. A következő táblázatban összefoglalom, hogy a különböző modellekre milyen AIC értékeket kaptam az egyes megyék esetében.

**4. táblázat**: a kipróbált ARIMA modellek AIC érétkei

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

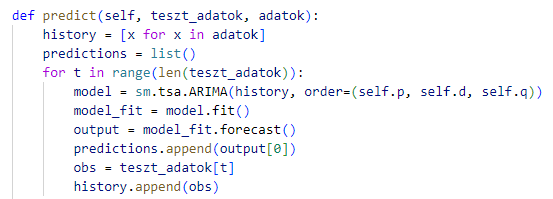
Jóval több modell is tesztelve lett, azonban csak a legjobb négyet tüntettem fel a táblázaton. Úgy tűnik, mindegyik megye esetében jobban illeszkedik az (2,1,2), mint a sima AR (2) modell. Az előrejelzések pontosságánál ez később be is igazolódott.

## 4.5 ARIMA modellek implementálása python-ben és előrejelzés

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

Az előrejelzés lépésről lépésre történt, az úgynevezett walk-forward (előrelépő) típusú keresztvalidációt alkalmaztam, (Brownlee, 2023) alapján. Az eljárás lényege az, hogy iteratívan, egyszerre csak egy értéket jelzünk előre, ezután az előrejelzett értékkel frissítjük a tanítóhalmazt, majd a frissített adathalmazzal újraillesztjük a modellt.

A legjobban teljesítő ARIMA (2, 1, 2) modellek előrejelzéseinek grafikonját, MSE, RRMSE, MAPE értékek táblázatát az eredményeket összefoglaló részben, az MLP és LSTM modellekkel együtt mutatom be, a könnyebb összehasonlítás miatt.



**8. ábra:** ARIMA modell illesztése előre lepő validációval Python-ben

# 5 Neurális hálózatok

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

Egy neuronháló általában háromféle rétegből áll:

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

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

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

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

(Brassai, 2019)

## 5.1 Neuronok

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



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

A neuronok a következő tényezőket használják:

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

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

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

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

(Brassai, 2019)

## 5.2 Aktivációs függvények

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

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



A ReLu (Rectified Linear Unit) aktivációs függvény grafikonja.  
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[[1]](#footnote-1). Sok rétegből álló (mély) neuronhálók esetében sokkal jobb, mint például a szigmoid vagy a tangens hiperbolikus függvény. (Brassai, 2019)

## 5.3 Perceptron, MLP

A perceptron egy régebbi típusú mesterséges neurális hálózat, 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

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

(Brassai, 2019)

## 5.4 Felhasznált tanítási stratégia

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

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

A tanulás során a neuronhálóháló lépésenként, az addigi ismeretei alapján kiszámítja, hogy az adott bemeneti adatból milyen kimeneti érték következik (a tanítóhalmaz következő elemét megjósolja, majd megnézi, mennyit tévedett). Az 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 tanítási folyamatot. Túl sok tanítási ciklus során könnyen előfordulhat, hogy a modellt túltanítjuk (overfitting). Ilyenkor a tanító halmazon ugyan a hiba csökken (egyre jobban illeszkedik rá), viszont a teszthalmaz elvárt eredményeitől egyre jobban távolodik. Ilyenkor le kell állítani a tanítási folyamatot, újra kell gondolni a háló paramétereit, tanítási algoritmusát. (Brassai, 2019)

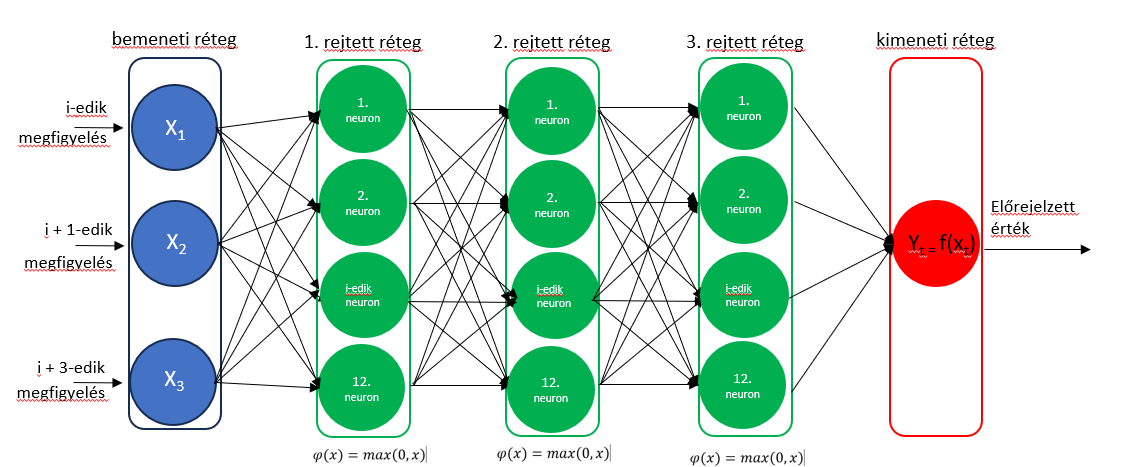
## 4.5 A tanító- és teszthalmaz előkészítése sliding windows módszerrel

Az idősorokból való tanító- és teszt halmazok készítésére az úgynevezett „sliding windows” módszert használtam, a (Brownlee, Predict the Future with MLPs, CNNs and LSTMs in Python, 2018) könyv szerint, ahol a függőváltozót (yt) n db (késleltetett érték, lag) korábbi megfigyelés határozza meg. Az előrejelzéshez viszont fel kell használni a tanítóhalmaz utolsó n darab megfigyelését, hogy az első előrejelzett értéket ki tudjuk számolni. Például, lag = 3 esetén, a 2022 júliusi, augusztusi és szeptemberi adatokból fog kiszámolódni az októberi (első előrejelzett) érték. A novemberi megfigyelést pedig már az augusztusi, szeptemberi és az előző lépésben kiszámított októberi érték adja.

X1, X2, X3 🡪 Y1   
[2.7, 2.8, 2.9] 🡪 3.1

Például, 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, hanem a legoptimálisabb állapota szerint megpróbálja azokat megjósolni. Az előrejelzés pontosságát így meg lehet határozni a teszthalmaz alapján.

## 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

# A Django webalkalamzás bemutatása

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

## MVC



Az MVC alapvető működési elve

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

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

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

## 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)



# 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

Brownlee, J. (2018). *Predict the Future with MLPs, CNNs and LSTMs in Python.*

Brownlee, J. (2023. november 18). *How to Create an ARIMA Model for Time Series Forecasting in Python*. Letöltés dátuma: 2024. március, forrás: machinelearningmastery.com: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

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

1. A gradiens-elhalás probléma akkor fordul elő, amikor a gradiensek túl kicsik lesznek, és így a súlyokat nehéz vagy lassú frissíteni a tanulás során. [↑](#footnote-ref-1)