UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE AUTOMATICĂ ȘI INFORMATICĂ INDUSTRIALĂ

PROIECT DE DIPLOMĂ

Tehnici statistice și de învățare automată pentru serii de timp

Rareș Tudor Buzatu

Coordonator științific:

Prof. dr. Ion Necoară

BUCUREȘTI
2020

Cuprins

1 Indroducere **1**

1.1 Context . . . 1

1.2 Problemă . . . 1

1.3 Obiective . . . 2

1.4 Soluție propusă . . . 2

1.5 Structura lucrării . . . 2

2 Metode statistice **3**

2.1 Noțiuni statistice . . . 3

2.1.1 Serie de timp . . . 3

2.1.2 Zgomot vs. semnal util . . . 4

2.1.3 Noțiuni generale . . . 5

2.1.3.1 Deviație standard . . . 5

2.1.3.2 Variantă . . . 5

2.1.3.3 Covariantă . . . 5

2.1.3.4 Corelație . . . 6

2.1.4 Noțiuni particularizate pe serii de timp . . . 6

2.1.4.1 Autocovariantă . . . 6

2.1.4.2 Autocorelație . . . 6

2.1.4.3 Autocorelație parțială . . . 7

2.1.4.4 Regresie liniară . . . 7

2.2 Modelul de predicție . . . 12

2.2.1 Caracteristicile modelului . . . 13

2.2.2 Autoregresie (AR) . . . 14

2.2.3 Medie alunecătoare (MA) . . . 16

2.2.4 Integrare (I) . . . 17

2.2.5 Sezonalitate(S) . . . 20

2.2.6 Testul de staționarizare: Dickey-Fuller . . . 21

2.2.7 Akaike Information Criterion(AIC) . . . 23
2.2.8 Bayesian Information Criterion(BIC) . . . 24

3 Metode de învățare automată 25
3.1 Noțiuni introductive . . . 25
3.1.1 Neuronul artificial . . . 25
3.1.2 Perceptronul . . . 26
3.1.2.1 Datele de intrare . . . 26
3.1.2.2 Ponderi . . . 26
3.1.2.3 Bias . . . 27
3.1.2.4 Funcția de activare . . . 27
3.1.2.5 Combinația liniară . . . 28
3.1.2.6 Datele de ieșire . . . 28
3.1.3 Rețele neurale . . . 28
3.1.3.1 Funcție de cost . . . 29
3.1.3.2 Propagare înapoi . . . 31
3.1.3.3 Optimizarea propagării înapoi . . . 33
3.2 Modelul de predicție . . . 34
3.2.1 Alegerea modelului . . . 34
3.2.2 Caracteristicile modelului . . . 35
3.2.3 Multi Layer Perceptrons (MLP) . . . 35
3.2.4 Prelucrarea datelor . . . 36
3.2.5 Configurarea rețelei . . . 37

4 Rezultate 38
4.1 Exemplu complet SARIMA . . . 38
4.1.1 Achiziționarea datelor . . . 38
4.1.2 Prelucrarea datelor . . . 39
4.1.2.1 Interpretarea datelor . . . 40
4.1.3 Construcția modelului . . . 43
4.2 Exemplu complet MLP . . . 48
4.2.1 Achiziționarea datelor . . . 48
4.2.2 Prelucrarea datelor . . . 48
4.2.3 Construirea rețelei . . . 49
4.3 Exemple SARIMA & MLP . . . 51
4.3.1 Serie de timp 1: Vânzări medicamente . . . 52

4.3.1.1 SARIMA 52
4.3.1.2 MLP 54
4.3.2 Serie de timp 2: Consum de energie 55
4.3.3 Serie de timp 3: Valori bursă 58
4.3.3.1 SARIMA 58
4.3.3.2 MLP 60

5 Concluzii și propuneri **61**
5.1 Concluzii 61
5.2 Propuneri 62

6 Bibliografie **63**

SINOPSIS

Această lucrare are scopul de a prezenta problema predicţiei seriilor temporale şi de a face o comparaţie între metodele statistice clasice cu rezultate relevante în acest domeniu şi metodele care implică învăţarea automată, metode mult mai moderne decât cele statistice, dar cu o performanţă încă necunoscută. Lucrarea va aborda atât din punct de vedere matematic, cât şi din punct de vedere practic identificarea unei metode cât mai performante pentru o predicţie cu o acurateţe cât mai bună, dar şi vizualizarea si interpretarea rezultatelor obţinute în urma predicţiilor prin diverse metode. Se ve enunța teoria care stă în spatele acestor metode, se va demonstra această teorie pe exemple concrete, iar pe baza ei se vor prezenta diverse tehnici de construire a unui model de predicție cât mai bine ales pentru problema propusă.

ABSTRACT

This paper aims to present the problem of forecasting time series and to do a comparison between classical statistical methods with relevant results in this field and methods involving machine learning, much more modern than statistical methods, but with a performance still unknown. The paper will address both, from a mathematical point of view as well as from a practical point of view the identification of the most efficient method for a prediction with the best possible accuracy, but also the visualization and interpretation of the results obtained from predictions by various methods. The theory behind these methods will be stated, this theory will be demonstrated on concrete examples, and based on it, there will be presented various techniques for building a prediction model best chosen for the proposed problem.

Capitolul 1

Indroducere

În acest capitol se prezintă motivația alegerii acestui subiect, obiectivele de îndeplinit, soluția propusă și structura lucrării.

1.1 Context

Predicţia seriilor de timp reprezintă un domeniu tot mai căutat la momentul actual deoarece, odată cu creşterea puterii de procesare a calculatoarelor, metodele de predicţie a seriilor de timp capătă o acurateţe din ce în ce mai mare. Oamenii au încercat să prezică viitorul încă de acum mii de ani, una dintre formele primare de prezicere fiind prezicerea vremii. Odată cu evoluţia tehnologică şi matematică, aceste preziceri au început să se transforme în predicţii, adică au inceput să se bazeze pe concepte matematice. În zilele noastre, acest domeniu şi-a extins graniţele spre o varietate extinsă de domenii[1], cum ar fi:

- Predicţia consumului energetic
- Predicţie demografică
- Predicţia cursurilor valutare
- Predicţia geopolitică

1.2 Problemă

Problema abordată în acest proiect constă în identificarea, testarea și compararea metodelor atât statistice, cât și a metodelor bazate pe învăţare automată. Întrucât metodele de predicţie sunt generale, ele nedepinzând de domeniul seriei de timp, lucrarea se va axa pe domeniul energetic şi financiar, deoarece dorinţa actuală a oamenilor este îndreptată spre aceste domenii, ele fiind domenii cu foarte multe date accesibile și pe care societatea actuală se bazează.

1.3 Obiective

Proiectul îşi propune să determine care dintre cele două variante de predicţie oferă rezultate mai bune şi în ce cazuri, dar şi să ofere o mai bună întelegere a implementării acestor metode, fără a utiliza programe *software* pentru a face acest lucru automat.

1.4 Soluție propusă

Soluţia cu care vine acest proiect constă în analiza literaturii de specialitate din lume pentru a găsi metodele cele mai bune pentru predicţie din cele două domenii: statistice şi bazate pe învăţare automată. După ce metodele au fost identificate, acestea vor fi analizate din punct de vedere matematic pentru o mai bună înţelegere a lor şi testate pentru verificarea performanţelor. Compararea rezultatelor metodelor pe aceleaşi seturi de date va oferi un răspuns clar la întrebarea:
Cine deţine avantajul acurateţii în momentul de faţă în ceea ce priveşte predicţia de date?

1.5 Structura lucrării

Lucrarea constă în 3 părţi principale, fiecare având un scop clar în rezolvarea problemei propuse:

1. Metode statistice
   Se va analiza din punct de vedere matematic una dintre metodele cu rezultate deosebite în domeniul predicțiilor şi se va construi manual modelul matematic pentru a aproxima cât mai bine seria de timp. Se vor enunţa reguli generale de construcţie a modelului pe baza rezultatelor obţinute.

2. Metode bazate pe învăţarea automată
   Se va explica principiul de funcţionare al învăţării automate. Se va particulariza acest principiu pentru a construi o rețea neurală care are drept scop implementarea unei metode folosită în predicția seriilor de timp. Se va testa rețeaua și se vor analiza performanțele metodei implementate.

3. Comparare rezultate
   Se vor compara rezultatele obținute folosind cele două variante enunțate mai sus pe aceleași seturi de date, variind diverși parametri ai seriilor de timp pentru a avea o privire de ansamblu completă asupra acurateții metodelor.

Capitolul 2

Metode statistice

În acest capitol se vor prezenta câteva elemente de statistică generale, apoi acestea se vor particulariza pe serii de timp, iar după aceea se va arăta cum se construiește un model de predicție bazat pe concepte statistice.

2.1 Noțiuni statistice

Pentru a întelege pe deplin cum funcționează metodele de predicție statistice, se va începe prin prezentarea unor noțiuni fundamentale necesare, menite să sprijine matematic conceptele ulterioare care vor fi prezentate.

2.1.1 Serie de timp

Primul lucru care trebuie înțeles atunci când se dorește predicția unei serii de timp este înțelegerea conceptului de serie de timp. O serie de timp reprezintă o funcție matematică cu domeniul de definiție în domeniul timp[2]. Seriile de timp pot fi continue sau discrete, valorile pe care le poate lua funcția aparținând numerelor reale, iar domeniul de definiție aparținând numerelor reale pozitive. Se consideră că seria de timp începe din momentul de timp 0:

$$X : R_+ \cup \{0\} \rightarrow R^n. \tag{2.1}$$

În continuare se vor considera doar seriile de timp discretizate, deoarece este imposibil de preluat dintr-un proces real funcția continuă pe un calculator pentru procesare. De asemenea, se vor utiliza doar funcții de timp univariate: codomeniul funcției aparține lui R^n, unde n=1. În Fig. 2.1 este exemplificată o asftel de serie de timp: funcția ia valori între 140 și 190 pe o durată de timp începând cu Februarie 2017 și terminând cu Mai 2018.

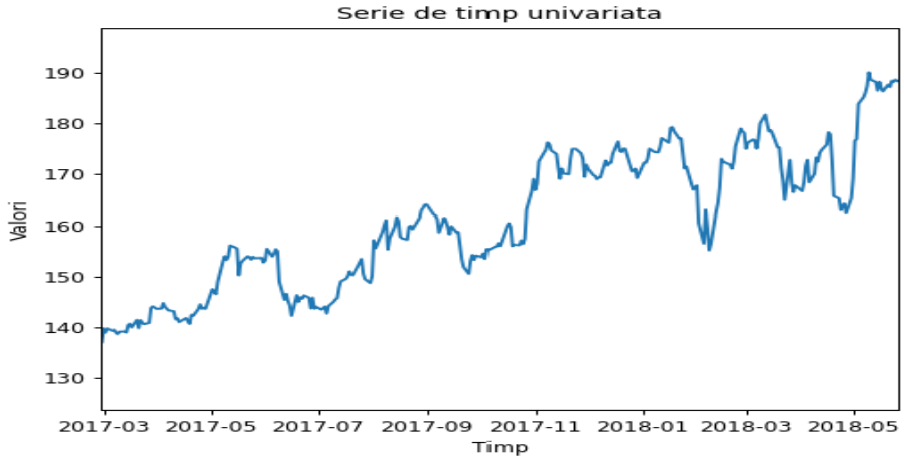

Figura 2.1: Serie de timp univariată.

2.1.2 Zgomot vs. semnal util

Orice semnal din lumea reală este afectat de zgomot. Cel mai des întâlnit tip de zgomot este zgomotul alb. Zgomotul alb reprezintă o funcție de timp cu valori aleatoare. Caracteristica principală a zgomotului alb o reprezintă frecvențele prezente în acesta: zgomotul alb reprezintă un semnal care are în teorie aceeași valoare a frecvențelor de la 0 la infinit[3]. Acest lucru se poate observa și în Fig. 2.2 , figură care reprezintă spectrul zgomotului alb.

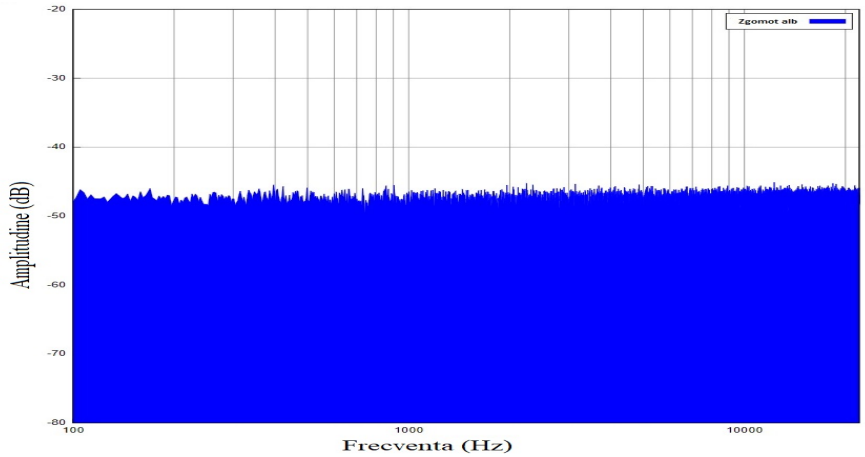

Figura 2.2: Spectru zgomot alb.

Motivul pentru care frecvențele nu au exact aceeași amplitudine îl reprezintă erorile generate de limitările calculului numeric pe calculator. Având în vedere că zgomotul nu se poate prezice și că semnalul util o să fie afectat de acest zgomot, va fi nevoie de câteva ipoteze inițiale pentru a putea lucra cu semnalul respectiv:

1. Semnalul util trebuie să difere de zgomot. Acest lucru se poate verifica foarte ușor

cu ajutorul Transformatei Rapide Fourier(FFT).

2. Nivelul zgomotului alb trebuie să fie mic relativ la nivelul semnalului util. Acest lucru permite diferențierea semnalului util de zgomot.

3. Apare nevoie de a lucra cu variabile mai generale: valoare medie, trend, sezonalitate.

2.1.3 Noțiuni generale

Fie X și Y două variabile aleatoare univariate.

2.1.3.1 Deviație standard

În statistică, deviația standard este o mărime care reprezintă cât de mult variază valorile unui set de date în jurul valorii medii[4]. O deviație standard mică, înseamnă că datele sunt foarte apropiate de valoarea medie, iar o deviație mare înseamnă că valorile sunt împrăștiate pe un interval mare în jurul valorii medii. În literatură, deviația standard este notată fie cu *SD*, fie cu σ și este egală cu:

$$\sigma = \sqrt{E[(X - \mu)^2]}, \tag{2.2}$$

unde X este o variabilă aleatoare, E este media (expectanța) lui X și μ este valoarea medie a lui X, adică este egal cu $E[X]$.

2.1.3.2 Varianță

Varianța reprezintă pătratul deviației standard și măsoară cât de depărtate sunt valorile unei variabile aleatoare de valoarea lor medie[3]:

$$\mathrm{Var}(X) = E[(X - \mu)^2] = \sigma^2. \tag{2.3}$$

2.1.3.3 Covarianță

În statistică, covarianța reprezintă o mărime corespunzătoare gradului de asociere dintre două variabile. Presupunând două variabile aleatoare X și Y, se poate spune că există o covarianță pozitivă între X și Y dacă atunci când se variază valoarea variabilei X într-un sens (ex: pozitiv), valoarea variabilei Y se modifică în același sens. De asemenea, există o covarianță negativă între X și Y dacă atunci când se variază valoarea variabilei X într-un sens, valoarea variabilei Y se modifică în sens opus. Mai jos este formula generală a covarianței pentru două variabile aleatoare X și Y[3]:

$$\mathrm{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])]. \tag{2.4}$$

2.1.3.4 Corelație

Corelația, în statistică, are aceeași semnificație precum covarianța, dar are drept codomeniu un interval foarte bine stabilit: [-1,1]. Formula matematică pentru corelație[3] este:

$$\text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}. \tag{2.5}$$

Așadar, pentru a afla corelația dintre două variabile aleatoare trebuie să se afle covarianța, iar apoi să se împartă la deviația standard a lui X înmulțită cu deviația standard a lui Y. În felul acesta se normează valoarea covarianței, iar rezultatul va aparține mereu intervalului [-1,1].

2.1.4 Noțiuni particularizate pe serii de timp

Fie $(X_t)_{t\geq 0}$ o serie de timp univariată.

2.1.4.1 Autocovarianță

Se numește autocovarianță covarianța dintre o variabilă aleatoare și ea însăși. Vorbind despre serii de timp, variabilele aleatoare implicate variază în timp. Datorită acestui fapt se poate considera o covarianță între semnalul X_t și semnalul X_t întârziat cu un număr stabilit de eșantioane. Prin urmare, matematic, autocovarianța dintre X_t și X_{t-k} [5] este:

$$\text{Cov}(X_t, k) = E[(X_t - E[X_t])(X_{t-k} - E[X_{t-k}])]. \tag{2.6}$$

Fiind vorba despre un semnal și același semnal întârziat, funcția de autocovarianță este simetrică față de k:

$$\text{Cov}(X, k) = \text{Cov}(X, -k). \tag{2.7}$$

2.1.4.2 Autocorelație

Autocorelația este foarte asemănătoare cu autocovarianța, ea având formula[5]:

$$\text{Corr}(X_t, k) = \frac{\text{Cov}(X_t, k)}{\sigma_{X_t} \sigma_{X_{t-k}}}. \tag{2.8}$$

Faptul că există o corelație între două serii de timp înseamnă că prima serie depinde de cea de-a doua. Fiind vorba despre un semnal și același semnal întârziat, acest lucru înseamnă că semnalul X_t depinde de valoarea precedentă a lui X_t, adică de X_{t-1}. Acest lucru este esențial în predicția seriilor de timp, deoarece acesta ne spune de câte intervale de timp anterioare momentului actual depinde valoarea seriei la momentul actual. Acest lucru este foarte intuitiv pe un exemplu meteo: vremea care va fi mâine depinde mult mai mult de vremea care este astăzi și care a fost ieri, decât de vremea care a fost acum

o saptămână sau acum o lună. Tradus în termeni matematici, autocorelația de ordin 1 și 2 este mult mai mare decât autocorelația de ordin 7 sau 30.

2.1.4.3 Autocorelație parțială

Autocorelația parțială reprezintă o noțiune statistică foarte asemănătoare cu autocorelația. Această noțiune se va folosi în secțiunea următoare, ea fiind vitală găsirii unui model care să aproximeze cât mai bine seria de timp. Formula generală a autocorelației parțiale este:

$$\mathrm{PCorr}(X_t, X_{t-k}) = \frac{\mathrm{Cov}(X_t, X_{t-k}|X_{t-1}, X_{t-2}, \ldots, X_{t-k+1})}{\sqrt{\mathrm{Var}(X_t|X_{t-1}, X_{t-2}, \ldots, X_{t-k+1})\mathrm{Var}(X_{t-k}|X_{t-1}, X_{t-2}, \ldots, X_{t-k+1})}}, \tag{2.9}$$

unde X este o variabilă aleatoare care depinde de momentele anterioare. De asemenea $\mathrm{Cov}(X_t, X_{t-k}|X_{t-1}, X_{t-2}, \ldots, X_{t-k+1})$ este covarianța dintre X_t și X_{t-k} condiționată de $X_{t-1}, X_{t-2}, \ldots, X_{t-k+1}$. În literatura de specialitate această funcție se găsește prescurtată $PACF$[6]. La fel ca și autocorelația, graficul acesteia se numește autocorelogramă parțială și arată extrem de asemănător cu cel al autocorelației.

2.1.4.4 Regresie liniară

Regresia reprezintă o metodă de a modela relația dintre o variabilă dependentă și o variabilă independentă. Regresia liniară presupune aproximarea funcției ce reprezintă relația dintre cele două variabile cu o funcție liniară[7]. Forma generală a unei regresii este:

$$y = X\beta + \epsilon, \tag{2.10}$$

unde $y = \begin{bmatrix} y_1 & y_2 & \ldots & y_n \end{bmatrix}^T$ = valori observate; $\beta = \begin{bmatrix} \beta_1 & \beta_2 & \ldots & \beta_p \end{bmatrix}^T$ = parametri; $\epsilon = \begin{bmatrix} \epsilon_1 & \epsilon_2 & \ldots & \epsilon_n \end{bmatrix}$ = erori; $X = \begin{bmatrix} X_1^T & X_2^T & \ldots & X_n^T \end{bmatrix}^T = \begin{bmatrix} x_{11}\ x_{12}\ \ldots\ x_{1p} \\ x_{21}\ x_{22}\ \ldots\ x_{2p} \\ \vdots \\ x_{n1}\ x_{n2}\ \ldots\ x_{np} \end{bmatrix}$ = regresori.

Nevoia de regresie apare în interpretarea grafică a noțiunilor de covarianță/corelație. În continuare este prezentat un exemplu care să demonstreze integrarea regresiilor în noțiunile discutate mai sus. Se consideră un experiment în care se măsoară energia consumată de către un consumator pe o perioadă de 10 zile[1]. Plecând de la premisa că experimentul este real, este natural să se asume că în datele măsurate există zgomot. Prin urmare funcția consumului de energie în funcție de numărul zilei nu se poate determina în totalitate. În schimb, aceasta se poate aproxima. Se consideră o regresie liniară pentru a putea aproxima funcția. Această regresie are rolul de a găsi o funcție de gradul I care

[1]Exemplul se găsește la `https://docs.google.com/presentation/d/1QT45APv3wmV5vvVAn4tdv483SZu8NC3Xg9GI1uMjCNM/edit#slide=id.g37764e8af4_0_45`

să minimizeze distanța de la funcția găsită la fiecare punct al graficului. Fiind o funcție de ordinul I, ea va fi egală cu:

$$y = \alpha + \beta X, \tag{2.11}$$

unde α și β sunt necunoscutele ce trebuie aflate. Pentru a afla această funcție, se poate folosi de exemplu metoda Celor Mai Mici Pătrate (CMMP)[7], care spune că funcția trebuie să minimizeze:

$$\min_{\alpha,\beta} \sum_{i=1}^{\text{NrZile}} \hat{\epsilon_i}^2 = \sum_{i=1}^{\text{NrZile}} (y_i - \alpha - \beta x_i)^2. \tag{2.12}$$

În Fig. 2.3 sunt reprezentate datele și funcția de minimizare găsită:

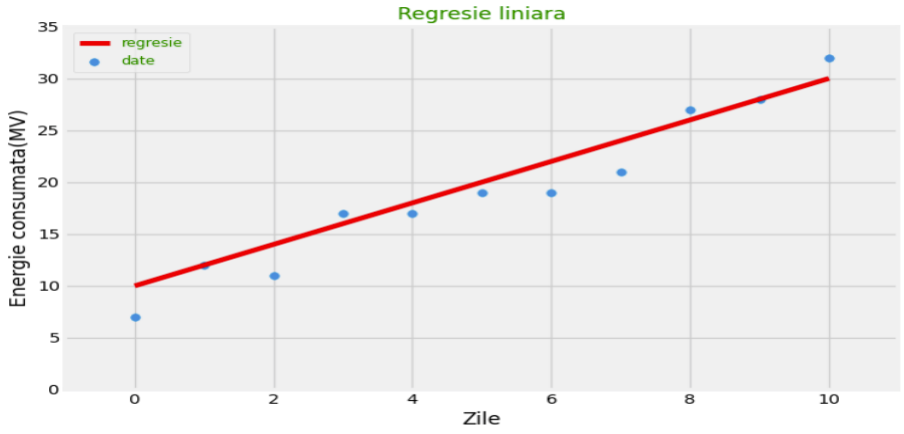

Figura 2.3: Regresie liniară.

Cu cât punctele sunt mai apropiate de linia roșie, cu atât covarianța dintre zile și energia consumată este mai mare în modul. Cu cât punctele sunt mai depărtate, cu atât covarianța este mai apropiată de 0. Dacă panta regresiei este pozitivă, atunci și covarianța va fi tot pozitivă. Dacă panta regresiei este negativă, atunci și covarianța va fi tot negativă. Dezavantajul folosirii covarianței îl reprezintă valorile pe care aceasta le ia. Covarianța este foarte sensibilă la valorile numerice ale datelor, de aceea ea este foarte dificil de interpretat, mai ales dacă nu se cunoaște domeniul valorilor datelor. De exemplu o covarianță cu valoarea 100 a unor date de ordinul zecilor de mii este mult "mai bună" decât o covarianță cu valoarea 10 a unor date de ordinul unităților. Prin "mai bună" se înțelege că datele sunt mult mai apropiate de funcția care le minimizează. Din cauza acestui dezavantaj, pentru a avea un control mai bun asupra semnificației covarianței, în locul său se folosește corelația. Mai jos este prezentat în Fig. 2.4 un grafic care arată nivelul corelației (cantitativ) raportat la distanța unor puncte față de o dreaptă de regresie:

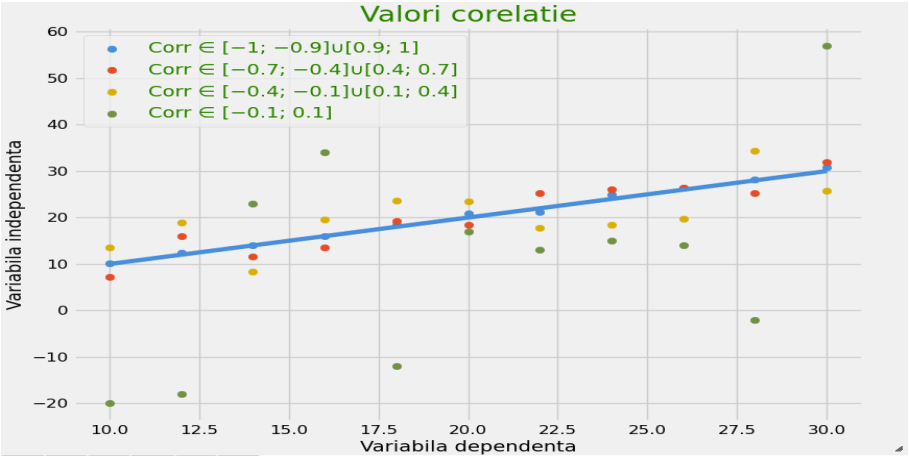

Figura 2.4: Valori corelație.

Determinarea autocovarianței engergiei consumate
Pentru a putea observa autocovarianța seriei de timp din exemplul anterior, se vor afișa în Fig. 2.5 datele din ziua curentă, datele din ziua anterioară și funcția de minimizare a regresiei liniare:

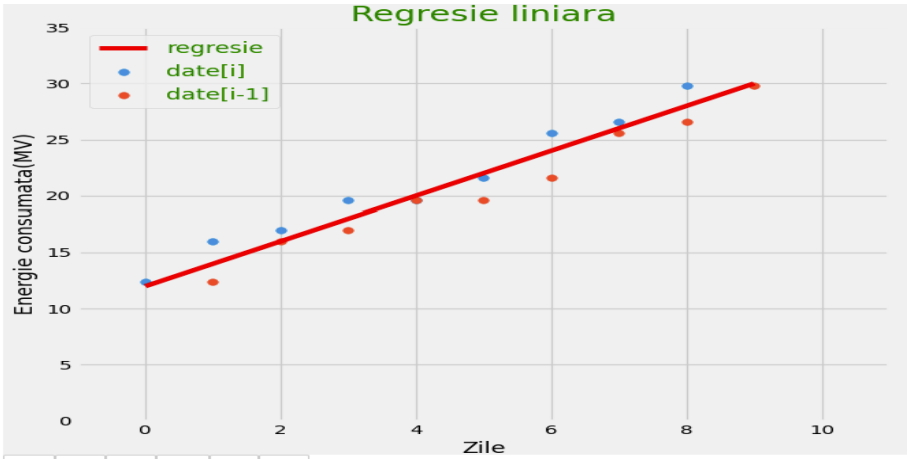

Figura 2.5: Date din ziua curentă și din ziua precedentă.

Se observă că datele portocalii sunt doar datele albastre întârziate cu o unitate de timp, în acest caz: o zi. Următorul pas îl reprezintă observarea relației dintre datele originale în funcție de datele întârziate cu o zi. Astfel se va putea determina autocovarianța $\mathrm{Cov}(X_t, 1)$:

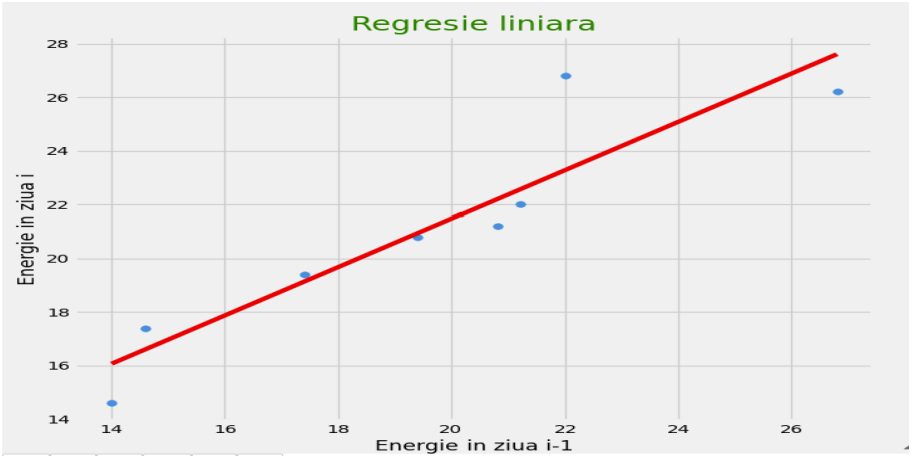

Figura 2.6: Date curente vs. date întârziate.

Din Fig. 2.6 se poate observa că există pantă pozitivă determinată de valorile punctelor, astfel că o regresie liniară poate aproxima destul de bine valorile punctelor. De asemenea, comparând Fig. 2.6 cu Fig. 2.4 se observă că Fig. 2.6 se aseamănă cel mai mult punctelor roșii. Prin urmare există o relație de covarianță medie între datele curente și datele întârziate cu o zi. Prin extindere, există și o corelație medie pozitivă între acestea.

Corelograma seriei de timp
Afișarea pe un grafic a autocorelațiilor unei serii de timp pornind de la ordinul 0 și continuând până la un ordin suficient de mare astfel încât valorile autocorelației să nu mai fie semnificative se numește corelogramă[8]. În Fig. 2.7 este prezentată corelograma energiei din exemplul anterior:

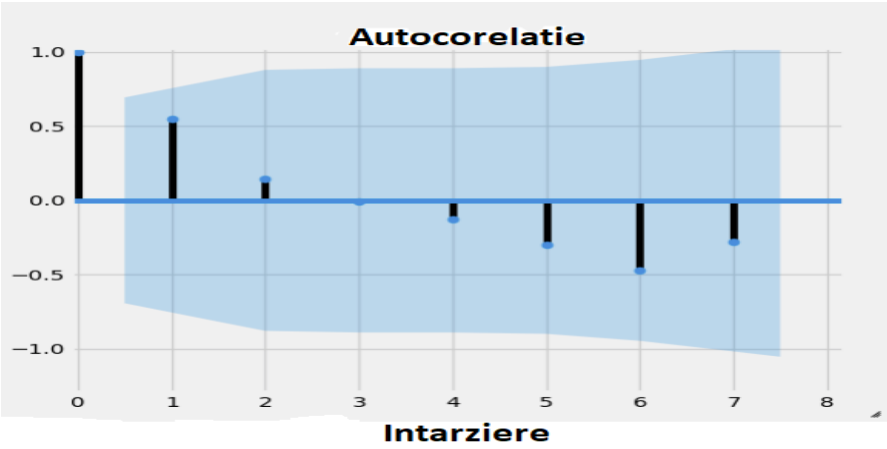

Figura 2.7: Autocorelație.

Întotdeauna valoarea autocorelației de ordin 0 va fi 1. Acest lucru rezultă din formula matematică:

$$\text{Corr}(X_t, 0) = \frac{\text{Cov}(X_t, 0)}{\sigma_{X_t}\sigma_{X_t}} = \frac{E[(X_t - E[X_t])(X_t - E[X_t])]}{\sigma_{X_t}\sigma_{X_t}} = \frac{\text{Var}(X_t)}{\text{Var}(X_t)} = 1. \qquad (2.13)$$

Zona albastră din Fig. 2.7 reprezintă intervalul de încredere. Acesta ne spune că în proporție de 95% valorile care se află în afara lui sunt datorită autocorelației, ci nu datorită unei întâmplări statistice. În literatura de specialitate, funcția de autocorelație este prescurtată AFC.

Calculul autocorelației parțiale:
Principala diferență dintre autocorelație și autocorelație parțială o reprezintă faptul că la autocorelația parțială, în loc să se găsească o funcție de ordinul I care să minimizeze erorile punctelor generate de datele din ziua curentă în funcție de ziua anterioară, se va căuta o funcție care să minimizeze rezidurile zilei curente în funcție de ziua anterioară[8]. Pentru o mai bună intuiție se va considera în continuare exemplul de mai sus în care apare energia electrică consumată în funcție de zile și se va calcula autocorelația parțială de ordin 2. În Fig. 2.6 este prezentat graficul seriei în funcție de seria de timp întârziată cu o unitate. Următorul pas este să se reprezinte erorile zilei curente față de funcția de minimizare în funcție de valorile seriei întârziate. Pentru acest lucru este nevoie să se găsească distanțele de la fiecare punct albastru la dreapta roșie. O reprezentare grafică intuitivă al acestui pas este rotirea în sensul acelor de ceasornic a Fig. 2.6 până când dreapta roșie devine orizontală:

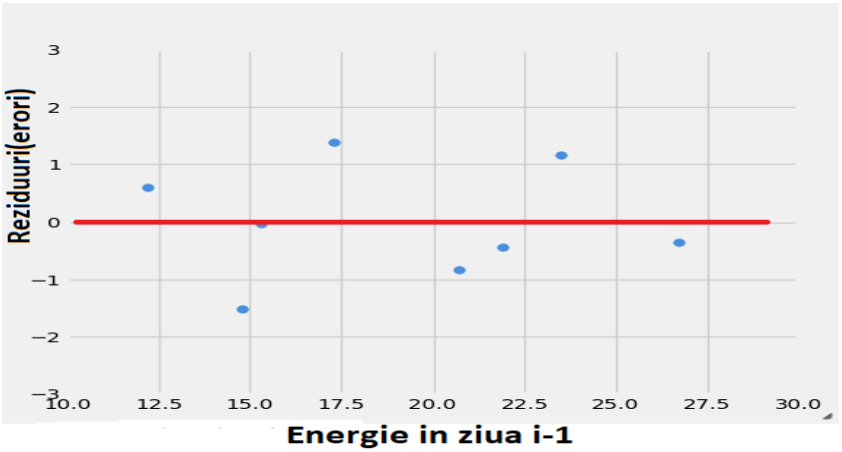

Figura 2.8: Reziduri vs. date întârziate cu o zi.

Ultimul pas îl reprezintă afișarea acelor reziduri în funcție de datele întârziate cu două unități de timp. Pentru acest ultim grafic se va găsi o regresie liniară care să minimizeze erorile. În funcție de acele erori se va determina corelația parțială (asemănător autocorelației):

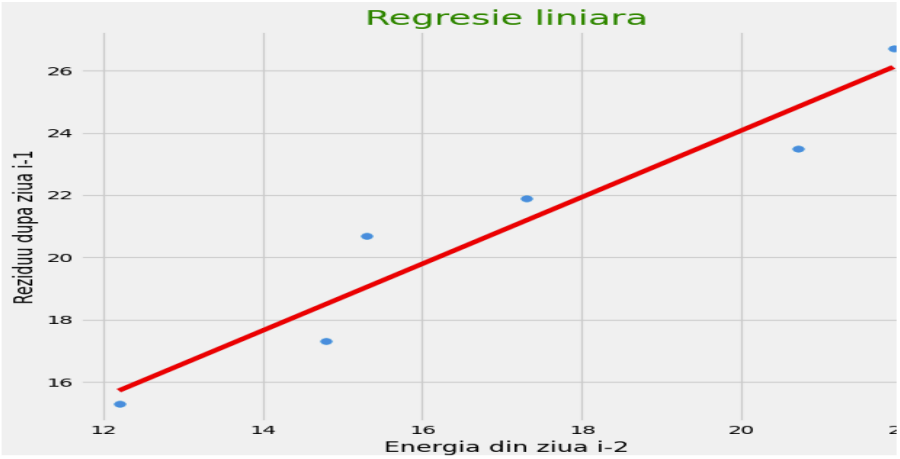

Figura 2.9: Reziduri după ziua i-1 vs. date întârziate cu două zile.

Din punct de vedere al interpretării, autocorelația parțială spune cât de mult este influențată valoarea din ziua curentă în funcție de erorile față de regresia liniară a zilei precedente. Un efect imediat al acestui lucru îl reprezintă eliminarea influenței celorlalte zile asupra valorii zilei curente. Astfel se poate găsi adevarata influență a zilei precedente asupra zilei curente, eliminând influența celorlalte zile anterioare.

2.2 Modelul de predicție

Pentru o comparație cât mai corectă între metodele statistice și cele bazate pe învățare automată, se vor considera metodele cu rezultatele cele mai bune în domeniul predicțiilor. Alegerea modelului statistic a fost facut pe baza articolului științific lui Antonio Rafael Sabino Parmezan ș.a[9] după ce aceștia au analizat mai multe metode, atât statistice, cât și bazate pe învățare automată. Mai jos sunt afișate metodele folosite de către aceștia și rezultatele asociate fiecărei metode în diverse cazuri:

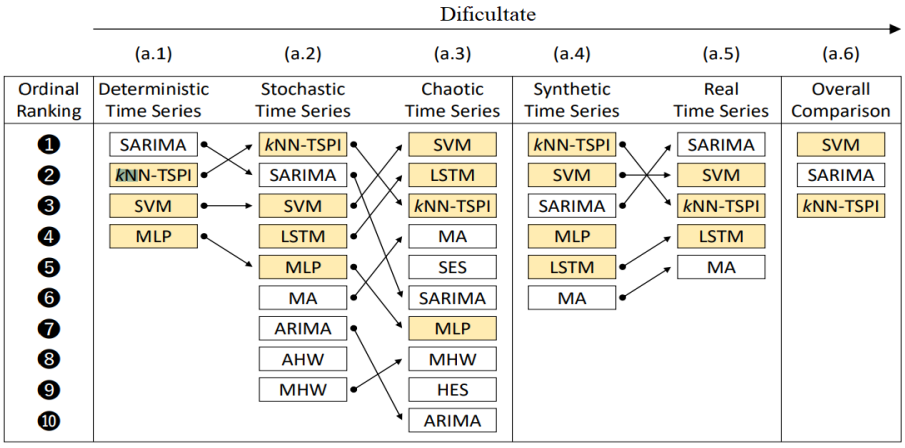

Figura 2.10: Rezultate metode de predicție 1[9].

Fig. 2.10 prezintă rezultatele metodelor pe serii de timp deterministe, stocastice, haotice, sintetice și reale, făcând o predicție pe un interval de timp de mai multe eșantioane. După fiecare valoare nouă prezisă, acea valoare este luată în considerare pentru următoarea predicție. Se poate observa că dintre metodele bazate pe statistică, metoda cu cele mai bune performanțe este *SARIMA*.

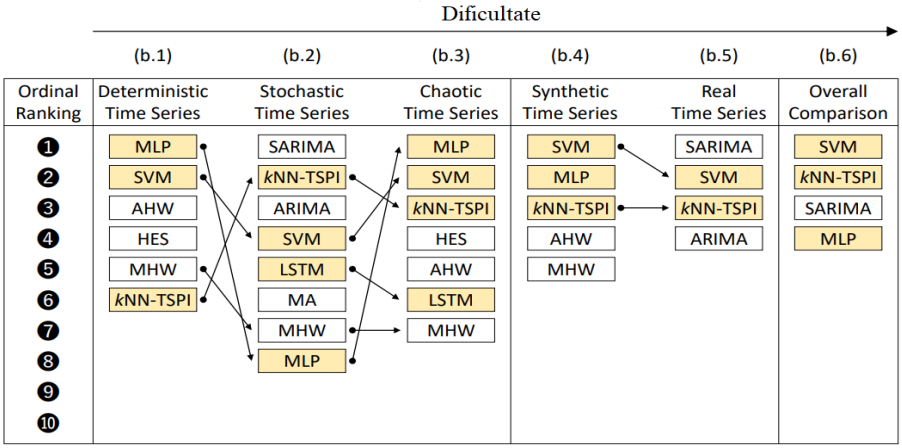

Figura 2.11: Rezultate metode de predicție 2[9].

Atunci când în loc de noua valoare prezisă s-a folosit valoarea reală din seria de timp pentru prezicerea următoarei valori, se observă același lucru: dintre metodele statistice, *SARIMA* oferă rezultatele cele mai bune. Mai mult decât atât, în ambele cazuri, *SARIMA* oferă cele mai bune rezultate dintre toate metodele atunci când seria de timp este una reală, preluată din mediu. Acestea fiind spuse, *SARIMA* pare să fie candidatul optim pentru această lucrare. Deci, în continuare, vor fi prezentate caracteristicile acestei metode.

2.2.1 Caracteristicile modelului

Modelul de tip SARIMA este compus din mai multe modele independente, fiecare dintre ele încercând să prezică câte o parte a seriei de timp. Aceste componente sunt:

1. AR: Autoregresie,
2. MA: Medie alunecătoare,
3. I : Integrare,
4. S : Sezonalitate.

În continuare vor fi prezentate trăsăturile cheie ale fiecărei părții a modelului, precum și modul de aflare al ordinului fiecărei părți utilizând noțiunile statistice discutate anterior.

2.2.2 Autoregresie (AR)

Modelul autoregresiv, după cum sugerează și numele său, pleacă de la premisa că valoarea actuală a seriei de timp depinde liniar de valorile sale anterioare. Din punct de vedere matematic, formula[3] care descrie modelul AR este:

$$y_t = c + \sum_{i=1}^{p}(\phi_i y_{t-i}) + \epsilon_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t, \tag{2.14}$$

unde y_t = valoarea curentă a seriei; c = constantă; p = ordinul modelului; y_{t-i} = valoarea seriei la momentul precedent momentului curent cu i eșantioane; ϕ_i = coeficientul momentului precedent momentului curent cu i eșantioane. Interpretarea informală și intuitivă a acestui model constă în faptul că modelul depinde doar de valorile sale anterioare. În literatură, modelul se scrie *AR(k)*, unde k reprezintă ordinul, adică numărul de eșantioane precedente momentului curent. Se consideră un model AR(1):

$$y_t = c + \phi_1 y_{t-1}, \tag{2.15}$$

adică modelul depinde doar de momentul precedent lui. Se consideră pentru ϕ valorile 0.8, respectiv -0.8 și se vizualizează graficele generate:

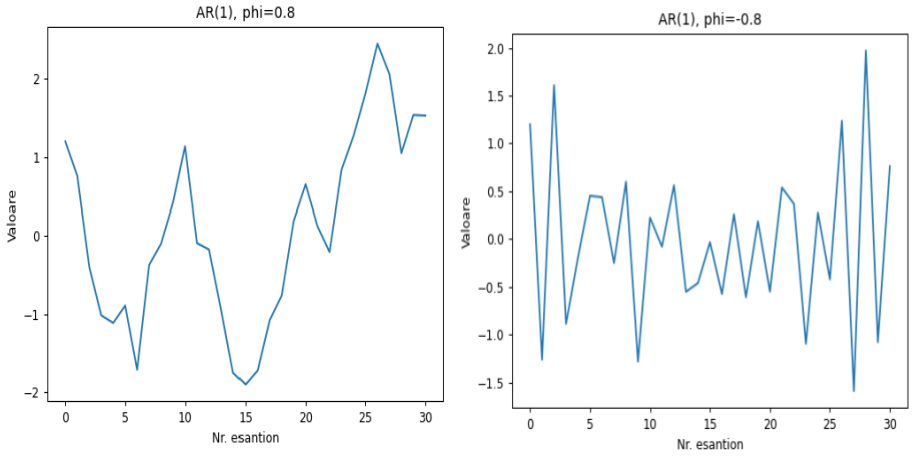

Figura 2.12: AR(1): a) $\phi = 0.8$ b) $\phi = -0.8$.

Se observă că atunci cand ϕ este pozitiv, graficul tinde să își păstreze direcția de variație a valorilor, iar atunci când ϕ este negativ, graficul își schimbă semnul aproape la fiecare nou eșantion. Acest lucru este în conformitate cu formula (2.15), semnul lui ϕ influențând semnul valorilor seriei. Pentru a identifica ordinul modelului AR optim pentru o serie de timp dată, se apelează autocorelația parțială. Atunci când se dorește identificarea modelului optim AR, defapt se dorește să se cunoască de câte eșantioane din trecut depinde eșantionul curent. Motivul pentru care se folosește autocorelația parțială, ci nu autocorelația este următorul: se dorește izolarea influenței fiecărui eșantion y_{t-i} asupra eșantionului y_t de restul eșantioanelor și vizualizarea influenței sale pe un grafic. Fiind

vorba despre un model AR, există o recursivitate asupra influenței eșantioanelor trecute asupra celui curent[10]. De aceea autocorelația nu este suficientă pentru a realiza această separare. Pentru a înțelege mai bine acest lucru se va considera un model AR(1):

$$\begin{aligned} y_t &= c + \phi_1 y_{t-1} \\ y_{t-1} &= c + \phi_1 y_{t-2} \\ y_t &= c + c\phi_1 + \phi_1^2 y_{t-2} \\ y_t &= c' + \phi_1^2 y_{t-2}. \end{aligned} \tag{2.16}$$

După cum se observă din ecuația (2.16), cu toate că modelul spune clar că y_t depinde doar de y_{t-1}, din calculul matematic rezultă că defapt acesta depinde și de eșantioanele anterioare. Având în vedere că $\phi \in [-1; 1]$, influența eșantioanelor anterioare va scădea pe măsura ce acestea se depărtează de eșantionul curent. Spre exemplu: se consideră $\phi = 0.8$:

$$y_t = c + 0.8y_{t-1} + 0.64y_{t-2} + \cdots + 0.8^n y_{t-n}. \tag{2.17}$$

Acest lucru se observă foarte bine pe corelogramă, deoarece la autocorelație, influența eșantioanelor anterioare nu este eliminată din autocorelația a două eșantioane. Prin urmare, atunci când plotăm o corelogramă a unui model AR, se va observa o descreștere exponențială, iar vizual un observator nu își va putea da seama de ordinul modelului AR. În Fig. 2.13 este afișată o asemenea corelogramă pentru un model AR(1):

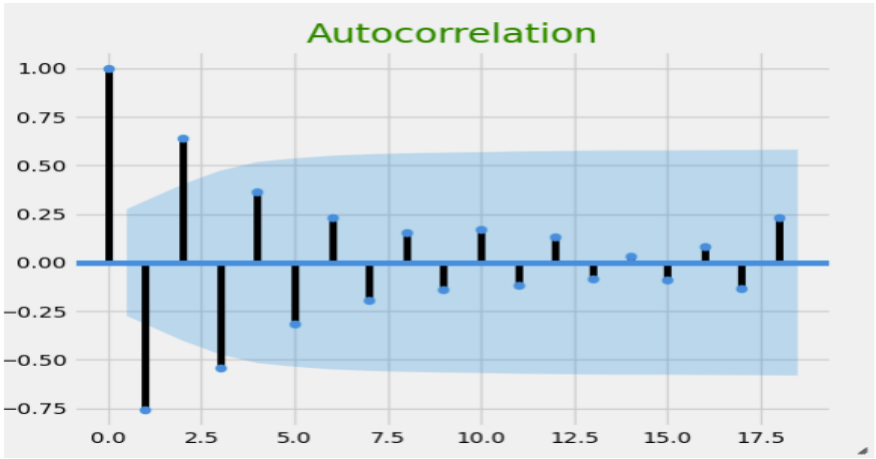

Figura 2.13: AR(1): Corelogramă.

Pentru a putea identifica în mod corect ordinul modelului se va apela autocorelația parțială pentru a elimina influența eșantioanelor nedorite. Deoarece în autocorelația parțială nu se ține cont de valorile efective anterioare, ci doar de erorile acestora, recurența din formula (2.15) va fi eliminată. În Fig. 2.14 sunt prezentate corelogramele parțiale ale unui model AR(1), respectiv AR(2):

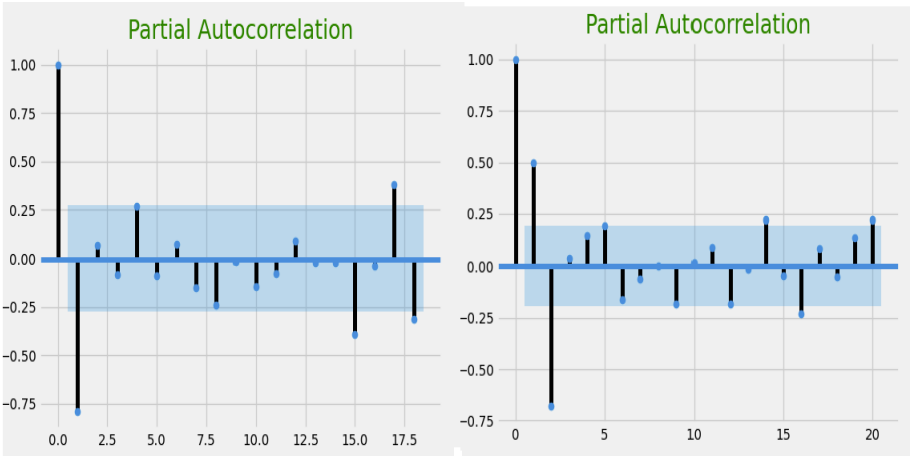

Figura 2.14: Corelogramă parțială: a) AR(1) b) AR(2).

Din Fig. 2.14 se poate determina foarte clar, chiar și cu ochiul liber, ordinul modelului.

2.2.3 Medie alunecătoare (MA)

Modelul de medie alunecătoare pleacă de la premisa că valoarea actuală a seriei de timp depinde liniar de erorile de predicție anterioare. Din punct de vedere matematic, formula[3] care descrie modelul MA este:

$$y_t = c + \sum_{i=1}^{p}(\theta_i \epsilon_{t-i}) + \epsilon_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_p \epsilon_{t-p} + \epsilon_t, \tag{2.18}$$

unde y_t = valoarea curentă a seriei; c = constantă; p = ordinul modelului; ϵ_i = valoarea erorii la momentul precedent momentului curent cu i eșantioane; θ_i = coeficientul erorii momentului precedent momentului curent cu i eșantioane. După cum se poate observa, modelul MA seamănă foarte mult cu modelul AR, ele diferind prin faptul că modelul MA ține cont de erorile anterioare ale predicției, în loc de valorile anterioare ale predicției. Așa cum a fost precizat și mai sus, seriile de timp care provin din lumea reală sunt afectate de zgomote. Având zgomote, care nu pot să fie modelate, este intuitiv că un simplu model care să țină cont de valorile anterioare ale seriei este insuficient. Apare nevoia de modelare a erorilor predicțiilor anterioare. Pentru a putea determina corect ordinul modelului MA, se va proceda în oglindă față de determinarea ordinului AR referitor la autocorelație: se va folosi autocorelația în loc de autocorelația parțială, deoarece la autocorelația parțială se iau în considerare erorile datelor, exact cum la AR se luau în considerare valorile datelor la autocorelație. Spre deosebire de AR, descreșterea exponențială se observă mai greu pe graficul autocorelării, dar ea este încă prezentă[10]. În Fig. 2.15 este afișată o asemenea corelogramă parțială pentru un model MA(1):

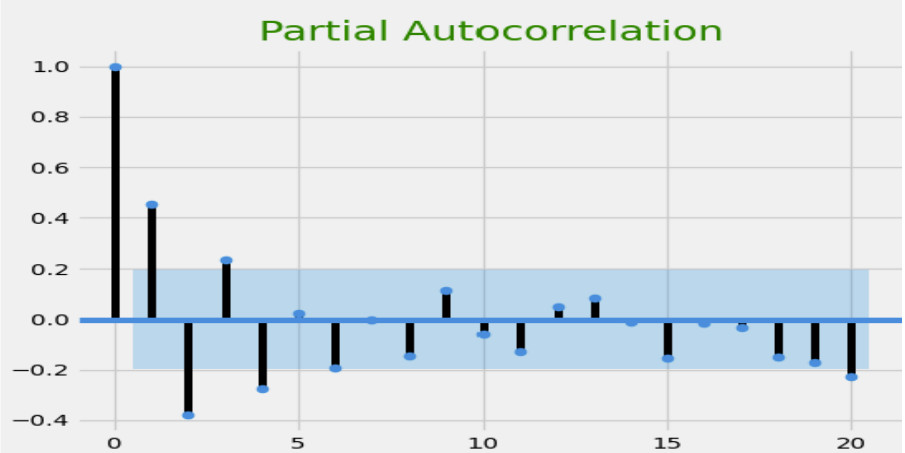

Figura 2.15: Corelogramă parțială MA(1).

Se poate observa mai ales pe semiplanul negativ acea descreștere exponențială despre care s-a vorbit mai sus. În schimb, corelograma este la fel de utilă ca și corelograma parțială pentru un proces AR. În Fig. 2.15 sunt prezentate corelogramele ale unui model MA(1), respectiv MA(2):

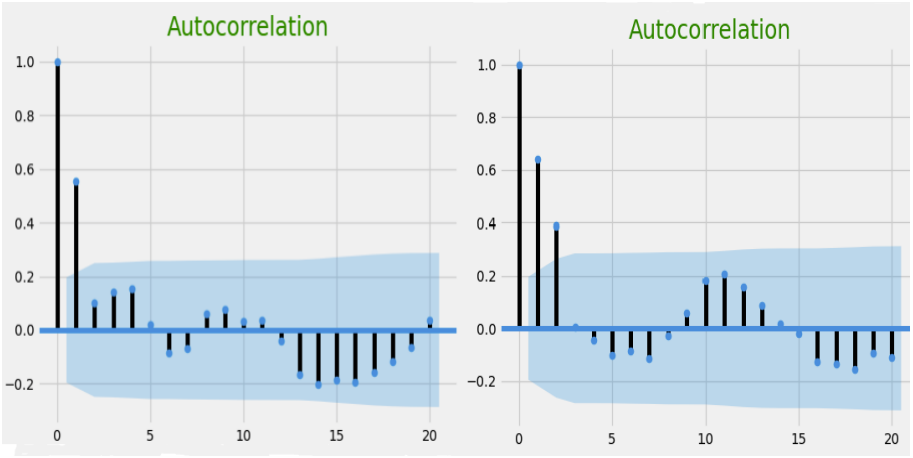

Figura 2.16: Corelogramă: a) MA(1) b) Ma(2).

Se poate observa cu ochiul liber o întrerupere bruscă a valorilor corelogramei dupa al doilea, respectiv al treilea eșantion, restul valorilor făcând parte din zona albastră care sugerează doar o întâmplare statistică.

2.2.4 Integrare (I)

Un mare neajuns al unui model de timp ARMA îl reprezintă staționaritatea datelor. Staționaritatea reprezintă o trăsătură cheie a seriilor de timp pentru predicție. Staționaritatea unei serii de timp este definită ca nemodificarea variației seriei de timp în raport cu

timpul[11]. Cu alte cuvinte, seria de timp își păstrează proprietățile indiferent de momentul de timp. Pentru a asigura o astfel de proprietate, trebuie asigurată o valoare medie constantă în timp, astfel varianța să nu se modifice în timp. Beneficiile staționarizării datelor se observă încă de la conceptul de deviație standard și până la conceptul de autocorelație. Asigurând o valoare medie egală cu 0 a datelor, formulele devin:

- Deviația standard: $\sigma = \sqrt{E[(X-\mu)^2]} = \sqrt{E[X^2]}$;
- Varianța: $\text{Var}(X) = E[(X-\mu)^2] = E[X^2] = \sigma^2$;
- Covarianța: $\text{Cov}(X,Y) = E[(X-E[X])(Y-E[Y])] = E[XY]$;
- Corelație: $\text{Corr}(X,Y) = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[XY]}{\sqrt{E[X^2]}\sqrt{E[Y^2]}}$;
- Autocovarianța: $\text{Cov}(X_t,k) = E[(X_t - E[X_t])(X_{t-k} - E[X_{t-k}])] = E[X_t X_{t-k}]$;
- Autocorelație: $\text{Corr}(X_t,k) = \frac{\text{Cov}(X_t,X_{t-k})}{\sigma_{X_t}\sigma_{X_{t-k}}} = \frac{E[X_t X_{t-k}]}{\sqrt{E[X_t^2]}\sqrt{E[X_{t-k}^2]}}$.

Două moduri foarte folosite pentru a obține date staționarizate sunt următoarele:

1. Eliminarea tendinței (*detrending*)[10]: Această metodă presupune găsirii unei regresii liniare care să aproximeze cât mai bine datele. Se consideră următoarea serie de timp dependentă de timp:

$$X_t = \alpha + \beta t + \sum_{i=1}^{p}(\phi_i Y_{t-i}) + \sum_{i=1}^{p}(\theta_i \epsilon_{t-i}) + c. \tag{2.19}$$

   Se consideră următoarea regresie:

$$Y'(t) = \alpha + \beta t. \tag{2.20}$$

   Funcția care reprezintă datele staționarizate va fi:

$$Z_t = X_t - Y'_t = \sum_{i=1}^{p}\phi_i Y'_{t-i} + \sum_{i=1}^{p}\theta_i \epsilon_{t-i} + c. \tag{2.21}$$

   Se poate observa că funcția Z_t se mulează perfect pe modelul ARMA. Această abordare se complică atunci când dependența de timp nu mai este liniară, regresia nemaiputând să aproximeze funcția de timp. Deci pentru a folosi această abordare trebuie cunoscută funcția ce determină dependența seriei în funcție de timp. Mai jos este prezentat în Fig. 2.17 un exemplu în care regresia este aleasă bine și în Fig. 2.18 un exemplu în care regresia este aleasă rău:

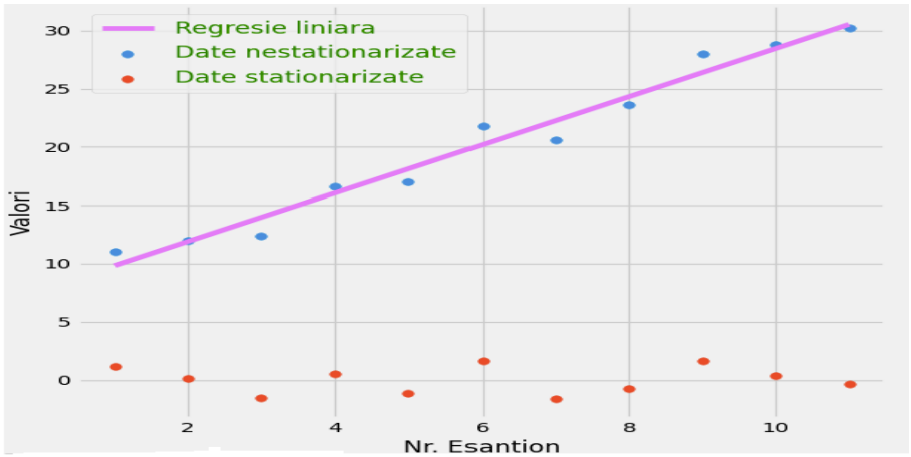

Figura 2.17: Regresie aleasă bine.

Se observă că datele staționarizate sunt toate centrate pe valoarea 0.

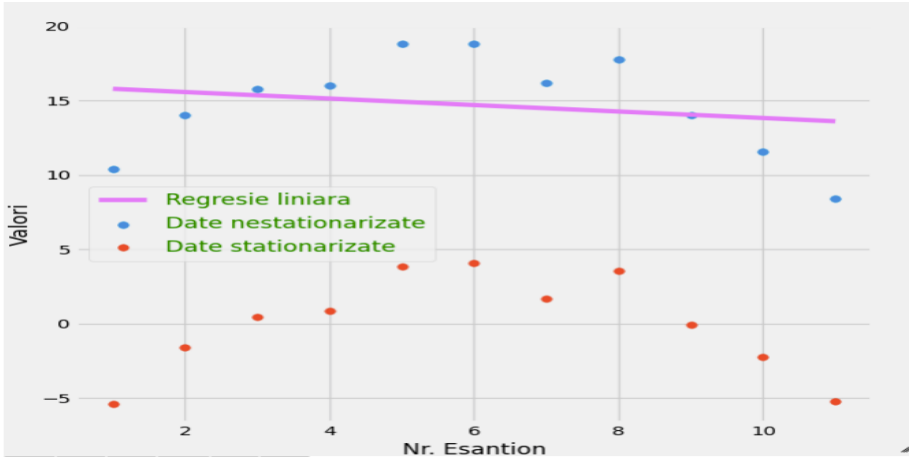

Figura 2.18: Regresie aleasă rău.

Se observă că datele "staționarizate" au la început o pantă pozitivă, iar apoi o pantă negativă. Deci, regresia liniară nu a putut să aproximeze foarte bine datele.

2. Diferențiere[10]: Procedeul de diferențiere se dovedește a fi mult mai stabil decât cel de eliminare a tendinței. Acest procedeu presupune, după cum îi spune și numele, lucrul cu diferențe în loc de valorile actuale ale seriei de timp. Prin urmare, funcția cu care se va lucra va fi:

$$Z_t = X_t - X_{t-1}. \tag{2.22}$$

Făcând această diferență se elimină la fiecare pas valoarea medie (comună) a valorilor implicate în diferență. Mai jos sunt reprezentate o serie de timp nediferențiată și aceeași serie de timp diferențiată pentru a observa creerea staționarității datelor:

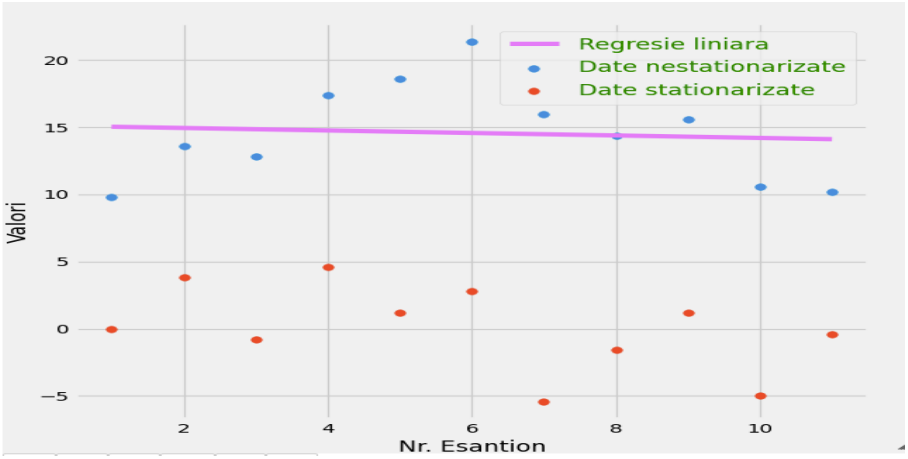

Figura 2.19: Diferențiere.

Se observă o centrare a datelor în jurul valorii 0 și păstrarea deviațiilor de valoarea medie a fiecărei valoare în parte. Avantajul acestei metode este că nu mai trebuie cunoscută dinainte funcția prin care depinde seria de timp de timp. Singurul lucru care se poate întâmpla și care trebuie luat în calcul este prezența unei derivate de ordin mai mare ca 1 a funcției. În acest caz, seria de timp trebuie diferențiată de un număr de ori egal cu ordinul maxim al derivatei nenule a funcției. Așadar partea de integrare are rolul de a diferenția seria de timp până când aceasta devine staționară. O diferențiere făcută de prea multe ori poate duce la pierdere de date, deci la scăderea acurateții predicției. În general, diferențierea se face de cele mai multe ori odată, iar în unele cazuri speciale de două ori. A diferenția de 3 ori un semnal este deja prea mult.

2.2.5 Sezonalitate(S)

Sezonalitatea are rolul de a acoperi neajunsurile integrării. În realitate există multe serii de timp care prezintă un șablon care se repetă odată la un interval de timp. Cele mai comune exemple sunt:

- Sezonalitate zilnică: consumul energiei dintr-o locuință(ziua consumul este mare, noaptea consumul este mic);
- Sezonalitate săptămânală: numărul de oameni care folosesc metroul(mare în timpul săptămânii, mic în weekend);
- Sezonalitate lunară: numărul plăților facturilor pentru utilități(mare după zilele tipice de salariu -începutul lunii-, mic în rest);
- Sezonalitate anuală: numărul de bomboane produse(mare iarna datorită sărbătorilor, mic în rest).

O serie de timp ce conține sezonalitate trebuie descompusă într-o serie asezonală și într-o serie ce conține doar sezonalitatea seriei originale. Partea de sezonalitate, repetându-se la intervale de timp mai mari decât trendul care variază după o funcție relativ simplă, trebuie și ea prezisă cu ajutorul unui model, la fel ca noua parte asezonală. De aceea, partea de sezonalitate a unui model nu introduce un singur parametru nou, ci 4 noi parametri[12]:

- ordinul modelului AR al sezonalității;
- ordinul modelului MA al sezonalității;
- ordinul modelului I al sezonalității;
- distanța dintre eșantioanele care se repetă(sezonalitate săptămânală + eșantionare zilnică $\Rightarrow$ distanța = 7).

Un alt efect al sezonalității se regăsește și în corelograma și corelograma parțială ale seriei de timp: vor apărea valori semnificative ale ambelor corelări la distanță egală cu numărul de eșantioane care se repetă. Spre exemplu: pentru o sezonalitate săptămânală și o eșantionare zilnică valoarea lui $\text{Corr}(X, 7)$ va fi semnificativ mai mare decât $\text{Corr}(X, 6)$ sau $\text{Corr}(X, 8)$.

2.2.6 Testul de staționarizare: Dickey-Fuller

De multe ori se poate întâmpla ca staționarizarea să nu fie suficient de evidentă pentru a se vedea cu ochiul liber. De aceea, este recomandat ca înainte de a găsi modelul pentru o serie de timp să se aplice testul Dickey-Fuller pentru a verifica staționarizarea și din punct de vedere matematic. Testul Dickey-Fuller pleacă de la un model de tip AR(1)[13]:

$$y_t = \phi y_{t-1} + \epsilon_t. \tag{2.23}$$

Pentru a putea testa staționarizarea trebuie mai întâi identificate componentele ce pot fi nestaționare. Acestea sunt:

- Coeficientul ϕ al lui y_{t-1}
- y_t și y_{t-1}

Condiția de staționaritate pentru ϕ: Se pleacă de la forma generală a lui AR(1):

$$y_t = \phi y_{t-1} + \epsilon_t \tag{2.24}$$

Se aplică varianța atât în stânga ecuației, cât și în dreapta:

$$\text{Var}(y_t) = \text{Var}(\phi y_{t-1} + \epsilon_t) \tag{2.25}$$

Se folosesc următoarele proprietăți ale variației:

1. $\text{Var}(\phi y_{t-1}) = \phi^2 \text{Var}(y_{t-1})$:

$$\begin{aligned}
\text{Var}(\phi y_{t-1}) &= E[(\phi y_{t-1} - E[\phi y_{t-1}])(\phi y_{t-1} - E[\phi y_{t-1}]] \\
&= E[(\phi y_{t-1} - \phi E[y_{t-1}])(\phi y_{t-1} - \phi E[y_{t-1}]] \\
&= E[(\phi(y_{t-1} - E[y_{t-1}]))(\phi(y_{t-1} - E[y_{t-1}])] \\
&= \phi\phi E[(y_{t-1} - E[y_{t-1}])(y_{t-1} - E[y_{t-1}]] \\
&= \phi^2 E[(y_{t-1} - E[y_{t-1}])(y_{t-1} - E[y_{t-1}]] \\
&= \phi^2 \text{Var}(y_{t-1}).
\end{aligned} \tag{2.26}$$

2. $\text{Var}(\phi y_{t-1} + \epsilon_t) = \phi^2 \text{Var}(y_{t-1}) + \sigma_\epsilon^2$:

$$\begin{aligned}
\text{Var}(\phi y_{t-1} + \epsilon_t) &= \phi^2 \text{Var}(y_{t-1}) + \text{Var}(\epsilon_t) + 2\phi \text{Cov}(y_{t-1}, \epsilon_t) \\
&= \phi^2 \text{Var}(y_{t-1}) + \text{Var}(\epsilon_t) \\
&= \phi^2 \text{Var}(y_{t-1}) + \sigma_\epsilon^2,
\end{aligned} \tag{2.27}$$

unde ϵ = variabilă aleatoare independentă de medie 0 și varianță $\sigma^2 \Rightarrow Cov(y_{t-1}, \epsilon_t) = 0$, deoarece variabilele nu au nicio legătură una cu cealaltă.

Se presupune că elementele seriei sunt staționarizate, deci $\Rightarrow Var(y_t) = Var(y_{t-1})$. Luând în considerare proprietățile 1 și 2 ale varianței si ipoteza staționarizării valorilor seriei, formula (2.25) este mai departe egală cu:

$$\begin{aligned}
&\text{Var}(y_t) = \phi^2 \text{Var}(y_{t-1}) + \sigma_\epsilon^2 = \phi^2 \text{Var}(y_t) + \sigma_\epsilon^2 \\
&\text{Var}(y_t)(1 - \phi^2) = \sigma_\epsilon^2 \\
&\text{Var}(y_t) = \frac{\sigma_\epsilon^2}{(1 - \phi^2)}.
\end{aligned} \tag{2.28}$$

Urmează calculul covarianței dintre y_t și y_{t-k}, dar mai întâi trebuie găsită relația dintre y_t și y_{t-k}:

$$\begin{aligned}
y_t &= \phi y_{t-1} + \epsilon_t, \qquad \epsilon_t = var.aleatoare(0, \sigma^2) \\
&= \phi(\phi y_{t-2} + \epsilon_{t-1}) + \epsilon_t = \phi^2 y_{t-2} + \phi\epsilon_{t-1} + \epsilon_t = \dots \\
&= \phi^k y_{t-k} + \sum_{i=0}^{k} (\phi^i \epsilon_{t-i}).
\end{aligned} \tag{2.29}$$

Deci, covarianța va fi egală cu:

$$\begin{aligned}
\text{Cov}(y_t, y_{t-k}) &= \text{Cov}(y_{t-k}, \phi^k y_{t-k} + \sum_{i=0}^{k} (\phi^i \epsilon_{t-i})) \\
&= \phi^k \text{Cov}(y_{t-k}, y_{t-k}) = \phi^k \text{Var}(y_{t-k}) \\
&= \frac{\phi^k \sigma_y^2}{1 - \phi^2}.
\end{aligned} \tag{2.30}$$

În formula (2.30) suma dispare deoarece conține doar termeni cu ϵ, variabilă aleatoare necorelată cu y. Pentru a afla autocorelația dintre y_t și y_{t-k} se aplică formula:

$$\text{Corr}(y_t, y_{t-k}) = \frac{\text{Cov}(y_t, y_{t-k})}{\text{Var}(y_{t-k})} = \phi^k. \tag{2.31}$$

Pentru a modela o serie de timp reală, întâlnită în natură sau în procese reale, seria de timp trebuie să fie slab dependentă, adică influența unui eșantion din trecut asupra eșantionului prezent să scadă cu cât eșantionul din trecut este mai depărtat de cel prezent. Cu alte cuvinte, autocorelația să scadă pe măsură ce indicele k crește. Acest lucru se dovedește a fi foarte intuitiv(a se vedea exemplul cu vremea menționat anterior). Din această condiție de slab dependență rezultă că:

$$|\phi| < 1. \tag{2.32}$$

Dacă ϕ este 1, atunci corelația seriei nu va scădea pe măsură ce k crește, iar dacă ϕ este mai mare ca 1 valorile seriei vor tinde spre infinit foarte repede[14]. Pentru a testa staționarizarea, testul presupune diferențierea datelor pentru a obține niște date(diferențe) staționarizate:

$$\begin{aligned} y_t - y_{t-1} &= \phi y_{t-1} - y_{t-1} + \epsilon_t \\ \Delta y_t &= (\phi - 1) y_{t-1} + \epsilon_t \\ \Delta y_t &= \gamma y_{t-1} + \epsilon_t. \end{aligned} \tag{2.33}$$

Valoarea lui ϕ poate fi maxim 1(a se vedea autocorelația), deci posibilele valori ale lui γ sunt cuprinse între 0 și -2. Când $-2 < \gamma < 0$ atunci seria de timp este staționară. Când $\gamma = -2, 0$, atunci seria de timp nu este staționară și, deci, mai trebuie diferențiată încă o dată, iar apoi reluat testul Dickey-Fuller pe noile diferențe.

2.2.7 Akaike Information Criterion(AIC)

Criteriul Informațional Akaike are rolul de a măsura cât de bine a reușit să prezică un model creat seria de timp care se dorea prezisă. Din punct de vedere matematic, criteriul este definit [16] astfel:

$$\text{AIC} = N \ln(\frac{\sum_{i=1}^{N} \varepsilon_i^2}{N}) + 2K, \tag{2.34}$$

unde N=numărul de observații realizate; K=numărul de parametrii ai modelului + 1; $\sum_{i=1}^{N} \varepsilon_i^2$ = suma erorilor ridicate la pătrat dintre datele reale și cele prezise. Spre exemplu, se consideră un model ARMA(2,2), se prezic 30 de date și se observă că suma erorilor este 150. Atunci $N = 30, K = 2 + 2 + 1 = 5$, iar AIC=$30 \ln(150/30) + 2 \cdot 5 = 91.241$. Atunci când se va căuta cel mai bun model, se va alege modelul cu valoarea AIC cea mai mică.

2.2.8 Bayesian Information Criterion(BIC)

Criteriul Informațional Bayesian sau Criteriul Informațional Schwarz este foarte asemănător cu Criteriul Informațional Akaike, el având exact același rol, dar o formulă de calcul diferită[16]:

$$\text{BIC} = N \ln(\frac{\sum_{i=1}^{N} \varepsilon_i^2}{N}) + K \ln(N), \tag{2.35}$$

unde N=numărul de observații realizate; K=numărul de parametrii ai modelului + 1; $\sum_{i=1}^{N} \varepsilon_i^2$ = suma erorilor ridicate la pătrat dintre datele reale și cele prezise. Formula este foarte asemănătoare cu cea a AIC, doar că BIC pedepsește mai mult numărul de parametrii suplimentari.

Capitolul 3

Metode de învățare automată

În acest capitol vor fi prezentate noțiunile de învățare automată, de la conceptele de bază, până la construirea modelului de predicție propriuzis.

3.1 Noțiuni introductive

Pentru a se înțelege mai bine metodele de învățare automată, se vor defini câteva concepte de bază ale acestui domeniu. Învățarea automată reprezintă un subdomeniu al inteligenței artificiale care presupune construcția de modele bazate pe date de antrenament cu rolul de a face predicții, de a face clasificări sau de a lua decizii. Tipul de învățare automată folosit în această lucrare se numește supervizat, adică întotdeauna, la antrenarea algoritmului de învățare automată, se vor da ca și date de antrenament atât intrările cât și ieșirile modelului ce trebuie construit[15]. Pentru a putea face o comparație cât mai complexă între metodele statistice și cele de învățare automată, alegerea făcută din domeniul învățării automate se va baza pe rețele neurale artificiale, cu scopul de a prezenta două abordări cu rezultate excelente din domeniul predicțiilor seriilor temporale, dar cu metode de implementare cât mai diferite. Numele de rețele neurale artificiale provine din asemănarea structurii acestor metode cu structura procesului de gândire uman. Metodele bazate pe rețele neurale încearcă să "mimeze" gândirea umană și modul de propagare a informație prin aceasta.

3.1.1 Neuronul artificial

Neuronul reprezintă structura de bază a unei rețelei neurale. Acesta este asociat cu un nod în care informația primită este prelucrată și trimisă mai departe. Entitatea de neuron artificial se aseamănă cu un neuron biologic din punct de vedere al modului de tratare a informației care trece prin acesta.

3.1.2 Perceptronul

Perceptronul reprezintă un concept matematic care nu este întâlnit în biologie. Acesta definește cea mai simplă arhitectură a unei rețele neurale: o rețea neurală cu un singur neuron[17]. De multe ori termenul de perceptron este confundat cu termenul de neuron și invers, diferența fiind că un neuron face parte din structura unui perceptron , dar un perceptron conține și alte elemente pe lângă acel neuron. În Fig. 3.1 este prezentată structura unui asemenea perceptron.

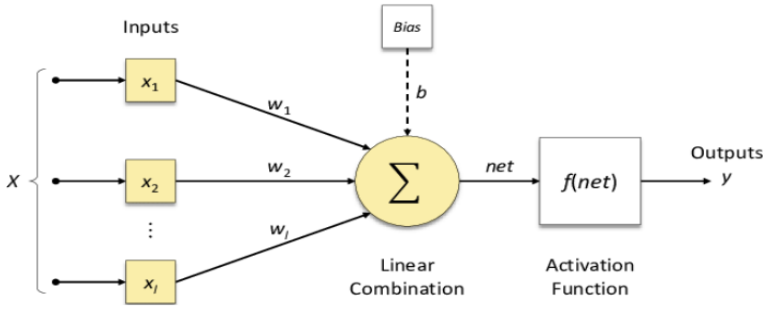

Figura 3.1: Structură perceptron[9].

În figura de mai sus se pot identifica câteva părți componente care vor fi explicate în continuare:

3.1.2.1 Datele de intrare

Datele de intrare notate în Fig. 3.1 prin X_i reprezintă valori numerice ale semnalului de intrare, fie că este vorba despre o imagine, un semnal audio, o serie de timp sau orice alt tip de semnal. Aceste valori vor fi folosite pentru construirea modelului (date de antrenament) și pentru testarea modelului (date de test). Dificultatea principală în formarea datelor de intrare o reprezintă lungimea fixă a acestora: pentru a folosi o rețea, lungimea lui X trebuie să fie constantă pntru toate datele de antrenare și testare. Această dificultate se reflectă în uniformizarea datelor. De exemplu: redimensionarea imaginilor, eșantionarea unui semnal cu o frecvență constantă, adăugarea completărilor până la dimensiunea dorită (*padding*).

3.1.2.2 Ponderi

Ponderile reprezintă influența fiecărei intrări asupra ieșirii. Dacă neuronii sunt asociați cu niște noduri, ponderile sunt asociate cu niște muchii, astfel încât întrega rețea este asociată cu un graf orientat.

3.1.2.3 Bias

Bias-ul reprezintă o pondere care nu este asociată cu nicio intrare. Rolul bias-ului într-o rețea este de a modifica valoarea obținută din datele de intrare și din ponderi pentru a putea oferi o predicție cât mai bună modelului respectiv.[17]

3.1.2.4 Funcția de activare

Funcția de activare are rolul de a determina ieșirea dintr-o rețea. Cea mai importantă trăsătură a funcțiilor de activare o reprezintă neliniaritatea[17]. Această trăsătură este esențială rețelelor neurale, deoarece fără această neliniaritate, rețelele nu ar fi nimic mai mult decât o simplă regresie liniară. Tocmai acest lucru diferențiază rețelele neurale de multe dintre metodele de predicție. Prin intermediul bias-ului se garantează că ieșirea rețelei nu va putea fi reprodusă print-o funcție liniară. Un alt mod de a privi importanța funcțiilor de activare este acela că fără acestea toate straturile rețelei ar putea fi combinate într-un singur strat care să facă o combinație liniară a intrărilor și din care să rezulte ieșirea rețelei. Dar cu aceste funcții, straturile nu se pot combina, ieșirea nu este doar o simplă combinație liniară a intrărilor, iar puterea de predicție a rețelei este mult mai mare, ea antrenându-se să prezică o funcție neliniară. Există mai multe tipuri de funcții de activare. În Fig 3.2 sunt reprezentate cele mai folosite dintre acestea:

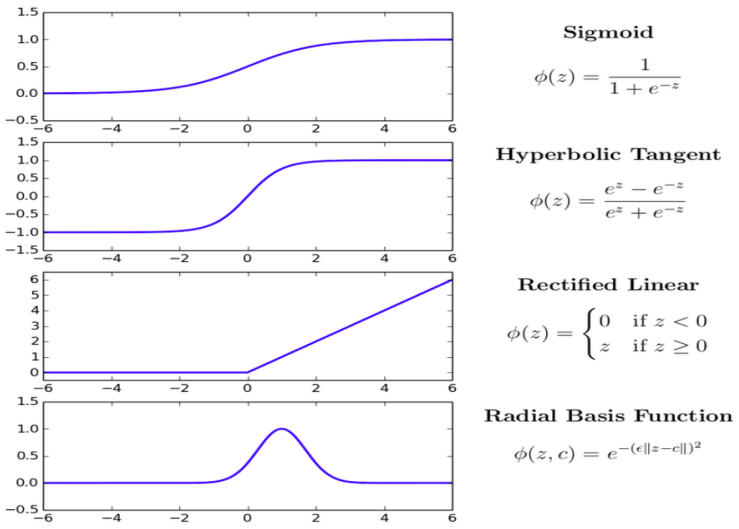

Figura 3.2: Funcții de activare.

Dintre funcțiile de mai sus, pentru a păstra simplitatea rețelei, mai ales când se lucrează cu date de dimensiuni foarte mari, se recomandă folosirea funcției *Rectified Linear (ReLU)*. Pentru stratul de ieșire al rețelei se recomandă în funcție de scopul rețelei o funcție de activare liniară pentru regresii deoarece se dorește scalarea ieșirii la valorile adevărate ale datelor sau funcții care să aibă codomeniul bine definit în sensul restricționării valorilor posibile la un interval mic și fix: de exemplu [0,1]. Atunci când se dorește o clasificare

cu mai mult de două clase se poate folosi o funcție de activare *softmax*:

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{nr.clase} e^{y_j}}, \tag{3.1}$$

unde y_i = valoare de ieșire din perceptron de pe poziția corespunzătoare clasei corecte; y_j = valoare de ieșire din perceptron corespunzătoare fiecărei clase.

3.1.2.5 Combinația liniară

Combinația liniară reprezintă adunarea influențelor tututor intrărilor și a bias-ului. Ea servește drept intrare funcției de activare care are rolul să introducă neliniaritate acesteia. Formula prin care se definește combinația liniară din cadrul unui neuron este[17]:

$$z = \sum_{i=1}^{l} w_i x_i + b, \tag{3.2}$$

unde z = rezultatul combinației liniare; l = numărul de intrări în rețea; x_i = intrarea i în rețea; w_i = ponderea intrării i în rețea; b = bias. O interpretare intuitivă a bias-ului este asocierea acestuia cu un concept mult mai cunoscut și anume termenul constant dintr-o regresie liniară.

3.1.2.6 Datele de ieșire

Datele de ieșire reprezintă rezultatul combinației liniare trecut prin funcția de activare. Formula pentru datele de ieșire[17] este:

$$y = f(\sum_{i=1}^{l} w_i x_i + b), \tag{3.3}$$

unde f = funcția de activare. În literatura de specialitate este notată cu σ.

3.1.3 Rețele neurale

O rețea este compusă din mai mulți neuroni. Neuronii pot fi amplasați pe un singur strat sau pe mai multe straturi. O rețea neurală cu mai multe straturi ascunse se numește rețea neurală adâncă (*Deep Neural Network*). Pentru următoarele formule se va folosi o asemenea rețea adâncă pentru a putea generaliza mai mult formulele matematice. În afară de numărul mai mare de neuroni, o rețea neurală are exact aceeași structură ca un perceptron. De aceea este greșită identitatea perceptron-neuron, fiindcă într-o rețea se modifică doar numărul de noduri interne, nu și structura per ansamblu. În Fig. 3.3 este prezentată structura unei rețele neurale adânci:

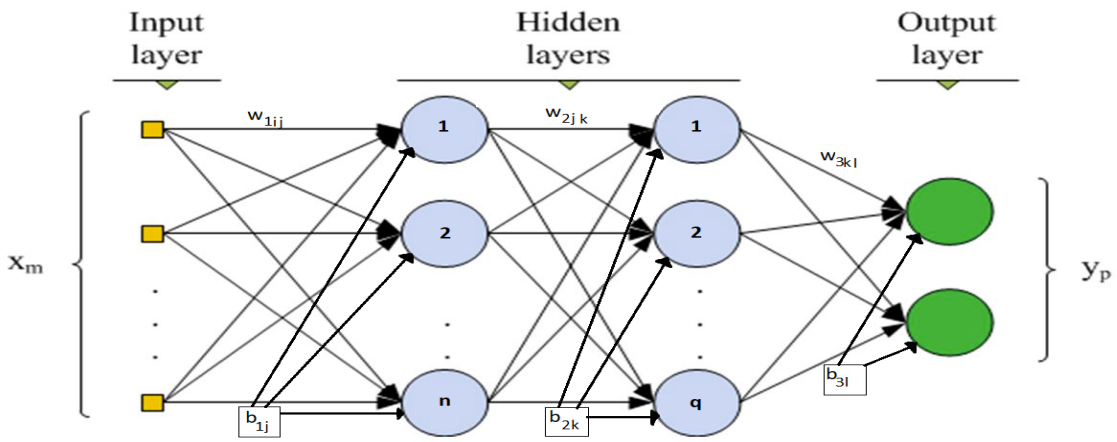

Figura 3.3: Rețea neurală adâncă.

Generalizând formulele pentru perceptron se obține următoarea formulă pentru ieșirea din rețea[17]:

$$y_l = f_1(\sum_{k=1}^{q} w_{1kl} f_2(\sum_{j=1}^{n} w_{2jk} f_3(\sum_{i=1}^{m} w_{3ij} x_i + b_{1j}) + b_{2k}) + b_{3l}), \tag{3.4}$$

unde y_l = ieșirea l; f = funcția de activare; w_{ij}= ponderea provenita de la nodul i a nodului j; b_{1j} = bias-ul nodului j de pe stratul 1.

3.1.3.1 Funcție de cost

Funcția de cost are rolul de a calcula eroarea generată de rețea în raport cu valorile adevărate ale ieșirii. Pentru a realiza acest lucru există o varietate largă de funcții de calcul al erorii, funcția cea mai bună diferind de la problemă la problemă, de la set de date la set de date. De regulă, pentru problemele de clasificare una dintre funcțiile cele mai generale de la care se pleacă este *Cross Entropy*[1]. :

- Clasificare binară
  Metoda se bazează de calculul entropiei dintre variabila reală și cea prezisă. Dacă cele două variabile sunt egale, atunci entropia dintre ele va fi egală cu entropia fiecăreia dintre ele. Formula (3.5) prezintă definirea acesteia:

$$\text{CrossEntropy} = -\frac{1}{N}\sum_{i=0}^{N}(y_i \ln(\hat{y}_i) + (1 - y_i)\ln(1 - \hat{y}_i)), \tag{3.5}$$

  unde y reprezintă valoarea reală(0 sau 1), iar $\hat{y}$ reprezintă valoarea prezisă.

[1]Douglas M. Kline, "Revisiting squared-error and cross-entropy functions for training neural network classifiers",Neural Comput And Applic, 2005

- Clasificare cu mai multe clase
  Funcția de *Cross Entropy* devine atunci când sunt prezente mai multe clase:

$$\text{CrossEntropy} = -\ln(\frac{e^{s_i}}{\sum_j^C e^{s_j}}) \tag{3.6}$$

  ,unde C reprezintă numărul de clase, s_i reprezintă probabilitatea prezisă de apartenentă la clasa adevărată a valorii respective. Dacă se exclude din formula de mai sus logaritmul și semnul "-", funcția devine egală cu funcția *softmax*, o altă funcție de pierdere foarte des întâlnită.

Pentru problemele de regresie, se consideră drept funcție de cost una dintre funcțiile generale care trebuie minimizate pentru găsirea unei regresii la metodele statistice. Câteva exemple de asemenea funcții sunt[2]. :

$$\begin{aligned}
\text{Mean Square Error(MSE)} &= \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \\
\text{Mean Absolute Error(MAE)} &= \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \\
\text{Mean Square Logarithmic Error(MSLE)} &= \frac{1}{N}\sum_{i=1}^{N}(\ln(y_i+1) - \ln(\hat{y}_i+1))^2.
\end{aligned} \tag{3.7}$$

Pe caz general, funcția de cost se poate scrie ca $L(\hat{y}, y)$. Fiecare dintre funcțiile de mai sus se folosește în diverse cazuri, necesitatea lor depinzând în întregime de structura datelor. De exemplu dacă datele au o deviație standard foarte mare și nu se dorește penalizarea puternică a valorilor prea mari, nu se va folosi *MSE*, ci *MSLE* deoarece aplicând funcția logaritm, valorile mari nu vor fi la fel de puternic penalizate. Un alt caz îl reprezintă prezența valorilor atât foarte mari, cât și foarte mici în jurul valorii medii. În acest caz se preferă *MAE* în detrimentul *MSE*. Dacă nu se cunosc prea multe informații despre date, cea mai sigură metodă pe caz general este *MSE*. Un efect imediat al valorii funcției de cost îl reprezintă stabilirea numărului de epoci de rulare: în practică, numărul de epoci trebuie ales astfel încât funcția de cost să nu mai descrească brusc. Un număr prea mic de epoci înseamnă că modelul putea să fie mai bine antrenat, iar un număr prea mare de epoci înseamnă că se crește riscul de *overfitting*[3]. *Overfitting*-ul este definit ca o supra-antrenare a datelor, astfel încât rețeaua în loc să învețe doar trăsăturile importante ale datelor, asociază pe dinafara o valoare de ieșire cu o valoare de intreare, așa că atunci când se vor da date noii rețele, pe care aceasta nu s-a antrenat, nu va da rezultate bune. Cel mai ușor mod de a identifica *overfitting*-ul este prin compararea acurateții pe datele de antrenament vs. pe datele de test: o acuratețe mare pe datele de antrenament + o acuratețe mică pe datele de test $\Rightarrow$ *Overfitting*.

[2]MSE:`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html`
MAE:`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html`
MSLE:`https://www.tensorflow.org/api_docs/python/tf/keras/losses/MSLE`

[3]Overfitting:`https://dl.acm.org/doi/pdf/10.1145/212094.212114`

3.1.3.2 Propagare înapoi

Propagarea înapoi reprezintă metoda prin care rețeaua neurală "învață". Diferența dintre rezultatul ieșirii rețelei și rezultatul real este influențată de ponderile fiecărui neuron. Așadar, pentru a avea un rezultat mai bun, acele ponderi trebuie modificate până când rezultatul real și cel prezis coincid. Coinciderea lor, înseamnă defapt că diferența dintre ele este minimă. Așadar pentru a putea calcula acest lucru se vor folosi metode bazate pe scăderea gradientului *Gradient descent*[17]. Aceste metode au rolul de a minimiza funcția de cost a rețelei, adică de a minimiza diferența dintre rezultatele reale și cele prezise. Primul lucru necesar pentru înțelegerea propagării înapoi este scrierea sub formă vectorizată a formulei (3.4):

$$\hat{y} = f(W_1 f(W_2 f(W_3 X + b_1) + b_2) + b_3, \tag{3.8}$$

unde $W_{1,2,3}$ = ponderile de pe straturile 1,2,3; $b_{1,2,3}$ = bias-urile de pe straturile 1,2,3; L = funcția de cost. În acest caz, problema care trebuie minimizată este:

$$\min_{W,b} \sum_{i=1}^{n} L(\hat{y}_i, y_i) = f, \tag{3.9}$$

unde n = numărul de date de intrare. În continuare se vor folosi notațiile z_k pentru a defini vectorul rezultatelor regresiilor liniare ale neuronilor de pe stratul k și a_k pentru a defini vectorul cu valorile de ieșire ale neuronilor de pe stratul k. Este nevoie de calcularea fiecărui vector de ponderi și de bias-uri de pe toate straturile. Pentru a face acest lucru se vor folosi derivatele parțiale și se va merge din aproape în aproape, recursiv, pentru a calcula toate derivatele de care este nevoie. În Fig. 3.4 sunt reprezentate dependințele funcțiilor din interiorul rețelei, a ponderilor și a bias-urilor pentru o mai bună înțelegere a noțiunilor care urmează:

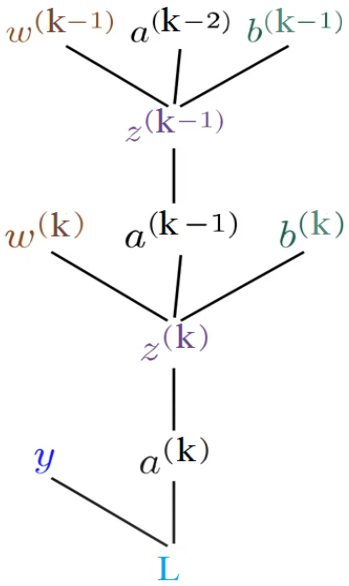

Figura 3.4: Dependințe funcții din rețea.[4]

Algoritm propagare înapoi:

1. Dependința funcție de cost - activare ultimul strat
   Se cunoaște că funcția de cost depinde de activările de pe ultimul strat ale neuronilor, adică de a_k, unde k reprezintă numărul de straturi. Așadar se poate scrie:

$$\frac{\partial f(a_k, y)}{\partial(a_k)}. \tag{3.10}$$

2. Dependința funcție de activare - funcție de combinație liniară
   Se cunoaște că la fiecare neuron, ieșirea din acesta reprezintă rezultatul combinației liniare (z_k) trecut prin funcția de activare neliniară. Așadar se poate scrie:

$$\frac{\partial(a_k)}{\partial(z_k)}. \tag{3.11}$$

3. Dependința funcție de combinație liniară - pondere & bias
   Se cunoaște faptul că funcția de combinație liniară depinde de ponderile stratului respectiv și de bias-urile acestuia. Așadar se pot scrie relațiile:

$$\begin{aligned} &\frac{\partial(z_k)}{\partial(w_k)}, \\ &\frac{\partial(z_k)}{\partial(b_k)}. \end{aligned} \tag{3.12}$$

4. Dependența funcție de cost - pondere & bias
   Aceasta este dependența de interes, scopul fiind observarea variației funcției de cost în funcție de ponderi și de bias-uri. Folosind pașii anteriori din algoritm, se poate deduce că:

$$\begin{aligned} \frac{\partial f(a_k, y)}{\partial(w_k)} &= \frac{\partial f(a_k, y)}{\partial(a_k)} \frac{\partial(a_k)}{\partial(z_k)} \frac{\partial(z_k)}{\partial(w_k)}, \\ \frac{\partial f(a_k, y)}{\partial(b_k)} &= \frac{\partial f(a_k, y)}{\partial(a_k)} \frac{\partial(a_k)}{\partial(z_k)} \frac{\partial(z_k)}{\partial(b_k)}. \end{aligned} \tag{3.13}$$

5. Dependența funcție de combinație liniară - activare strat precedent
   Combinația liniară depinde de ponderile stratului, de bias-ul stratului și de valorile stratului anterior. Așadar, se poate scrie relația:

$$\frac{\partial(z_k)}{\partial(a_{k-1})}. \tag{3.14}$$

6. Recursivitate
   Aplicând toți pașii anteriori împreună, se ajunge la o formulă recurentă din care se

[4]Imaginea este preluată din cadrul `https://www.youtube.com/watch?v=tIeHLnjs5U8&t=395s`

pot determina toate derivatele parțiale de care este nevoie. Mai jos sunt prezentate formulele pentru dependența variației funcției de minimizat în raport cu ponderile și bias-urile de pe un strat oarecare notat $k-p$[17]

$$\frac{\partial f(a_k,y)}{\partial(w_k)} = \frac{\partial f(a_k,y)}{\partial(a_k)}\frac{\partial(a_k)}{\partial(z_k)}\frac{\partial(z_k)}{\partial(a_{k-1})}\frac{\partial(a_{k-1})}{\partial(z_{k-1})}\frac{\partial(z_{k-1})}{\partial(a_{k-2})}\cdots\frac{\partial(z_{k-p+1})}{\partial(a_{k-p})}\frac{\partial(a_{k-p})}{\partial(z_{k-p})}\frac{\partial(z_k)}{\partial(w_{k-p})},$$
$$\frac{\partial f(a_k,y)}{\partial(b_k)} = \frac{\partial f(a_k,y)}{\partial(a_k)}\frac{\partial(a_k)}{\partial(z_k)}\frac{\partial(z_k)}{\partial(a_{k-1})}\frac{\partial(a_{k-1})}{\partial(z_{k-1})}\frac{\partial(z_{k-1})}{\partial(a_{k-2})}\cdots\frac{\partial(z_{k-p+1})}{\partial(a_{k-p})}\frac{\partial(a_{k-p})}{\partial(z_{k-p})}\frac{\partial(z_{k-p})}{\partial(b_{k-p})}. \tag{3.15}$$

Așadar, prin propagarea înapoi a derivatelor parțiale se pot modifica ponderile și bias-urile rețelei pentru a minimiza funcția de cost. După cum se observă din calculele de mai sus, alegerea funcțiilor de cost și de activare influențează în mod direct nu numai calculul derivatelor, ci și viteza de calcul a derivatelor. De aceea, atunci când se lucrează cu foarte multe date se recomandă alegerea unor funcții de activare cât mai simple pentru a nu îngreuna mult rețeaua la calculul derivatelor parțiale. De exemplu o funcție al cărei gradient este foarte simplu de calculat este *ReLU*.

3.1.3.3 Optimizarea propagării înapoi

Având în vedere calculul extrem de complex din punct de vedere al puterii computaționale, cu timpul au început să apară metode de optimizare al calculului gradienților pentru a crește viteza de procesare. Mai jos sunt prezentate câteva dintre metodele de optimizare cele mai des întâlnite. Se vor considera notațiile: $L_{i,j}(\theta)$ reprezintă funcția de cost care trebuie minimizată pentru intrarea j în rețea la epoca i, unde $\theta = (w_i, b_i)^T$. În continuare sunt prezentate câteva metode de bază ale *gradient descent*[18]:

1. Batch(Vanilla) gradient descent
   Reprezintă metoda clasică, această metodă calculează gradientul o singură dată per epocă, adică pentru toate datele de input împreună.

$$\begin{aligned} &while||\theta_{k+1}-\theta_k|| \leq \epsilon \\ &\qquad \theta_{k+1} = \theta_k - \alpha_k(\frac{1}{n}\sum_{i=1}^{n}\nabla L(\theta_k, y_i)), \end{aligned} \tag{3.16}$$

   unde n=numărul de intrări în rețea. Problema acestei metode provine din alegerea parametrului α: un α prea mic conduce la timp de rulare foarte mare, iar un α prea mare poate face modelul să oscileze în jurul minimului fără să îl atingă.

2. Stochastic gradient descent
   Această metodă este mult mai rapidă decât Vanilla gradient descent, deoarece ea folosește o singură intrare în rețea per epocă pentru a calcula gredientul. Astfel,

metoda aceasta este mult mai rapidă, dar și mult mai haotică, valorile gradientului convergând mai lent la valoarea sa minimă. Formula pentru această metodă este:

$$\begin{aligned} &while||\theta_{k+1} - \theta_k|| \leq \epsilon \\ &\quad \theta_{k+1} = \theta_k - \alpha_k \nabla L(\theta_k, y_i), \end{aligned} \tag{3.17}$$

unde $i \in (1, n)$. i se poate alege fie la întâmplare de fiecare dată din mulțimea valorilor sale, fie se alege ciclic.

3. Mini-batch gradient descent
   Această metodă reprezintă un compromis între primele două metode: ea folosește un subset de date de intrare pentru fiecare epocă. Astfel se face un compomis asupra vitezei de calcul împreună cu viteza de convergență spre minimul gradientului.

$$\begin{aligned} &while||\theta_{k+1} - \theta_k|| \leq \epsilon \\ &\quad \theta_{k+1} = \theta_k - \alpha_k(\frac{1}{n}\sum_{i \in J} \nabla L(\theta_k, y_i)), \end{aligned} \tag{3.18}$$

   unde J reprezintă o submulțime a intrărilor în rețea. Pentru metoda de bază, viteza de învățare α poate să varieze sau poate să fie constantă, în schimb, la celălalte două metode, viteza de învățare trebuie să scadă după fiecare iterație pentru ca algoritmul să conveargă. O variantă mult mai eficientă, dar și mult mai complicată din punct de vedere matematic, de a calcula gradientul funcției de cost, o reprezintă Adam(Adaptive moment estimation). Această funcție este adesea folosită în practică datorită rezultatelor sale pe seturi de date generale.

3.2 Modelul de predicție

În continuare este prezentat modelul de predicție ales, de la motivația alegerii făcute, până la implementarea acestuia pe o serie de date.

3.2.1 Alegerea modelului

Pentru alegerea modelului de predicție bazat pe învățare automată s-a considerat aceeași sursă de inspirație ca și la metodele statistice[9]. Din Fig. 2.10 și 2.11 se poate observa că dintre metodele bazate pe învățare automată, cele mai bune rezultate au obținut *SVM* și *kNN-TSPI*. Cu toate acestea, pentru a obține o comparație cât mai amplă între cele două domenii: statistică și învățare automată, s-au considerat drept metode de învățare automată doar modelele care se bazează pe rețele neurale. Cum *SVM* și *kNN-TSPI* se bazează foarte mult pe termenul de regresie, se consideră că o comparație între două concepte cât mai diferite, dar cu același scop, cel de predicție, poate spune mai multe lucruri despre domeniile respective din care acestea fac parte. Având în vedere

că *SARIMA* diferă de SVM prin faptul că se consideră doar regresii liniare, în timp ce la *SVM* ele pot fi și neliniare, este mult mai interesantă comparația *SARIMA*-ei cu o metodă care să introducă rețele neurale.

3.2.2 Caracteristicile modelului

Dintre metodele bazate pe rețele neurale, două metode din studiul [9] dau rezultate bune: *LSTM* și *MLP*. Criteriul care a avantajat-o pe una dintre ele pentru a fi aleasă în acest studiu îl reprezintă numărul de date al seriei de timp. În articolele de specialitate, *LSTM*-ul dă rezultate mai bune decât *MLP* cu o condiție: numărul de date al seriei de timp să fie suficient de mare(de la ordinul milioanelor în sus)[20]. Pentru a face o comparație corectă cu *SARIMA*, am considerat că datele nu au lungimea mai mare de câteva mii-zeci de mii. Se pleacă de la această premisă întrucât pentru a rula *SARIMA* pe un număr mai mare de câteva zeci de mii de date durează extrem de mult. Așa că s-a preferat un număr de date mai mic pentru a acorda șanse egale ambelor abordări. *MLP*, având o arhitectură mult mai simplă decât *LSTM* poate să lucreze cu date mult mai puține și totuși să dea rezultate bune.

3.2.3 Multi Layer Perceptrons (MLP)

MLP reprezintă arhitectura clasică, generală, a unei rețele neurale. Din această arhitectură derivă o multitudine de arhitecturi folosite în diverse domenii. Arhitectura de bază constă în: un strat de intrare, un strat de ieșire și straturile ascunse, unde straturile ascunse pot fi[17]:

- Dense: Strat complet conectat. Cel mai întâlnit tip de strat din cadrul MLP.
- Dropout: Stratul Dropout setează o parte din inputurile stratului anterior lui la valoarea 0. Motivul pentru care există un asemenea strat este prevenirea overfitting-ului.
- Merge: Stratul merge combină inputurile din mai multe modele într-un singur model. Este cel mai rar tip de strat folosit dintre cele 3 straturi menționate. El este folosit în arhitecturi mai complicate ale modelelor.

Plecând de la aceste 3 straturi, se pot construi diverse arhitecturi, fiecare avându-și utilitatea în domenii specifice. De exemplu, un tip particular de arhitectură îl reprezintă rețelele convoluționale. Ce face o rețea neurală să fie convoluțională este prezența unor tipuri specifice de straturi cum ar fi: strat convoluțional, strat pooling, strat flatten. Fiecare dintre aceste straturi își are rolul său în transformarea datelor pentru a putea lucra cu convoluții. Domeniul convoluțiilor este specific prelucrării de imagini.

3.2.4 Prelucrarea datelor

Datele sunt reprezentate printr-un vector uni-dimensional. Primul lucru care trebuie făcut pentru a aduce datele sub o formă mai bună îl reprezintă normalizarea datelor. În principiu, orice normalizare a datelor este mai bună decât a utiliza datele nenormalizate, dar o normalizare recomandată pentru acest gen de problemă este normalizarea la varianță unitară și medie 0. În Fig. 3.5 este reprezentată o astfel de serie de timp normată la medie 0 și varianță unitară:

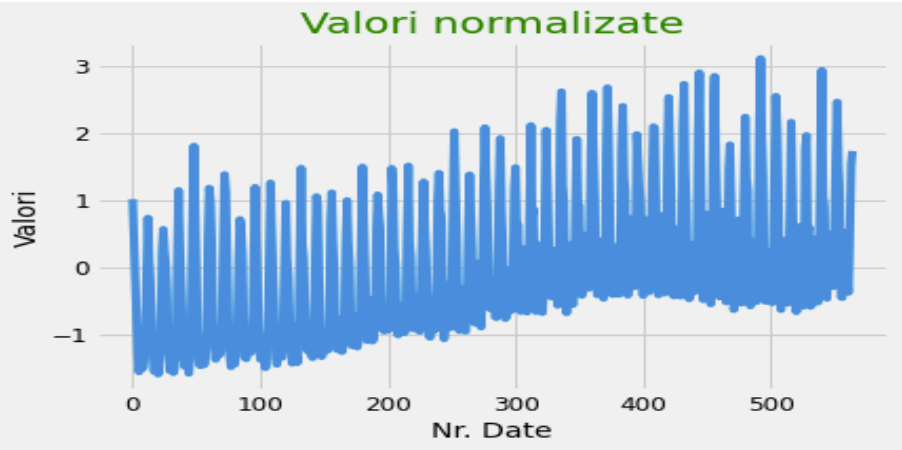

Figura 3.5: Date normalizate.

Principala modificare ce trebuie aplicată datelor pentru acest tip de problemă este creerea datelor de intrare și de ieșire din rețea. În primul rând, datele sunt împărțite în antrenament și test, exact la fel ca în Fig. 4.8. Apoi trebuie stabilită lungimea dorită la intrarea, respectiv la ieșirea rețelei. Pentru o predicție cu un singur pas în viitor, ieșirea va fi mereu 1. Intrarea pe de altă parte, reprezintă o variabilă care influențează acuratețea rețelei. Se presupune că intrarea are lungime *m*. Fiecare intrare în rețea va fi reprezentată de un vector uni-dimensional de dimensiune (m,1). Pentru a forma toate intrările în rețea se va pleca de la primele *m* valori ale setului de date, apoi se vor selecta pentru a doua intrare *m* valori decalate la dreapta cu o valoare față de primele. Valoarea de ieșire va fi întotdeauna a *m+1* valoare de la prima valoare de antrenament. Se parcurge setul de date până când valoare de test este egală cu ultima valoare din șirul de valori ale datelor de antrenament/test. Pentru o înțelegere mai bună a acestui procedeu, a se vedea Fig. 3.6:

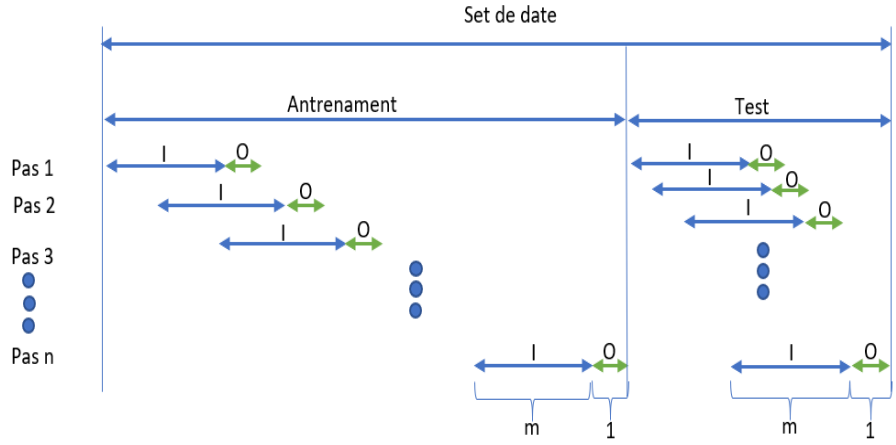

Figura 3.6: Structură date pentru MLP.

În felul acesta se vor forma două matrici: intrări antrenament și intrări test și doi vectori: ieșiri antrenament și ieșiri test. În Fig. 3.6, n este dat de lungimea datelor de antrenament/test împărțită la lungimea datelor de intrare+ieșire antrenament/test și reprezintă numărul de perechi intrare-ieșire pentru antrenament, respectiv test.

3.2.5 Configurarea rețelei

Pentru a obține rezultate cât mai bune trebuie variați destul de mulți parametri ai rețelei. O metodă tipică de a realiza acest lucru este *grid search*: se formează câteva *for*-uri cu valorile dorite pentru fiecare paramentru, se impune o metodă de cuantizare a acurateții (de exemplu *MSE*) și se rulează rețeaua de foarte multe ori pe toate cazurile. Dezavantajul acestei metode este că nu se pot interpreta foarte bine toate rezultatele. Pentru a putea trage mai multe concluzii despre variația paramterilor s-a ales variația manuală a parametrilor și interpretarea rezultatelor și graficelor în detaliu. Mai jos este lista parametrilor a căror variație are efect asupra acurateții:

- Numărul de date de intrare
- Funcția de normare a datelor
- Dimensiunea intrării
- Numărul de epoci
- Numărul de straturi
- Numărul de neuroni de pe fiecare strat
- Funcțiile de activare
- Funcția de optimizare
- Metoda de verificare a acurateții

Capitolul 4

Rezultate

În această secțiune este prezentat mai întai câte un exemplu pentru SARIMA și pentru MLP, explicat în detaliu, pentru înțelegerea punerii celor două metode în practică, iar apoi sunt prezentate 3 exemple pe diverse seturi de date pentru a compara cât mai bine rezultatele.

4.1 Exemplu complet SARIMA

Exemplul a fost în totalitate generat utilizând limbajul Python și mediul de dezvoltare Jupyter-Notebook. Având în vedere că până acum au fost prezentate câteva exemple simple pentru fiecare noțiune, acum se va prezenta o serie de timp reală, complexă, care să conțină toate noțiunile discutate până acum: autoregresie, medie alunecătoare, integrare și sezonalitate.

4.1.1 Achiziționarea datelor

Seria de date folosită[19] reprezintă consumul total de energie al sectorului rezidențial din Statele Unite. Datele sunt măsurate în $10^{12}Btu$, unde $1Btu = 0.000293KWh$. Datele sunt eșanstionate lunar din Ianuarie 1973 până în Februarie 2020. Acest set de date a fost ales deoarece el are potențial de a conține un trend pozitiv, întru-cât prezintă consumul energetic pe o perioadă de aproape 50 de ani, iar cu explozia tehnologică din ultimii zeci de ani, un trend pozitiv este foarte probabil. De asemenea, fiind eșantionat lunar, sunt șanse mari de a observa o sezonalitate în fiecare an a consumului de energie.

4.1.2 Prelucrarea datelor

Un pas foarte important care trebuie respectat înainte de orice predicție este prelucrarea datelor. Structura seriilor de date trebuie standardizată pentru a putea aplica o predicție pe diverse seturi de date, deoarece acestea diferă de la domeniu la domeniu, de la website la website. Câteva dintre problemele des întâlnite la bazele de date cu serii de timp sunt:

- Lispa unor valori din serie, lucru ce poate strica sezonalitatea datelor,
- Duplicarea datelor, cu același efect negativ ca problema de mai sus,
- Formatul fișierului în care sunt salvate datele,
- Modalitatea de separare a datelor,
- Prezența de caractere non-alfanumerice,
- Neordonarea datelor.

În general, câteva tehnici de combatere a acestor inconsitențe sunt:

- Modificarea separatorilor din caracterul găsit în baza de date în virgulă,
- Exportarea bazei de date într-un tip de fișier propice analizei(ex: .CSV),
- Eliminarea liniilor cu index duplicat,
- Modificarea caracterelor non-alfanumerice într-o valoare stabilită(ex: Nan, 0),
- Reindexarea tuturor datelor pentru a scăpa de problema lipsei unei date,
- Ordonarea datelor cronologic,
- Interpolarea datelor lipsă pentru o mai bună vizualizare și predicție,
- Eșantionarea datelor la o frecvență mai mică dacă datele sunt mult prea des eșantionate(ex: la fiecare oră sau minut).

După ce datele au fost prelucrate, ele sunt afișate pe un grafic pentru o inspecție rapidă a valorilor, a consistenței și a formei seriei de timp. Pentru exemplul actual, graficul este reprezentat în Fig. 4.1 :

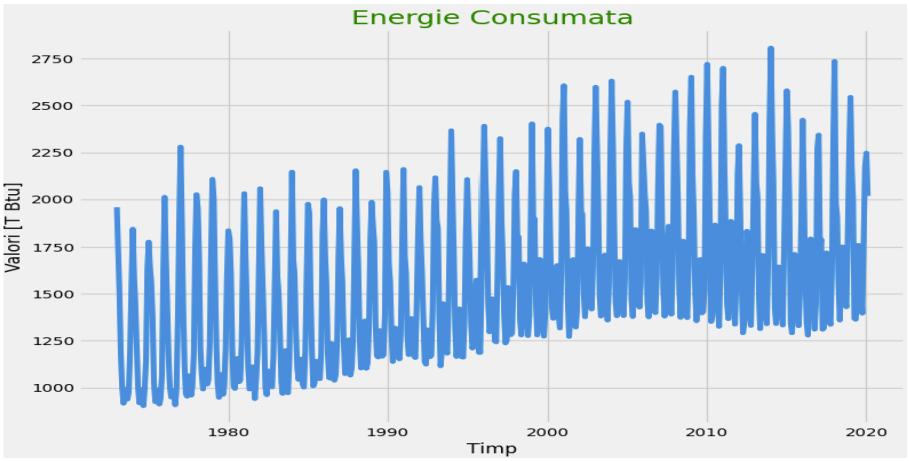

Figura 4.1: Energie consumată.

La prima vedere seria de timp pare să conțină atât trend pozitiv, cât și sezonalitate.

4.1.2.1 Interpretarea datelor

O funcție esențială în Python pentru a interpreta mai bine seria de timp este funcția *seasonal_decompose*[1]. Această funcție primește drept parametrii seria de timp și perioada sezonalității. În cazul de față, având date ce se repetă din an în an cu date eșantionate din lună în lună, perioada este 12. Funcția aplică o diferențiere asupra seriei, dar în loc să scadă din y_t pe y_{t-1}, o să îl scadă pe y_{t-k}, unde k=perioada. După ce scapă de sezonalitate, funcția scade din seria originală seria staționarizată, rămânând astfel doar partea de sezonalitate. Mai jos sunt afișate în ordine seria orginilă, trendul, sezonalitatea și rezidurile:

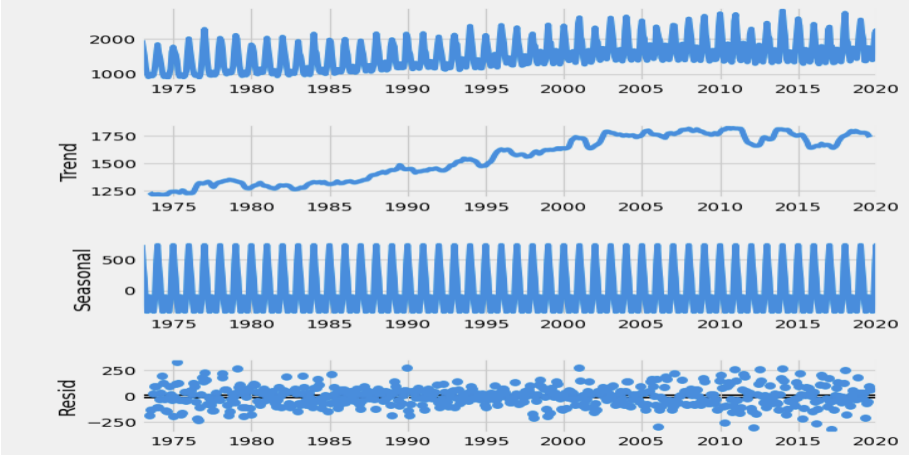

Figura 4.2: Seria originală a) Total b)Trend c) Sezonalitate d) Reziduri.

[1] `https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html`

Din Fig. 4.2 se poate observa că perioada 12 a fost bine aleasă deoarece sezonalitatea este uniformă, iar trendul nu prezintă oscilații care să semene cu sezonalitatea. De asemenea, se mai poate observa și că trendul are o pantă pozitivă, deci datele nu prezintă staționarizare. Pentru a verifica acest lucru se va rula testul Dickey-Fuller. Rezidurile de asemenea sugerează că perioada sezonalității a fost bine aleasă, deoarece ele sunt centrate pe 0, iar dispersia lor este uniformă de-a lungul axei OX.Urmează Testul Dickey-Fuller: În Python acest test se realizează cu funcția *adfuller*[2] care primește ca parametru seria de timp și returnează un coeficient care reprezintă certitudinea ca seria să fie staționarizată. În general, pentru a cataloga o serie drept staționarizată, acel coeficient trebuie să fie mai mic de 5%, adică 0.05. Rezultatul se află în Fig. 4.3:

```
result = adfuller(df.value)
print('p-value: %f' % result[1])
```

```
p-value: 0.548434
```

Figura 4.3: Dickey-Fuller Test pe seria originală.

După cum s-a intuit încă din Fig. 4.2, seria de timp trebuie diferențiată, dar ordinul de integrare al sezonalității este 0, sezonalitatea fiind deja staționară. După diferențiere, decompoziția sezonală arată astfel:

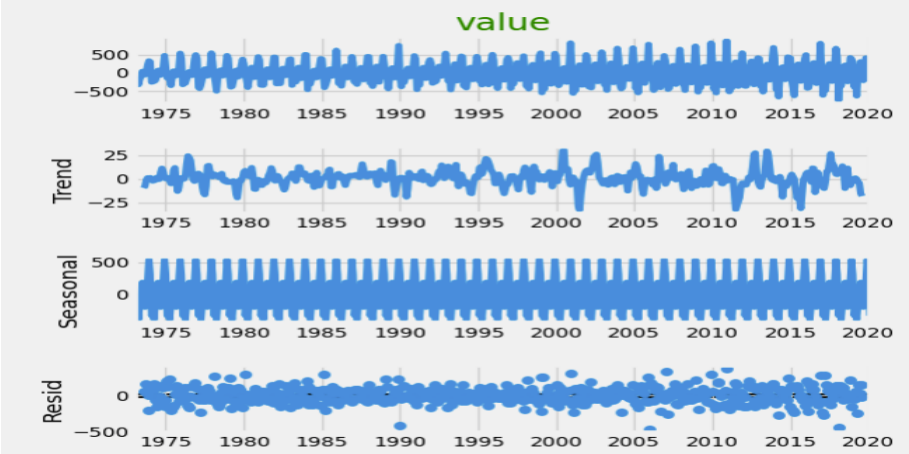

Figura 4.4: Seria diferențiată a) Total b) Trend c) Sezonalitate d) Reziduri.

Testul Dickey-Fuller va fi:

```
result = adfuller(df.diff().dropna())
print('p-value: %f' % result[1])
```

```
p-value: 0.000000
```

Figura 4.5: Dickey-Fuller Test pe seria diferențiată.

[2] `https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html`

În momentul de față se cunoaște valoarea termenului I din modelul SARIMA, acesta fiind egal cu 1, deoarece este nevoie de o diferențiere pentru a staționariza datele. Fiind un model complex, provenit din realitate, interpretarea autocorelației și autocorelației parțiale nu va mai fi la fel de ușoară ca în exemplele date mai sus. De aceea în practică, mai întâi se observă autocorelația și autocorelația parțială, se trag niște concluzii preliminare pe baza lor, iar apoi se aplică o căutare cartezianǎ (*grid search*) în jurul valorilor ordinelor găsite și se extrage modelul cu valoarea AIC cât mai mică. În Fig. 4.6 este prezentată autocorelația parțială a seriei difențiate:

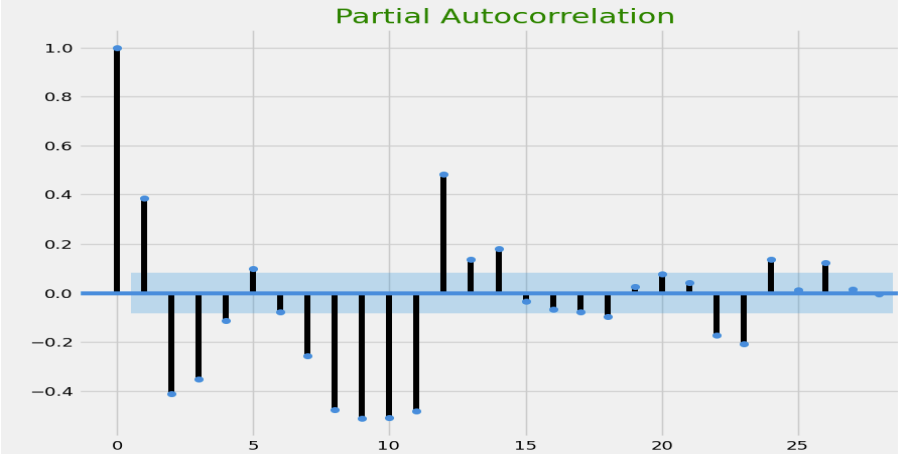

Figura 4.6: Autocorelația parțială a seriei difențiate.

La prima vedere, Fig. 4.6 pare să sugereze o componentă AR de ordin mare, dar privită mai atent, figura sugerează o descreștere exponențială a valorilor, putând fi vorba defapt despre o componentă ARMA. Neavând nicio valoare mult mai mare decât celălalte, excluzând-o pe prima care este mereu 1, se poate trage concluzia că este vorba despre o componentă MA, fără parte de AR. Din punct de vedere al părții AR al sezonalității, lucrurile devin și mai greu de observat, dar o valoare intuitivă ar fi 1 sau 2 deoarece din 12 în 12 eșantioane sunt prezente 1 sau 2 valori mari. În Fig. 4.7 este prezentată autocorelația parțială a seriei diferențiate:

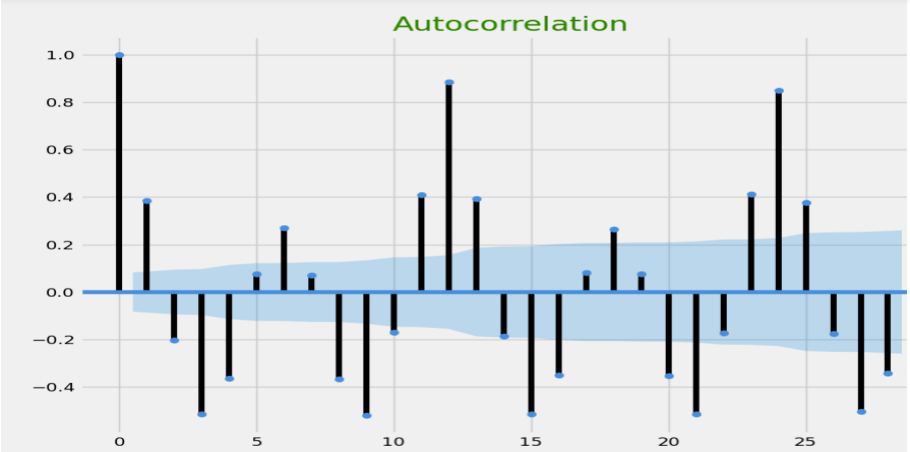

Figura 4.7: Autocorelația a seriei diferențiate.

Primul lucru care se observă din acest grafic este prezența a două șabloane care se repetă. Unul dintre ele este format din valorile pozitive care se repetă din 12 în 12, sugerând sezonalitatea anuală, iar celălalt este format din valorile negative care se repetă din 6 în 6, sugerând sezonalitatea semestrială. Valorile pozitive sunt mai mari decât cele negative în modul, așadar sezonalitatea anuală este mai pronunțată decât cea semestrială. Alt lucru demn de notat este faptul că eșantioanele scad continuu până la eșantionul 2, după aceea ele crescând ca și valoare în modul. Acest lucru poate să sugereze că este vorba de un model MA(2) sau MA(1). Din punct de vedere al părții MA a sezonalității, la fel ca la partea AR, este greu de intuit un ordin, dar o valoare intuitivă ar fi 3 deoarece apar mereu câte 3 valori mari consecutive.

4.1.3 Construcția modelului

Primul lucru care trebuie realizat este împărțirea datelor în date de antrenament și date de test. În mod normal datele de test trebuie să reprezinte între 10 și 20 la sută din lungimea datelor de antrenament. Pentru acest exemplu se va alege pentru datele de test un procent de 15% din datele totale, adică aproximativ 17% din datele de antrenament.

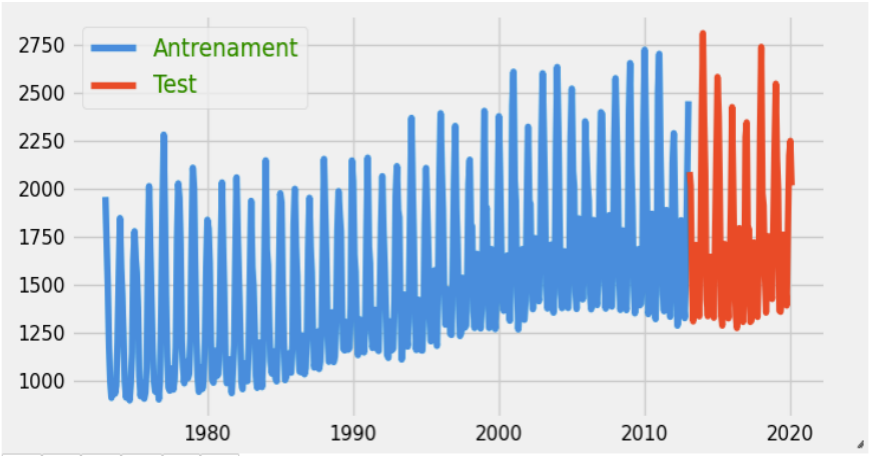

Figura 4.8: Date Antrenament Vs. Date Test.

Urmează construcția modelului propriu-zis. Pentru acest model se va apela funcția *SARIMAX*[3] din biblioteca *statespace*. Această funcție necesită următorii parametrii:

- Ordinul AR pentru trend: S-a găsit mai sus ca fiind 0, dar se va încerca și cu 1 pentru mai multe rezultate
- Ordinul MA pentru trend: S-a găsit ca fiind 1 sau 2. Se va încerca și cu 3 din același motiv
- Ordinul I pentru trend: S-a găsit ca fiind 1. Nu se vor considera alte valori deoarece testul Dickey-Fuller are o precizie mult mai mare decât interpretarea datelor.
- Perioada sezonalității: S-a găsit ca fiind 12. Nu se vor încerca alte valori fiindcă este clar că datele sunt periodice anual.
- Ordinul AR pentru sezonalitate: S-a găsit mai sus ca fiind 1 sau 2, dar se va încerca și cu 3 pentru mai multe rezultate
- Ordinul MA pentru sezonalitate: S-a găsit ca fiind 1,2 sau 3. Nu se va încerca și alte valori fiindcă s-a observat clar că nu sunt mai mult de 3 valori consecutive de valoare mare.
- Ordinul I pentru sezonalitate: S-a găsit ca fiind 0. Nu se vor considera alte valori deoarece testul Dickey-Fuller are o precizie mult mai mare decât interpretarea datelor.

[3] `https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html`

Așadar se vor încerca modele cu toate combinațiile de ordine posibile din lista de mai sus și se va extrage modelul cu indicele AIC cel mai mic. În Fig. 4.9 sunt afișate valorile AIC pentru fiecare model, precum și ordinele fiecărei componente a modelului. Se poate observa că modelele ARIMA(0,1,3)x(1,0,3)12 și ARIMA(0,1,3)x(2,0,3)12 au obținut cel mai mic scor AIC, deci aceste două modele sunt cele mai performante pentru seria de timp stabilită. Modelul ales este aproape în totalitate conform cu valorile care au fost găsite pe baza graficelor, singura diferență fiind la ordinul MA al trendului. De aceea se recomandă încercarea mai multor modele. Următorul pas care trebuie efectuat, este antrenarea modelului pe datele de antrenament. Acest lucru, cum nu ține de construcția modelului în sine, ci este un pas generic pentru orice tip de model, se va face cu ajutorul funcției *fit* din Python. După ce modelul este antrenat, se poate vizualiza efectul fiecărui parametru din model cu funcția *summary*. Din datele afișate, coloana cu numele $P > |z|$ arată importanța fiecărui parametru. Interpretarea rezultatului este asemănătoare cu cea a testului Dickey-Fuller din funcția *adfuller*, adică dacă valorile sunt mai mari de 5% atunci parametrul are o influență mult prea mică asupra modelului și poate fi eliminat. Mai jos este prezentat tabelul cu valori:

```
ARIMA(0,1,1)x(1,0,1,12) - AIC:5547.7835241904995
ARIMA(0,1,1)x(1,0,2,12) - AIC:5414.58654627531
ARIMA(0,1,1)x(1,0,3,12) - AIC:5265.455895086761
ARIMA(0,1,1)x(2,0,1,12) - AIC:5436.3155356486905
ARIMA(0,1,1)x(2,0,2,12) - AIC:5409.993937854644
ARIMA(0,1,1)x(2,0,3,12) - AIC:5314.865739437815
ARIMA(0,1,1)x(3,0,1,12) - AIC:5291.349779080505
ARIMA(0,1,1)x(3,0,2,12) - AIC:5295.758717472442
ARIMA(0,1,1)x(3,0,3,12) - AIC:5566.886801453587
ARIMA(0,1,2)x(1,0,1,12) - AIC:5456.212674667613
ARIMA(0,1,2)x(1,0,2,12) - AIC:5324.91180700063
ARIMA(0,1,2)x(1,0,3,12) - AIC:5182.6458749155145
ARIMA(0,1,2)x(2,0,1,12) - AIC:5357.98917915246
ARIMA(0,1,2)x(2,0,2,12) - AIC:5320.100427410305
ARIMA(0,1,2)x(2,0,3,12) - AIC:5183.73255885607
ARIMA(0,1,2)x(3,0,1,12) - AIC:5219.085323854817
ARIMA(0,1,2)x(3,0,2,12) - AIC:5218.264516973931
ARIMA(0,1,2)x(3,0,3,12) - AIC:5191.786018700615
ARIMA(0,1,3)x(1,0,1,12) - AIC:5443.411584939157
ARIMA(0,1,3)x(1,0,2,12) - AIC:5310.850856720703
ARIMA(0,1,3)x(1,0,3,12) - AIC:5167.538647935623
ARIMA(0,1,3)x(2,0,1,12) - AIC:5356.93018025066
ARIMA(0,1,3)x(2,0,2,12) - AIC:5305.76408497696
ARIMA(0,1,3)x(2,0,3,12) - AIC:5169.686067564272
ARIMA(0,1,3)x(3,0,1,12) - AIC:5216.501985274882
ARIMA(0,1,3)x(3,0,2,12) - AIC:5216.873433308447
ARIMA(0,1,3)x(3,0,3,12) - AIC:5182.218352106522
ARIMA(1,1,1)x(1,0,1,12) - AIC:5469.297199002344
ARIMA(1,1,1)x(1,0,2,12) - AIC:5353.925184968226
ARIMA(1,1,1)x(1,0,3,12) - AIC:5194.928518882272
ARIMA(1,1,1)x(2,0,1,12) - AIC:5348.859569959758
ARIMA(1,1,1)x(2,0,2,12) - AIC:5451.917349321511
ARIMA(1,1,1)x(2,0,3,12) - AIC:5206.477902399747
ARIMA(1,1,1)x(3,0,1,12) - AIC:5204.318798021082
ARIMA(1,1,1)x(3,0,2,12) - AIC:5224.084661551001
ARIMA(1,1,1)x(3,0,3,12) - AIC:5307.769139431966
ARIMA(1,1,2)x(1,0,1,12) - AIC:5457.160262760866
ARIMA(1,1,2)x(1,0,2,12) - AIC:5429.735998213202
ARIMA(1,1,2)x(1,0,3,12) - AIC:5264.0921133903275
ARIMA(1,1,2)x(2,0,1,12) - AIC:5385.043966082307
ARIMA(1,1,2)x(2,0,2,12) - AIC:5430.273054783935
ARIMA(1,1,2)x(2,0,3,12) - AIC:5278.367143598162
ARIMA(1,1,2)x(3,0,1,12) - AIC:5223.674312505034
ARIMA(1,1,2)x(3,0,2,12) - AIC:5401.865881382381
ARIMA(1,1,2)x(3,0,3,12) - AIC:5245.76918519509
ARIMA(1,1,3)x(1,0,1,12) - AIC:5448.357287933797
ARIMA(1,1,3)x(1,0,2,12) - AIC:5318.761127894726
ARIMA(1,1,3)x(1,0,3,12) - AIC:5177.265068171844
ARIMA(1,1,3)x(2,0,1,12) - AIC:5635.862999072287
ARIMA(1,1,3)x(2,0,2,12) - AIC:5369.384369659851
ARIMA(1,1,3)x(2,0,3,12) - AIC:5420.474296179425
ARIMA(1,1,3)x(3,0,1,12) - AIC:5653.066522868159
ARIMA(1,1,3)x(3,0,2,12) - AIC:5512.985658741599
ARIMA(1,1,3)x(3,0,3,12) - AIC:5227.578936001329
```

Figura 4.9: AIC pentru diverse modele.

```
results = mod.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.4553      0.039    -11.589      0.000      -0.532      -0.378
ma.L2         -0.3718      0.053     -7.071      0.000      -0.475      -0.269
ma.L3         -0.0954      0.056     -1.708      0.088      -0.205       0.014
ar.S.L12       0.9983      0.003    314.174      0.000       0.992       1.005
ma.S.L12      -0.7820      0.045    -17.223      0.000      -0.871      -0.693
ma.S.L24      -0.1168      0.059     -1.964      0.050      -0.233      -0.000
ma.S.L36       0.1414      0.048      2.950      0.003       0.047       0.235
sigma2      6891.0516    313.812     21.959      0.000    6275.991    7506.113
==============================================================================
```

Figura 4.10: Valori parametrii model ales.

Din Fig. 4.10 se observă că niciun parametru nu are pe coloana $P > |z|$ valoarea mai mare de 5%, deci modelul a fost ales bine.
Există două tipuri de testare a acurateții modelului, cu toate că numele lor seamănă foarte mult și are același sens, în literatură ele sunt denumite astfel[4]:

1. Predicție (*Prediction(in-sample)*)
   Acest tip de aflare al datelor presupune predicția datelor de antrenament. Datele de test nu sunt folosite absolut deloc în acest tip de prezicere. La rândul său, acest tip de testare a acurateții este împărțit în două sub-metode, una mai eficientă și alta mai puțin eficientă:

   - Metoda statică: Se folosesc toate datele de antrenament și se prezic diverse valori din acele date de antrenament. De obicei din această metodă rezultă o acuratețe foarte bună, dar ea nu este neapărat reală.
   - Metoda dinamică($Cross-Validation$): Se folosesc date de antrenament până în momentul prezicerii, iar apoi se face prezicerea. Odată prezisă o valoare sau un set de valori, valorile reale sunt adăugate la datele de antrenament, se reantrenează modelul ținând cont și de ele și se prezice următoarea valoare sau următorul set de valori. Din această metodă rezultă o acuratețe mai mică decât la cealaltă, ea fiind mai dură, dar acuratețea dată de ea este mai apropiată de cea reală.

2. Prognoză (*Forecasting(out-of-sample)*)
   Prognoza presupune aflarea valorilor din setul de test, nu cel de antrenament. Ca și predicția, și ea este împărțită în două sub-metode:

   - Prognoză cu pas simplu(*One-Step Forecast*): presupune aflarea unei singure valori din datele de test.

[4]Ramazan Gencay, Xian Yang, "A forecast comparison of residential housing prices by parametric versus semiparametric conditional mean estimators", Economics Letters 52,1 Mai 1996

- Prognoza cu pas multiplu(*Multi-Step Forecast*): Se prezic mai multe valori din setul de date , iar modelul se reantrenează ținând cont de valorile reale (updated) sau prezise anterior (approximated), nu de cele reale. Această sub-metodă de predicție este cea mai dură dintre cele 4 descrise, acuratețea ei fiind de multe ori cea mai mică.

Pentru a măsura acuratețea predicțiilor, există mai multe metode de a testa erorile dintre datele reale și cele prezise. Metoda folosită în continuare poartă numele de *Eroarea medie absolută în procente(eng. MAPE).*

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{|y_t - \hat{y}_t|}{|y_t|} \tag{4.1}$$

Aceasta, spre deosebire de o mare parte dintre celelalte metode are avantajul unui interval al valorilor luate foarte precis. Mai precis această metodă returnează procentul erorii medii, așa că valorile sale vor fi mereu în intervalul [0,1]. În Fig. 4.11 sunt prezentate datele de antrenament și datele prezise cu metoda de Predicție Statică:

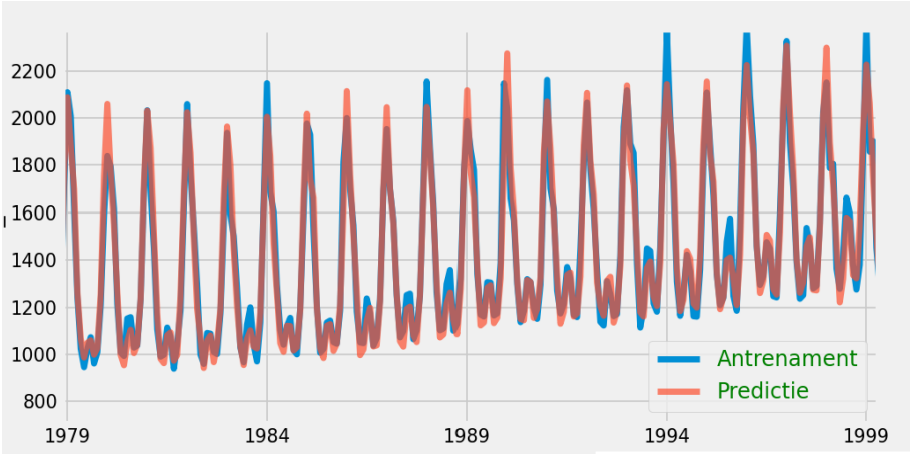

Figura 4.11: Predicție Statică.

Valoarea MAPE obținută pentru acest grafic a fost 0.0445, așadar s-a obținut o acuratețe de 95.55%. În Fig. 4.12 sunt prezentate datele de antrenament și datele prezise cu metoda de Prognoză cu pas multiplu:

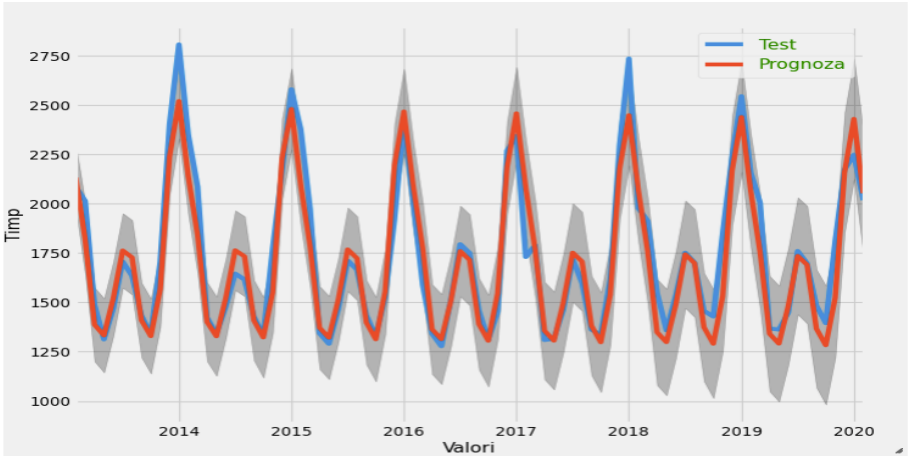

Figura 4.12: Prognoză multiplă.

Valoarea MAPE obținută pentru acest grafic a fost 0.0507, așadar s-a obținut o acuratețe de 94.93%. Se observă că acuratețea celei de-a doua metodă este mai mică, acest lucru fiind de așteptat. Cu toate acestea, conform articolului[5], aceste valori ale acurateții denotă o predicție foarte bună.

4.2 Exemplu complet MLP

În continuare se va prezenta un exemplu practic pentru a avea o mai bună înțelegere a modului de utilizare a rețelelor neurale. Acest exemplu are rolul de a demenostra cum se pot pune în practică toate informațiile de mai sus legate de învățare automată.

4.2.1 Achiziționarea datelor

Pentru a face o comparație cât mai corectă se va folosi același set de date ca și la metodele statistice[19] și se vor aplica aceleași prelucrări asupra setului de date pentru a avea niște date consistente.

4.2.2 Prelucrarea datelor

Datele originale au fost normalizate la medie 0 și variantă 1, apoi au fost construite structurile pentru intrarea și ieșirea din rețea. În Fig. 4.13 se poate observa structura intrărilor de antrenament: fiecare rând reprezintă rândul precedent deplasat la stânga cu o valoare, iar la final are următoarea valoare din seria de timp:

[5] https://www.researchgate.net/figure/nterpretation-of-typical-MAPE-values_tbl1_257812432

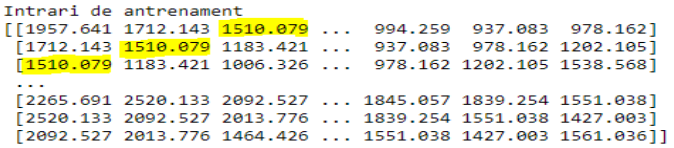

Figura 4.13: Intrări rețea nenormate.

4.2.3 Construirea rețelei

Au fost încercate mai multe valori ale parametrilor, iar cele mai bune au fost următoarele:

- Lungime intrare: 24
  S-a ales un număr divizibil cu perioada sezonalității semnalului. Lungimea de 2 ani este suficient de mare pentru a putea observa atât sezonalitatea, cât și trendul, dar suficient de mică pentru a preveni *overfitting*-ul.

- Numărul de straturi ascunse: 2 Dense
  Deoarece seria de timp este relativ mică pentru o rețea neurală, a fost ales un număr mic de straturi.

- Numărul de epoci: 60
  S-a observat că după câteva zeci de epoci, funcția de cost nu mai scade, deci modelul este antrenat. Un număr mai mare de epoci ar fi crescut riscul de *overfitting*.

- Numărul de neuroni pe fiecare strat: 24, respectiv 12
  Au fost alese aceste numere pentru ca fiecare neuron să reprezinte ponderea fiecărei luni din an, întrucât sezonalitatea este anuală.

- Funcția de activare: ReLU
  A fost alesă funcția ReLU deoarece se dorește modelarea unei regresii, nu a unei clasificări. De asemenea ReLU în practică oferă printre cele mai bune rezultate.

- Funcția de optimizare pentru minimizarea gradientului: Adam
  Motivul pentru care s-a folosit Adam îl reprezintă rezultatele sale foarte bune pe cazuri generale.

- Funcție de cost: Mean Squared Error(MSE)
  Având valorile normate, nu are sens să se considere o funcție de cost care să țină cont de valori prea mici sau prea mari. Ridicarea la pătrat asigură o penalizare mai mare a valorilor depărtate cu mult de cele reale.

- Funcție de verificare a acurateții: MAPE
  Întrucât valorile sunt readuse la cele dinainte de normare atunci când se aplică funcția MAPE, nu există riscul ca ele să fie foarte apropiate de 0 astfel încât această funcție să devină inutilă.

Mai jos este reprezentat codul Python folosit, în care se regăsesc toate aceste informații listate mai sus:

```
model = Sequential()
model.add(Dense(24, activation='relu', input_dim=24))
model.add(Dense(12, activation='relu', input_dim=24))
model.add(Dense(1))
model.compile(optimizer='Adam', loss='mean_squared_error', metrics=['accuracy'])
history=model.fit(x_train, y_train, epochs=60, verbose=0, validation_split=0.2,callbacks=[PrintLogs(epochs)])
```

Figura 4.14: Cod rețea neurală Python.

Pentru a identifica numărul optim de epoci, s-a urmărit graficul valorii funcției de cost la fiecare epocă:

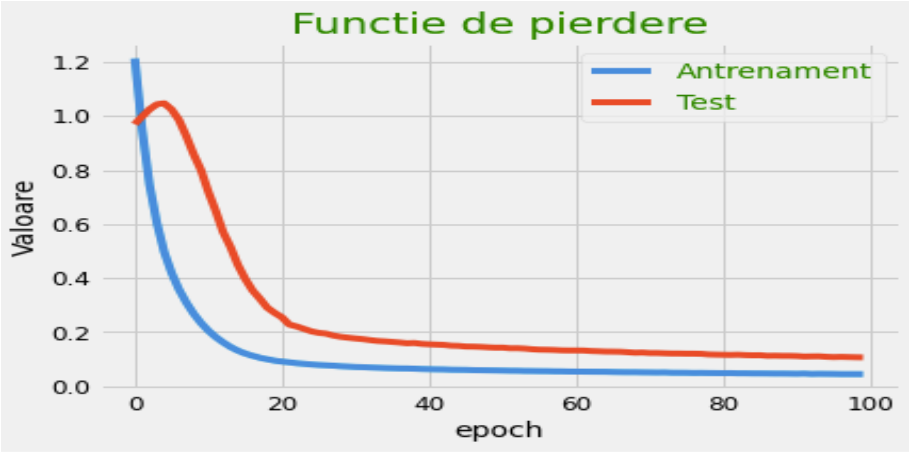

Figura 4.15: Valori funcție de cost.

Cu configurațiile de mai sus, s-au obținut următoarele valori MAPE pentru datele de test:

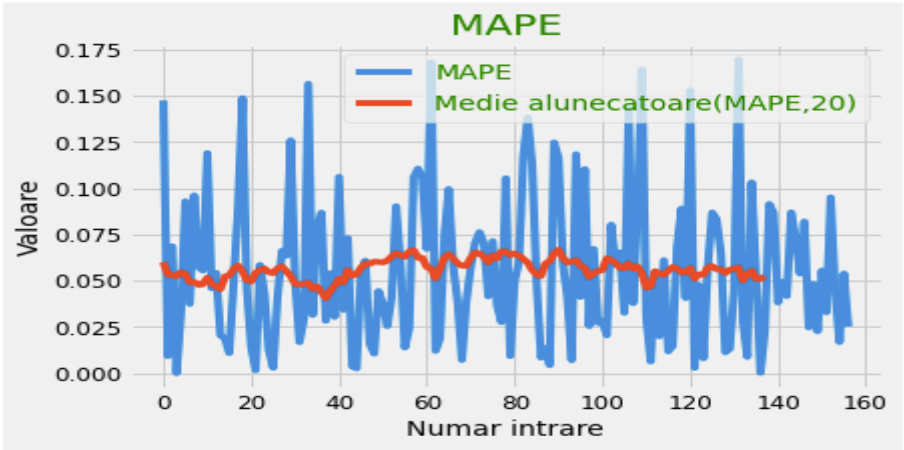

Figura 4.16: Valori MAPE pentru date de test.

Se poate observa din graficul de mai sus că valorile MAPE variază între 0.000 și 0.175, ceea ce denotă la prima vedere un rezultat relativ bun. Ce este și mai important este că valoarea mediei alunecătoare cu pas 20 a MAPE este în jurul valorii 0.06. Acest lucru înseamnă că această metodă produce în medie valori cu o acuratețe de 94%. Cel din urmă rezultat este mult mai bun decât ce părea la prima vedere că denotă graficul și de asemenea este și o interpretare mai corectă din punct de vedere statistic. Datele prezise pot fi vizualizate în Fig. 4.17 alături de datele reale de test:

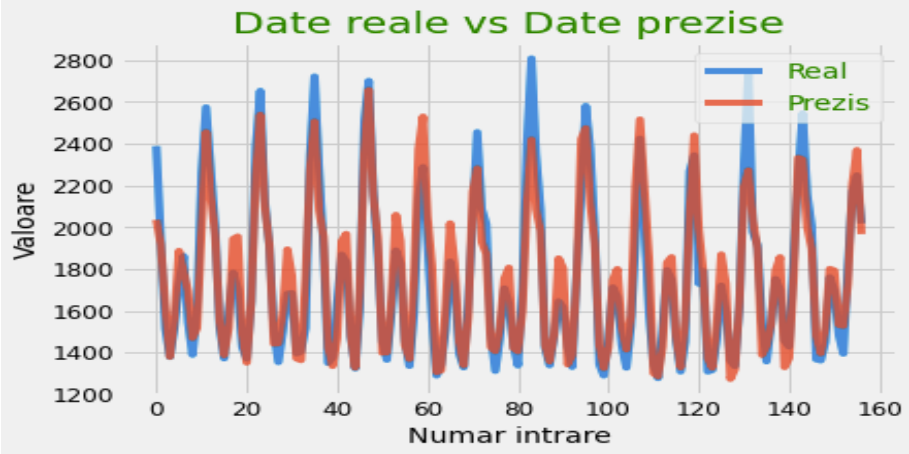

Figura 4.17: Valori de test reale vs. prezise.

Fig. 4.17 reflectă rezultatul MAPE de mai sus. Într-adevăr predicția urmează cu o precizie foarte mare forma datelor reale.

4.3 Exemple SARIMA & MLP

În secțiunea ce urmează sunt prezentate în paralel rezultatele celor două metode de predicție analizate, SARIMA și MLP, pe diverse seturi de date. Vor fi analizate următoarele caracteristici ale celor două metode:

- Parametrii aleși
- Timp de rulare
- Acuratețe predicție

Pentru a avea o perspectivă cât mai corectă asupra acestor caracteristici, se vor folosi seturi de date cât mai diversificate astfel încât să cuprindă o varietate cât mai mare dintre următoarele aspecte legate de seriile de timp:

- Lungimea seriei: Se vor căuta serii de timp ale căror lungimi să se afle într-un interval cât mai larg.

- Se vor căuta serii de timp cu / fără trend.
- Se vor căuta serii de timp cu / fără sezonalitate.

Toate seriile de timp ce vor fi prezentate sunt serii de timp reale. Nu s-au folosit serii de timp deterministe, stocastice, haotice sau sintetice, deoarece din punct de vedere al dificultății ele sunt mai ușor de prezis și sunt mult mai rar întâlnite în practică. Toate aceste informații vor fi specificate la fiecare serie de timp în parte. Rezultatele vor fi afișate fie sub formă de grafice, fie sub formă de tabele. Exemplele următoare sunt sortate după dificulatatea lor. Toate rezultate de predicție sunt făcute pe date de test care nu fac parte din cele de antrenament (*Out-of-Sample Forecasting*)

4.3.1 Serie de timp 1: Vânzări medicamente

Seria de date de mai jos reprezintă vânzările de medicamente anti-diabet exprimate în mii de dolari pe perioada de timp 1993 - 2005[6].
Dificultate serie: scăzută.
Număr total date: 204.
Perioadă eșantionare: lunară.
Trend: pozitiv.
Perioadă sezonalitate: anuală.
Grafic serie de timp împărțit în date de antrenament și de test:

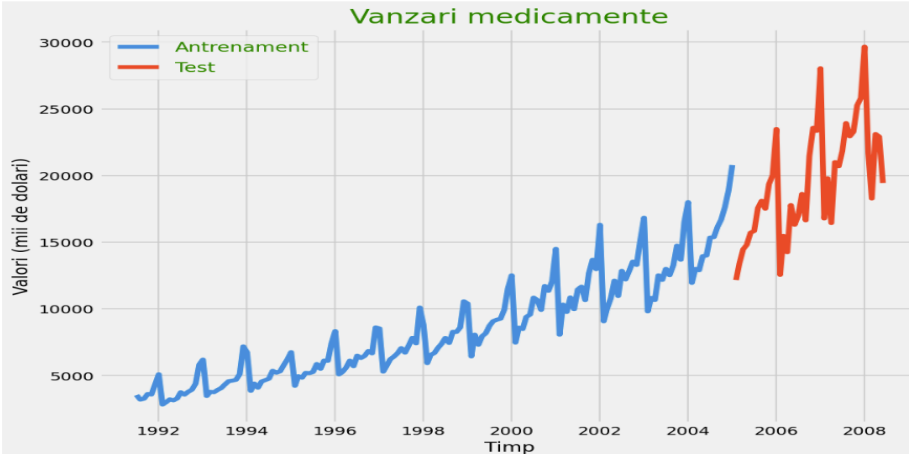

Figura 4.18: Serie de timp: Medicamente.

4.3.1.1 SARIMA

Se va testa staționatizarea seriei de timp: Dickey-Fuller(serie originală): 1.00

[6]Seria este disponibilă la `https://raw.githubusercontent.com/selva86/datasets/master/a10.csv`

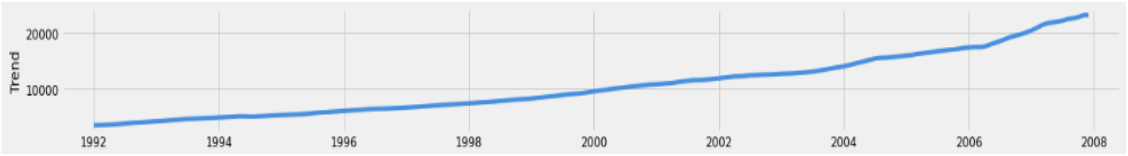

Figura 4.19: Trend serie originală.

Dickey-Fuller (serie diferențiată odată): 0.11

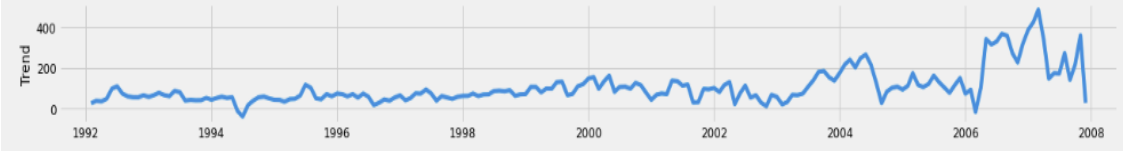

Figura 4.20: Trend serie diferențiată odată.

⇒ Seria va fi diferențiată o singură dată, cu toate că teoria spune că ar trebui diferențiată de două ori în acest caz, valoarea 0.11 fiind doar cu puțin mai mare decât pragul de 0.05, este posibil ca încă o diferențiere să producă o pierdere de date prea mare. În urma interpretării graficelor ACF și PACF și a construirii mai multor modele, s-a identificat ca fiind cel mai bun model pentru această serie de timp modelul: ARIMA(2,1,2)x(2,0,1,12) cu o valoare AIC de 1748. Valorile prezise sunt afișate mai jos:

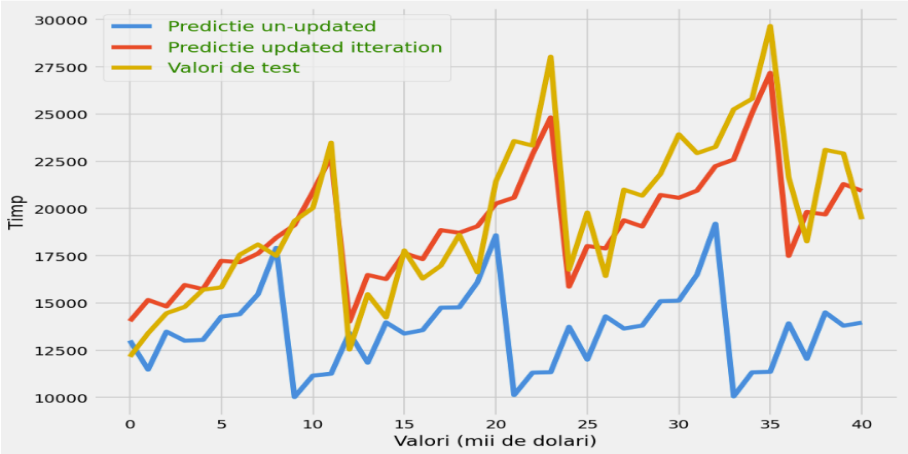

Figura 4.21: Valori antrenament, test și predicție.

Valoarea MAPE a predicției multi-step (approximated itteration) este: 0.191.
Valoarea MAPE a predicției multi-step (updated itteration) este: 0.076.
Se va considera doar timpul de rulare necesar construirii unui singur model, deoarece timpul găsirii celui mai bun model depinde de cât de mult se variază parametrii modelului.
Timp de rulare formare model: 0.821s

4.3.1.2 MLP

După ce datele au fost normate la medie 0 și variantă 1, se încearcă diverse configurări ale rețelei pentru a vedea rezultatele cele mai bune. Se va folosi funcția de optimizare Adam. Se verifică funcția de pierdere pentru a considera un număr de epoci ideal.

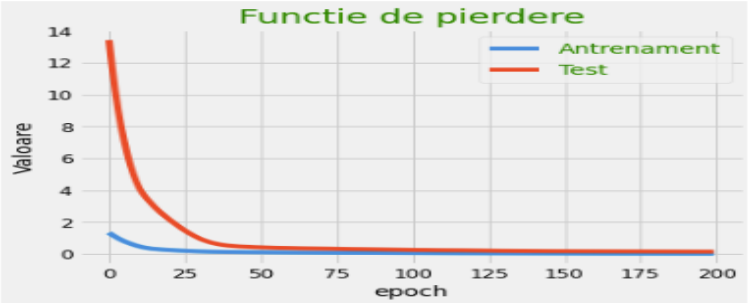

Figura 4.22: Valoare funcție de pierdere.

Se vor alege doar 100 de epoci de antrenare, întrucât la valoarea 100, pierderea este deja stabilizată. Verificarea acurateții se va face de asemenea cu funcția MAPE. Se vor încerca valori pentru numărul de straturi, numărul de neuroni de pe fiecare strat și dimensiunea intrărilor și se va considera modelul cu valoare MAPE cea mai mică. În funcție de cât de mult se variază toți parametri se va obține un timp de rulare mai mic sau mai mare. Mai jos este prezentat timpul de rulare pentru o singură configurație a rețelei. Cel mai bun rezultat a fost găsit pentru următoare configurație:
Număr de straturi: 2
Număr de neuroni pe fiecare strat: 12 / 12
Lungimea intrării: 12
Valoarea MAPE a predicției multi-step (un-updated) este: 0.096.
Valoarea MAPE mediei a predicției multi-step (updated itteration) este: 0.054
Valoarea MAPE a predicției multi-step (un-updated) s-a determinat dându-i rețelei la ieșire dimensiunea orizontului de predicție, în loc de dimensiunea 1. Valoarea MAPE pentru fiecare valoare în timp a datelor de test, precum și media ei alunecătoare cu pasul 20 sunt prezentate mai jos:

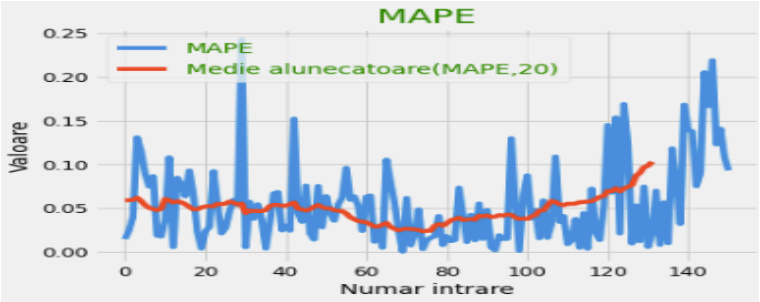

Figura 4.23: Valori MAPE.

Graficul cu valori prezise se află în Fig. 4.24:

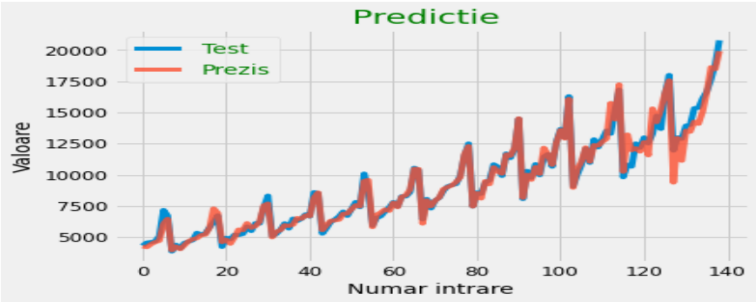

Figura 4.24: Predicție out-of-sample.

4.3.2 Serie de timp 2: Consum de energie

Următoarea serie de date reprezintă consumul de energie electrică din zona de Nord a statului American Illinois din anul 2004 până în anul 2011 exprimat în MW(MegaWatts)[7].
Dificultate serie: medie
Număr total date: 58400
Perioadă eșantionare: orar
Trend: nenul
Perioadă sezonalitate: zilnică / săptămânală / lunară / anuală

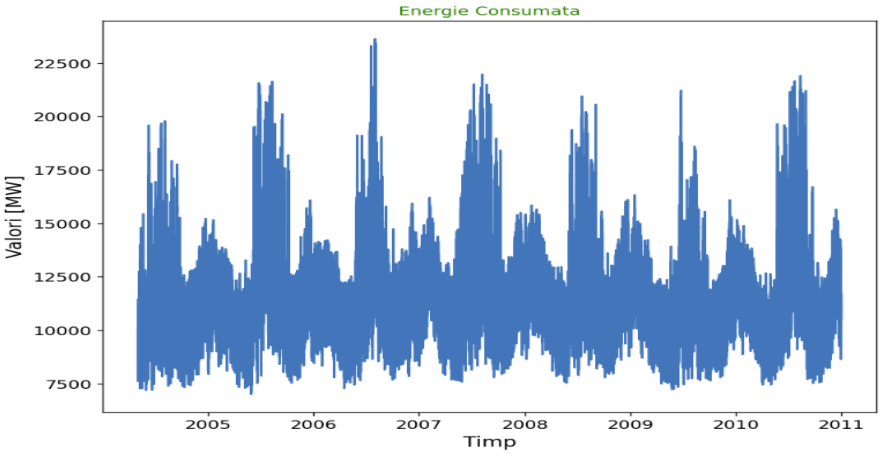

Figura 4.25: Valori serie de timp.

Dificultatea acestei serii de timp provine din două locuri: numărul mare de date pe care îl are și sezonalitatea multiplă. Datele, fiind eșantionate din oră în oră, prezintă mai multe niveluri de sezonalitate. SARIMA este capabilă să împartă semnalul doar în două părți: trend și sezonalitate, unde partea de sezonalitate are o frecvență bine stabilită. Pe acest

[7]Seria este disponibilă la `https://www.kaggle.com/robikscube/hourly-energy-consumption`

set de date, perioada sezonalității este:
24 pentru sezonalitate zilnică,
$24 \cdot 7$ pentru sezonalitate săptămânală, $24 \cdot 7 \cdot 31$ pentru sezonalitate lunară, $24 \cdot 7 \cdot 31 \cdot 12$ pentru sezonalitate anuală. Așadar, conceperea unui model de timp SARIMA performant pentru aceste date este imposibilă. Cu toate aceste, micșorând scopul prezicerii datelor, se pot obține următoarele preziceri:

- Prezicere anuală cu datele re-eșantionate odată pe lună timp de câțiva ani
- Prezicere lunară cu datele re-eșantionate odată pe zi timp de câteva luni
- Prezicere zilnică cu datele eșantionate odată pe oră timp de câteva zile.

Așadar, pentru a avea o acuratețe bună trebuie avut grijă ca datele să aibă un singur nivel de sezonalitate, pentru ca acestea să poată fi modelate de către SARIMA. O altă problemă majoră care apare la sezonalitatea multiplă o reprezintă distanța dintre eșantionale sezonalității celei mai rare. Spre exemplu dacă datele sunt eșantionate orar și se dorește o predicție pe câțiva ani, se va ține cont de sezonalitatea anuală. Cu toate acestea, perioada acestei sezonalități va fi $24 \cdot 7 \cdot 31 \cdot 12$ eșantioane. Acest număr trebuie dat ca parametru construirii modelului, iar pentru acest număr se vor construi toate autocorelațiile și auto-corelațiile parțiale de la ordinul 0, până la ordinul $24 \cdot 7 \cdot 31 \cdot 12$. Acest lucru este imposibil de făcut din cauza limitărilor de calcul și este extrem de neintuitiv ca pentru o predicție de câțiva ani să se ia în calcul valoarea fiecărei ore. Cu toate acestea, ce se poate face în această privintă, este re-eșantionarea datelor la o frecventa suficient de mică astfel incat perioada sezonalitatii principale să fie mai mică, iar calcularea modelului să se poata face într-un timp rezonabil. Exemplul ales reprezintă luarea în calcul a sezonalității anuale, așa că se vor testa diverse nivele ale re-eșantionării datelor pentru a observa cum se modifică acuratețea modelelor în funcție de perioada sezonalității. Mai jos este reprezentată seria de timp eșantionată diferit:

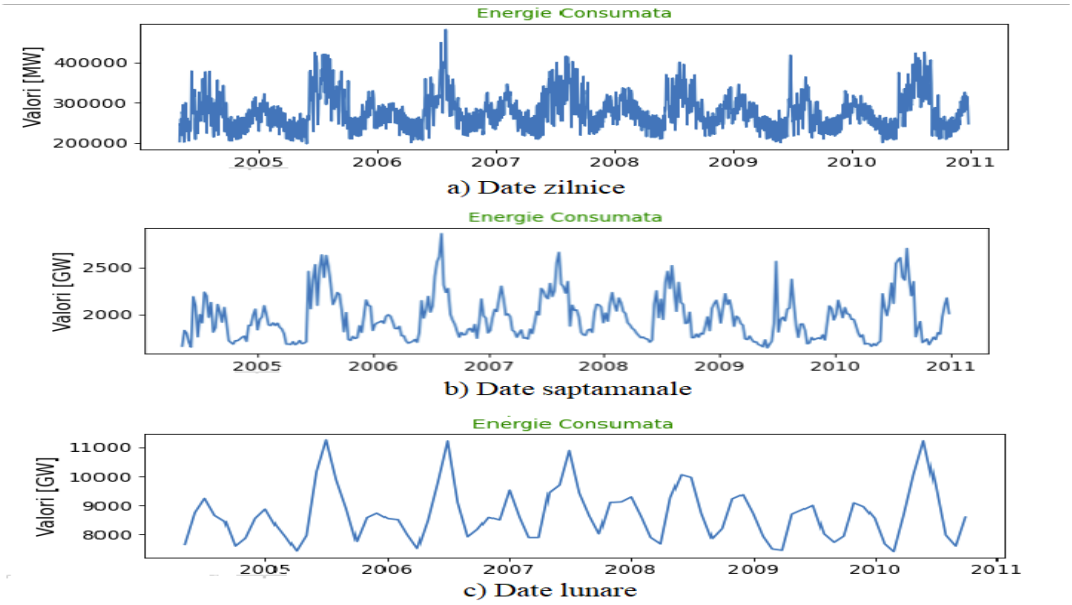

Figura 4.26: Valori serie de timp zilnice/săptămânale/lunare.

Rezultatele pentru cele mai bune modele găsite pentru ultimele două cazuri sunt afișate în tabelele 4.1, 4.2. Primul caz, fiind eșantionat foarte des, este neintuitiv pentru o predicție de câteva luni/câțiva ani:

Model	**Timp de rulare**	**Acuratețe updated**	**Acuratețe approximated(o lună)**
SARIMA	0.844 s	0.062	0.109
MLP	1.42 s	0.065	0.103

Tabela 4.1: Rezultate eșantionare săptămânală.

Parametrii SARIMA sunt (1,0,1)x(1,0,1,52), iar MLP are urmatorii parametrii: epoci=100, lungime intrare=52, număr straturi=2, neuroni pe fiecare strat=52/7.

Model	**Timp de rulare**	**Acuratețe updated**	**Acuratețe approximated(o lună)**
SARIMA	0.865 s	0.083	0.0924
MLP	1.63 s	0.06	0.109

Tabela 4.2: Rezultate eșantionare lunară.

Parametrii SARIMA sunt (2,1,2)x(1,0,2,12), iar MLP are următorii parametrii: epoci=500, lungime intrare=12, număr straturi=2, neuroni pe fiecare strat=12/12. Motivul pentru care timpul de rulare este mai mare de data aceasta la MLP, cu toate că sunt mai puține date, este acela că de data aceasta au fost necesare 500 de epoci pentru ca funcția de cost să se stabilizeze.

4.3.3 Serie de timp 3: Valori bursă

Următorul set de date reprezintă valorile unei acțiuni Apple de-a lungul timpului, din anul 2015 până în anul 2020[8].
Dificultate serie: ridicată.
Număr total date: 1824.
Perioadă eșantionare: zilnică.
Trend: pozitiv.
Perioadă sezonalitate: necunoscută.
Dificultatea acestei serii de timp provine din 2 motive: faptul că se cunosc foarte puține date despre sezonalitate (dacă există și ce perioadă are) și din volatilitatea datelor. Fiind date ce țin de bursă, ele depind de foarte mulți factori externi (reclame, angajări, demisii, lansări de produse), așa că modelul matematic bazat pe datele din urmă reprezintă un procent mic din adevărata valoare viitoare a datelor.

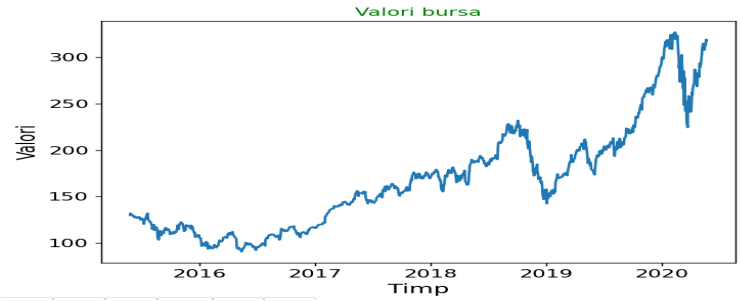

Figura 4.27: Valori bursă Apple.

4.3.3.1 SARIMA

Componenta care se poate identifica cel mai ușor din această serie de timp este staționazitatea. Se aplică funcția Dickey-Fuller și se observă că seria este nestationară. După diferențiere, aceasta devine stationară. Observarea corelogramei și corelogramei parțiale arată că în afară de ordinul 0, toate celelalte corelații sunt apropiate de valoarea 0. Acest lucru se regăsește și la zgomotul alb, semn că seria aceasta este greu de prezis și că depinde de fenomene haotice, greu de modelat matematic. Mai jos sunt prezentate corelogramele:

[8]Seria este disponibilă la `https://www.nasdaq.com/market-activity/stocks/aapl/historical`

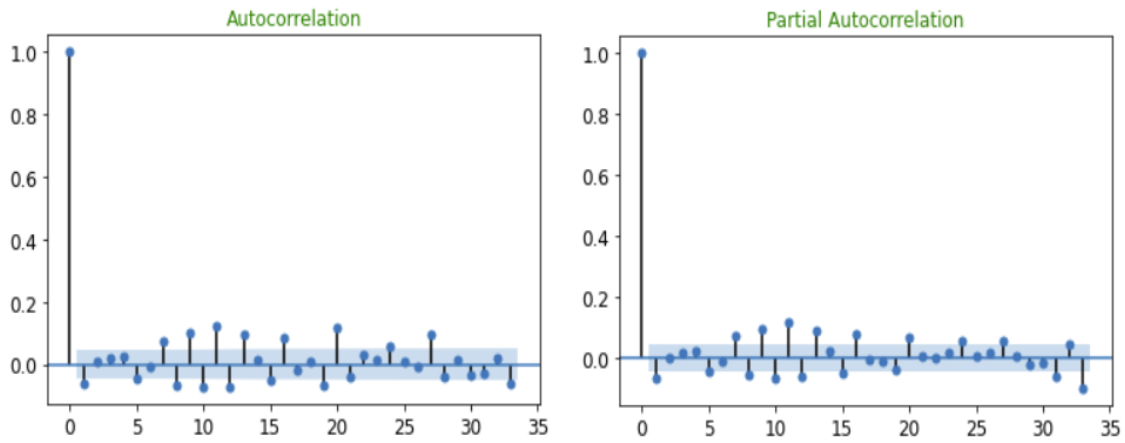

Figura 4.28: Corelograma & Corelograma parțială.

Se va incerca o căutare de tip *grid search* cu parametrii AR și MA cât mai variați. Partea de sezonalitate se va ignora, deoarece, neștiind perioada sezonalității, o sezonalitate greșită poate să facă mai mult rău decât bine. Așadar, modelul folosit va fi ARIMA(2,1,6). Graficul predicției este:

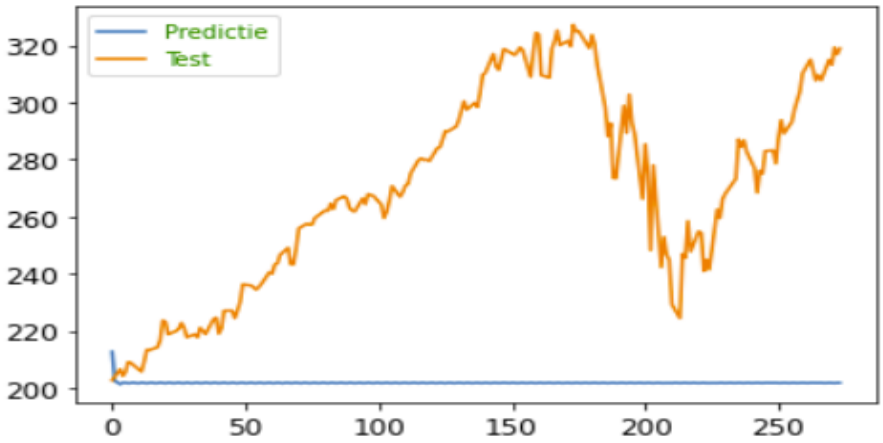

Figura 4.29: Predicții SARIMA.

Predicția approximated-itteration nu este altceva decât o linie dreaptă. Această linie dreaptă reprezintă ultima valoare din datele de antrenament, deoarece conform corelogramelor, ultima valoare nu depinde aproape deloc de valorile sau erorile din urmă, așa că modelul consideră că aceasta este cea mai bună predicție în aceste condiții. Acest fenomen este observat și în alte lucrari de specialitate. Calculul valorii MAPE nu are niciun rol în această predicție. Acest tip de date reprezintă limitele puterii acestei metode, întrucât componenta haotică a acestui semnal este foarte puternică, mult prea puternică pentru a putea fi modelată cu o regresie liniară.

4.3.3.2 MLP

Pentru rețeaua neurală s-au încercat diverse valori ale paramterilor. Aceste valori au fost mult mai aleatoare față de valorile alese la celelalte modele din cauza lipsei de informații despre serie. Cea mai bună combinație găsită a fost: Număr epoci = 100; Număr straturi=2; Număr nenuroni pe strat=15/15. Graficul valorilor prezise prin această metodă este:

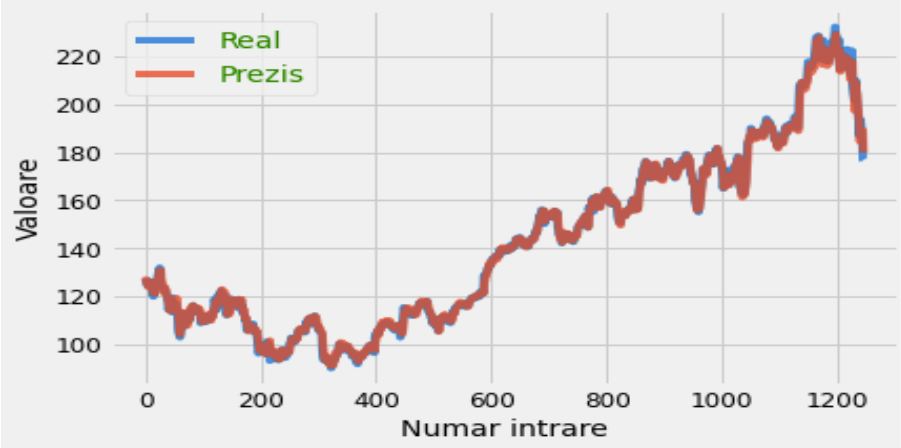

Figura 4.30: Predicții MLP.

Valoarea medie a MAPE este: 0.0094, iar graficul MAPE este:

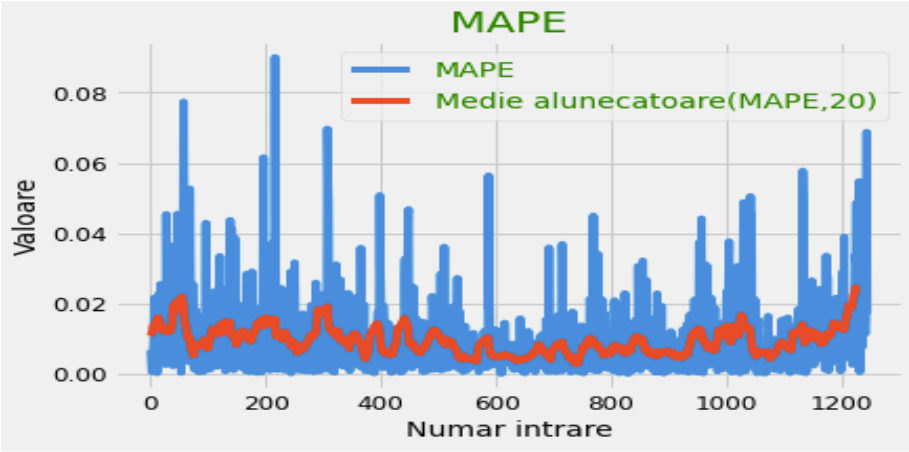

Figura 4.31: Valori MAPE.

Se observă o predicție foarte bună a valorilor, cu toate că seria este foarte dificil de prezis. Acest lucru se datorează multitudinii pe parametrii ai rețelei neurale. Faptul că această metodă prezintă rezultate foarte bune pe serii de timp haotice este menționat și de [9] în Fig. 2.10 și 2.11.

Capitolul 5

Concluzii și propuneri

În acest capitol se prezintă sub formă de concluzii observațiile legate de rezultatele experimentelor efectuate și de asemenea se propune o direcție de continuare a lucrării.

5.1 Concluzii

În urma experimentelor făcute în secțiunea trecută, se pot formula mai multe concluzii generale legate de folosirea celor două metode:

1. Ambele metode reprezintă două metode performante pentru predicția seriilor de timp univariate.
2. Au fost observate rezultate mai bune în cazul folosirii MLP, față de SARIMA, din punct de vedere al acurateții.
3. SARIMA necesită o cunoastere apriori a structurii seriei de timp pe care se dorește făcută predicția.
4. Folosirea SARIMA-ei necesită cunoștințe destul de avansate de statistică pentru a construi un model performant.
5. Modelul SARIMA este mult mai simplu decât cel construit cu MLP, el fiind liniar și având un număr de parametrii de ordinul unităților, față de modelul MLP care este neliniar și care poate ajunge să aibă un numar de parametrii de ordinul zecilor de mii.
6. Noțiunile de statistică folosite de SARIMA se regăsesc și în construcția modelului MLP: staționarizarea datelor este necesară la ambele metode, perioada sezonalității de la SARIMA se regăsește în numărul de neuroni de pe straturile ascune ale rețelei, autocorelația dintre date se regăsește în inputul dat rețelei sub formă de secvență,

construirea modelului cu ajutorul funcției de minimizare de la SARIMA se regăsește în minimizarea funcției de cost de la rețea. Toate aceste lucruri indică faptul că teoria din spatele SARIMA-ei este prezentă și la MLP, dar MLP se folosește mult mai bine de puterea de calcul actuală, alcătuind modele mult mai elaborate.

7. Se recomandă folosirea SARIMA-ei când seria de timp are un model ușor de observat și se pot aplica concepte statistice avansate pentru determinarea modelului cel mai bun.

8. Se recomanda folosirea MLP când seria de timp este una complicată, cu foarte multe date de intrare și când nu se cunosc prea multe concepte statistice care să faciliteze construirea unui model la fel de performant, dar liniar, cu mult mai puțini parametrii.

9. Timpul de rulare este mult mai mic la construcția modelului SARIMA, decât la MLP, lucru intuitiv având în vedere numărul de paramterii care trebuie calculați în cazul MLP.

5.2 Propuneri

O direcție de continuare a dezvoltării ideii abordate în această lucrare o reprezintă documentarea și studierea competițiilor de Makridakis (Competițiile-M)[1]. Aceste competiții constau în modificarea metodelor deja cunoscute de predicție a seriilor de timp, prin combinarea mai multor metode, modificarea parametrilor sau chiar conceperea unei noii metode și testarea respectivelor metode pe seturi de date stabilite de un juriu pentru testarea diverselor calități ale metodelor. Până în momentul de față au fost în total 5 astfel de competiții, prima fiind în 1982, iar ultima în 2020. Metodele prezentate în cadrul competițiilor nu sunt făcute publice, decât după un anumit timp de la momentul desfășurării acestora, în momentul de față rezultatele ultimei competiții nefiind disponibile gratis pe internet.Cu toate acestea, analizarea metodelor prezentate în cadrul lor, chiar și de la edițiile mai vechi, va asigura o viziune completă asupra întrebării retorice formulată în introducerea acestei lucrări, și anume: *Cine deţine avantajul acurateţii în momentul de faţă în ceea ce priveşte predicţia de date?*

[1]Site-ul oficial al competițiilor: `https://mofc.unic.ac.cy/m5-competition/`

Capitolul 6

Bibliografie

[1] Chris Chatfield, "Time-series forecasting", Chapter 1, Octombrie 2000, editura CHAPMAN& HALL/CRC. Disponibil la https://books.google.ro/books?hl=ro&lr=&id=PFHMBQAAQBAJ&oi=fnd&pg=PR7&dq=time+series+forecasting&ots=fYdn9TbCvd&sig=ah2pmJ7oyseSDvoIZPM2woVAnf4&redir_esc=y#v=onepage&q&f=false

[2] Chris Chatfield, "Time-series forecasting", Chapter 2, Octombrie 2000, editura CHAPMAN& HALL/CRC. Disponibil la https://books.google.ro/books?hl=ro&lr=&id=PFHMBQAAQBAJ&oi=fnd&pg=PR7&dq=time+series+forecasting&ots=fYdn9TbCvd&sig=ah2pmJ7oyseSDvoIZPM2woVAnf4&redir_esc=y#v=onepage&q&f=false

[3] Tudor Sebastian, Stefănoiu Dan, Culiță Janetta, "Abordări experimentale în identificarea proceselor și fenomenelor", Septemnbrie 2012

[4] Xiang Wan, Wenqian Wang, Jiming Liu, Tiejun Tong, "Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range", 2014, BMC Medical Research Methodology.

[5] J.R.M Hosking, "Asymptotic distributions of the sample mean, autocovariances and autocorrelations of long-memory time series", September 1984, University of Wisconsin-Madison mathematics research center. Disponibil la: https://apps.dtic.mil/sti/pdfs/ADA149409.pdf

[6] Fred L. Ramsey, "Characterization of the partial autocorrelation function", The Annuals of Statistics, 1974, Vol. 2, No. 6. Disponibil la: https://projecteuclid.org/download/pdf_1/euclid.aos/1176342881

[7] Bogdan Dumitrescu, Corneliu Popeea, Boris Jora, "Metode de calcul numeric metriceal. Algoritmi fundamentali", 1998, Editura ALL.

[8] Lon-Mu Liu, Gregory B. Hudak, "Forecasting and time series analysis using the SCA statistical system", Volume 1, Scientific Computing Associates Corp.,1992. Disponibil la http://scausa.com/SCADocs/SCAFTS_V1.pdf

[9] Antonio Rafael Sabino Parmezana, Vinicius M. A. Souzaa, Gustavo E.A.P A. Batistaa, "Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model", January 2019. Disponibil la https://www.researchgate.net/profile/Antonio_Parmezan/publication/330742498_Evaluation_of_statistical_and_machine_learning_models_for_time_series_prediction_Identifying_the_state-of-the-art_and_the_best_conditions_for_the_use_of_each_model/links/5c558145a6fdccd6b5dc3e2e/Evaluation-of-statistical-and-machine-learning-models-for-time-series-predicti
pdf

[10] Ani Katchova, "Time Series ARIMA Models",2013. Disponibil la https://drive.google.com/file/d/0BwogTI8d6EEiaDJCRXd0dmU1ZDA/edit

[11] M. B. Priestley, T. Subba Rao, "A Test for Non-Stationarity of Time-Series", Journal of the Royal Statistical Society. Series B (Methodological), Vol. 31, No. 1, 1969. Disponibil la https://www.stat.tamu.edu/~suhasini/test_papers/priestley_subbarao70.pdf

[12] Billy M. Williams,Lester A. Hoel, "Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results", 2003. Disponibil la http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.4524&rep=rep1&type=pdf

[13] Yin-Wong Cheung, Kon S. Lai, "Lag Order adn Critical Values of the Augmented Dickey-Fuller Test", Journal of Business & Economic Statistics, Iulie 1995, Vol. 13, No. 3. Disponibil lahttp://www.kslai.net/csula/KLPaper/JBES95Jy.pdf

[14] Ben Lambert, "A Students Guide to Bayesian Statistics Paperback ", 4 Mai 2018. Disponibil la https://www.youtube.com/watch?v=jbT5MDzoRy0

[15] Marco Lippi, Matteo Bertini, and Paolo Frasconi, "Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning", 2013

[16] Samuel Olorunfemi Adams, Muhammad Ardo Bamanga, "Modelling and Forecasting Seasonal Behavior of Rainfall in Abuja, Nigeria; A SARIMA Approach", American Journal of Mathematics and Statistics, Octombrie 2019. Disponibil la http://article.sapub.org/10.5923.j.ajms.20201001.02.html#Sec4.2

[17] D. Dumitrescu, Hariton Constin, "Rețele neuronale", Editura Teora, 1996.

[18] Ion Necoară, "Tehnici de optimizare", 2013. Disponibil la https://www.scribd.com/document/223391081/Tehnici-de-optimizare-Necoara

[19] Seria de timp folosită este disponibilă la https://catalog.data.gov/dataset/monthly-energy-consumption-by-sector

[20] Chiou-Jye Huang,Ping-Huan Kuo 2,"A Deep CNN-LSTM Model for Particulate Matter (PM2.5) Forecasting in Smart Cities", Iulie 2018