# **Aplicatia 1.1: Prognoza univariată a stocurilor unei compani prin intermediul rețelelor neuronale, modelului SARIMA și tehnicilor de netezire exponențială**

## **1. Metodologie**

**Date Folosite:**

În vederea realizării studiului, am optat să folosesc date privind acțiunile companiei Google pentru implementarea și antrenarea modelelor, urmând ca apoi să testez modele gata antrenate pe acțiunile NFLX; META și TSLA.

Datele au fost preluate de pe yahoo finance și cuprind intrări zilnice din luna Iulie 2023 până în Mai 2024.

| Date |
| --- |
| Open |
| High |
| Low |
| Close |
| Adj Close |
| Volume |

Tabel 4.1: Conținut GOOGL - prelucrare personală

Pe baza acestui set de date, am realizat o serie de timp a prețului de închdere (close) GOOGL.

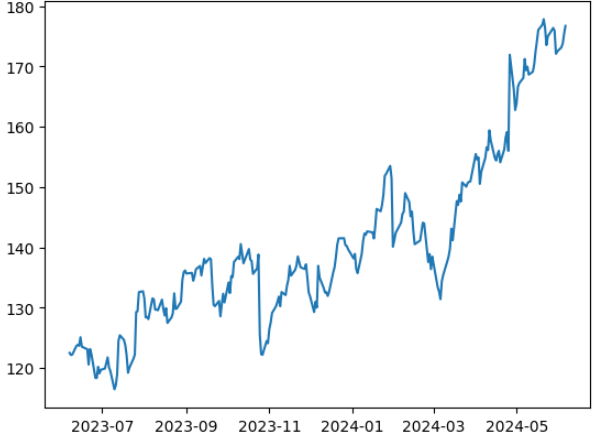


Figura 4.1: Serie de timp GOOGL - prelucrare personală în python

### **Elaborarea Modelului Sarima:**

I) Analiza staționarității și a sezonalității

| Test | P-value |
| --- | --- |
| ADF Test | 0.8519 |
| Ljung-Box Test | ~0.0 |
| Box-Pierce Test | ~0.0 |
| Phillips-Perron Test | 0.935 |
| KPSS Test | 0.01 |

Tabel 4.2: Teste staționaritate pe seria de timp GOOGL - prelucrare personală

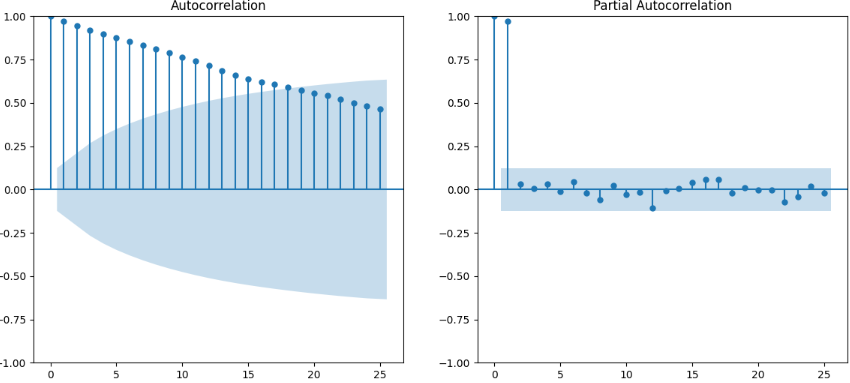


Figura 4.2: ACF și PACF GOOGL - prelucrare personală în python

După cum se poate observa toate testele realizate indică prezența pronunțată a nestaționarității în seria de tip GOOGL, iar după cum ilustrează graficul ACF din figura 4.2, exista autocorelație care scade succesiv în intensitate între laguri, indicând astfel prezența clară a unui trend.

Pentru a trata inițial această problemă, am decis să logaritmez seria de timp apoi să vizualizez componenta sezonieră.

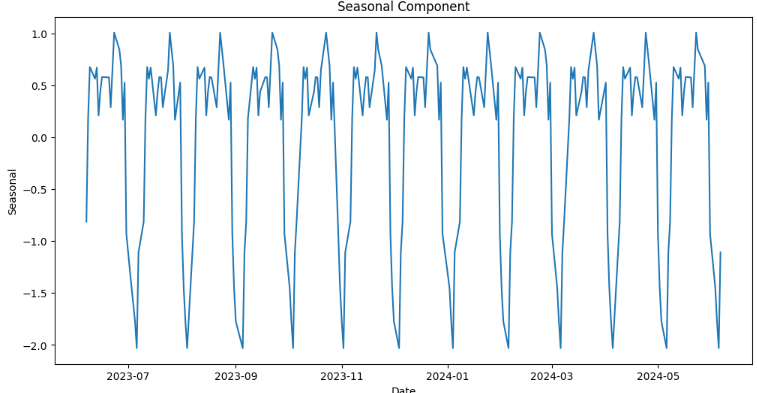


Figura 4.3: Componenta sezonieră a seriei de timp GOOGL - prelucrare personală în python

Figura de mai sus ilustrează prezența clară a unui pattern sezonier stabil, așadar am decis să continui prin a diferenția sezonier seria de timp pentru o perioadă de 21 zile, dat fiind faptul că bursa este deschisă tranzacțiilor doar în zilele lucrătoare ale săptămânii.

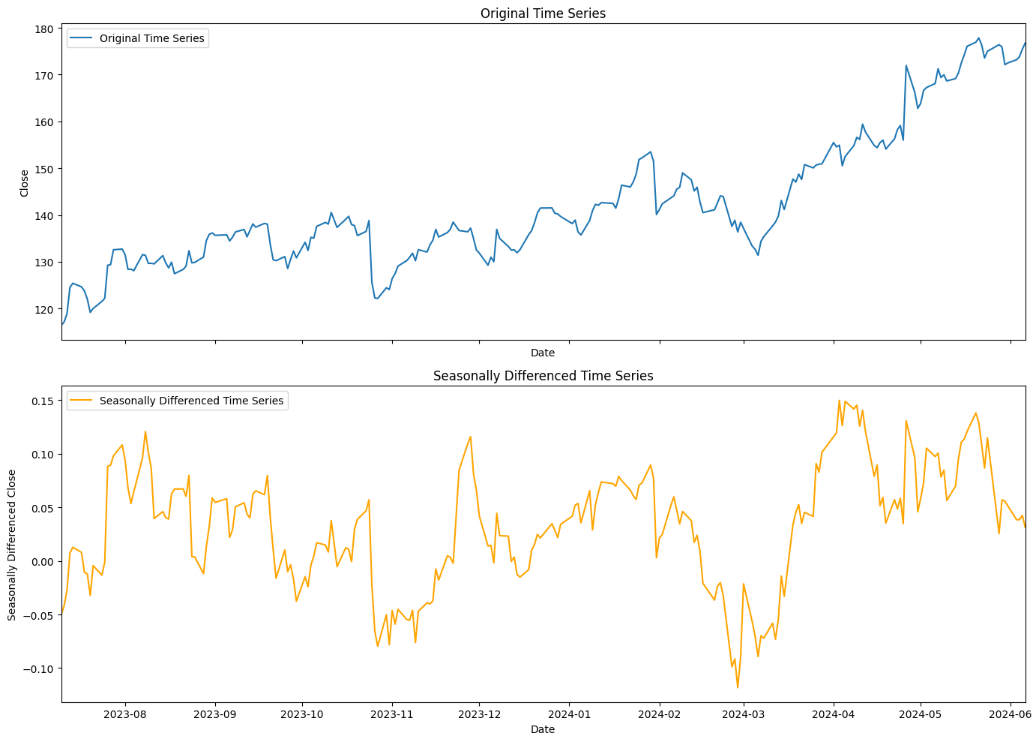


Figura 4.4: Comparație între seria de timp originală GOOGL și cea ajustată sezonier - prelucrare personală în python

Așa cum este ilustrat în figura 4.4, diferențierea sezonieră a rezultat în schimbări semnificative atât în materie de medie cât și dispersie, așadar am optat să realizez testele de staționaritate și reprezentarea grafică a ACF și PACF încă o dată.

| Test | P-value |
| --- | --- |
| ADF Test | 0.00016 |
| Ljung-Box Test | ~0.0 |
| Box-Pierce Test | ~0.0 |
| Phillips-Perron Test | 0.002 |
| KPSS Test | 0.0557 |

Tabel 4.3: Teste staționaritate pe seria de timp GOOGL diferențiată sezonier - prelucrare personală

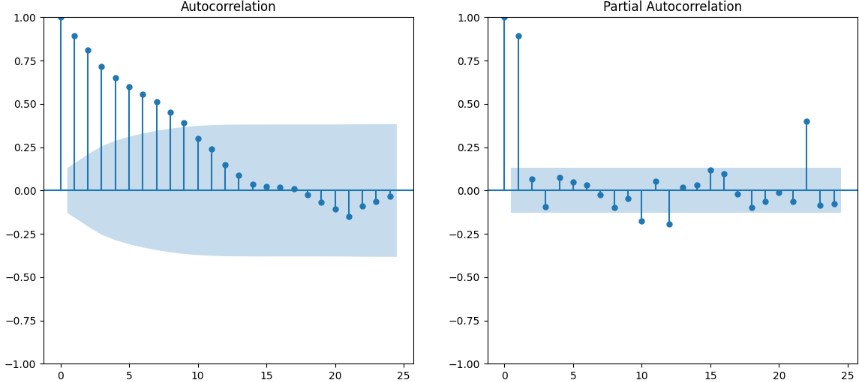


Figura 4.5: ACF și PACF GOOGL ajustat sezonier - prelucrare personală în python

Pe baza rezultatelor din tabelul 4.3, ipoteza nulă a testului ADF cum că ar fi prezentă rădăcina unitate este refuzată, dar și cea a Testului PP, însă ipoteza nulă a testului KPPS, cum că seria este staționară este la limita de a fi respinsă, iar testele Ljung-Box și Box-Pierce, precum și graficele ACF și PACF din figura 4.5, indică prezența unei autocorelații semnificative, trend-ul nefiind eliminat complet. Așadar, este indicat ca seria ajustată sezonier să fie diferențiată.

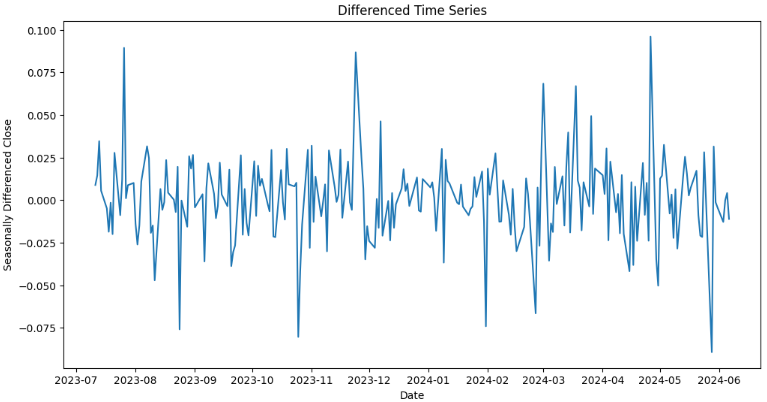


Figura 4.6: Seria GOOGL în urma diferențierii - prelucrare personală în python

| Test | P-value |
| --- | --- |
| ADF Test | 4.5696 \* |
| Ljung-Box Test | 0.0124 |
| Box-Pierce Test | 1.5739× |
| Phillips-Perron Test | ~0.000 |
| KPSS Test | 0.1 |

Tabel 4.4: Teste staționaritate pentru seria diferențiată GOOGL

Rezultatele testeleor ADF, PP, și KPPS indică în mod cert că seria este staționară în urmă diferenșierii, însă atât testele Ljung-Box și Box-Pierce, cât și graficele ACF și PACF arată că încă exista autocorelație între anumite laguri, aceasta poate să fie neglijită totuși.

Se poate proceda cu implementarea unui model SARIMA.

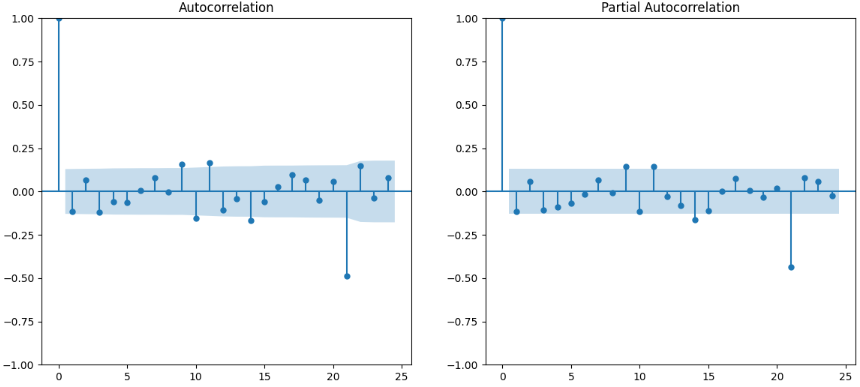


Figura 4.7: ACF și PACF GOOGL diferențiat - prelucrare personală în python

II) Proprietățiile modelului

Vizualizând lag-urile din graficele ACF și PACF din figura 4.7, se poate observa ca ordinul maximal pe care îl pot lua parametrii unui model SARIMA sunt:

* p = 1
* d = 1
* q = 1
* P = 1
* D = 1
* Q = 1

Pe baza mai multor încercari, am determinat ca modelul SARIMA(1,1,0)(1,1,0)[21] este cel mai puțin problematic.

| Parameteru | Coeficient | Eroare | p-value |
| --- | --- | --- | --- |
| ar.L1 | 0.0833 | 0.070 | 0,232 |
| ar.S.L21 | -0.5310 | 0.040 | ~ 0 |
| sigma2 | 7.6245 | 0.510 | ~ 0 |

Tabelul 4.5: Estimările modelului - prelucrare personală

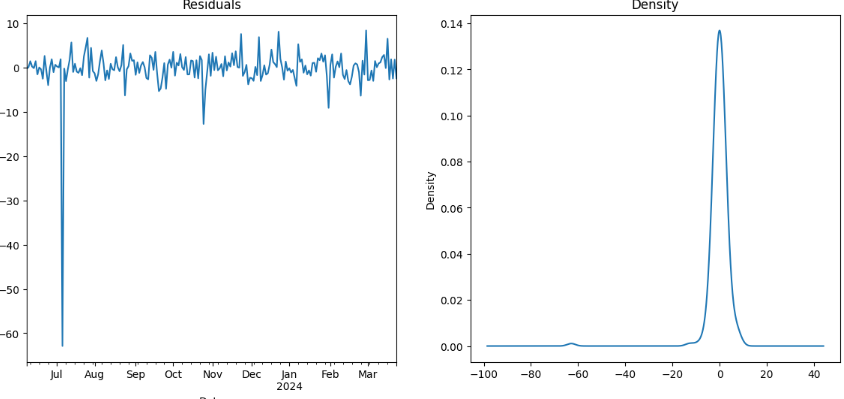


Figura 4.8: Analiza rezidurilor modelului SARIMA - prelucrare personală în python

În tabelul 4.5 se poate observa ca coeficientul atribuit componentei autoregresive nesesozniere este nesemnifiicativ statistic, de asemenea, figura 4.8 indică ca rezidurile modelului nu urmează o distribuție normală, acestea prezentând asimetrie la stanga și o distribuție letokurtică.

Prin urmare, aplicarea modelului SARIMA nu reprezintă o metodă optimă pentru previzunea pretului acțiunilor GOOGL, însă în cadrul acestei părți a studiului de caz am continuat să il folosesc în cadrul unei previziuni în scop pur demonstrativ.

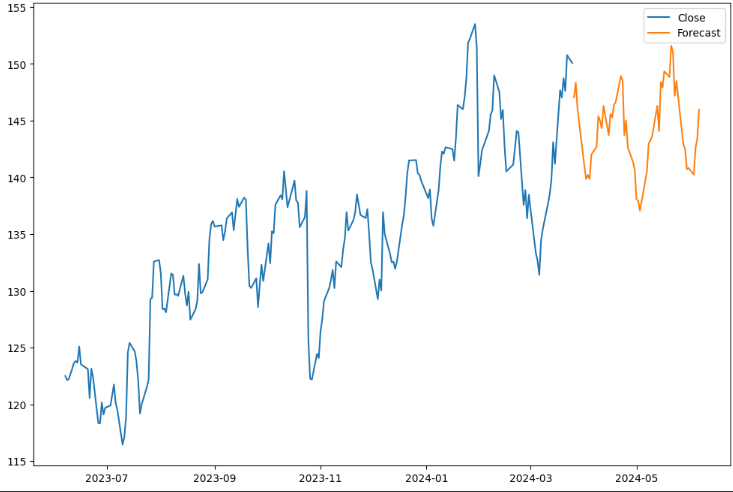


Figura 4.9: Prognoză cu modelul SARIMA - prelucrare personală în python

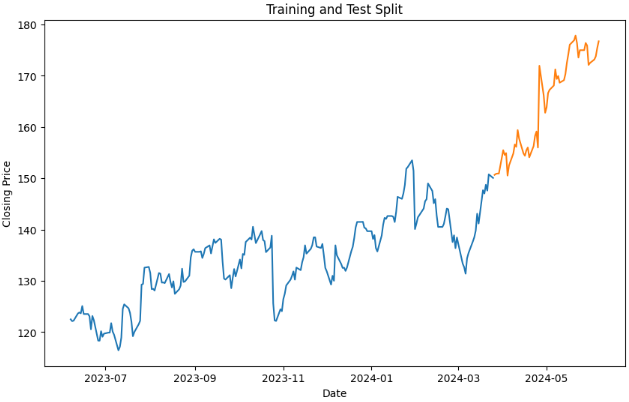


Figura 4.10: Împărțirea setului de date în antrenare și testare - prelucrare personală în python

Elaborarea Modelului Holt-Winters aditiv și multiplicativ:

În cadrul antrenării fiecăruit model din acest studiu de caz, fie el statistic sau de deep learning, datele de antrenare cuprind primele 80% de observații din seria de timp, iar cele de testare ultimele 20%.

Atât modelul aditiv, cât și cel multiplicativ a primit ca parametrul pentru sezonalitate valoarea 21, corespunzătoare cu zilele lucrătoare ale lunii.

Cele două modele Holt-Winters au avut parte de o performanță foarte apropiată, din acest motiv am ales să includ mai sus doar un grafic al prognozei cu modelul aditiv urmând să fac o revenire la modelul multiplicativ în figura comparativă a prognozelor finale.

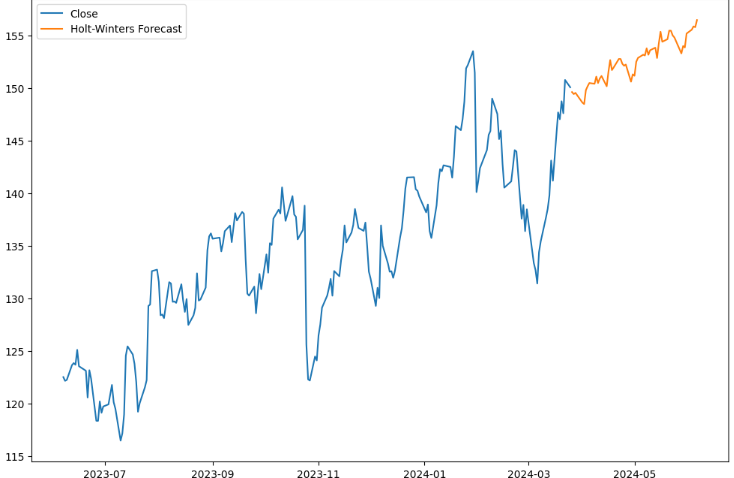


Figura 4.11: Prognoza seriei de timp prin intermediul Holt Winters aditiv - prelucrare personală în python

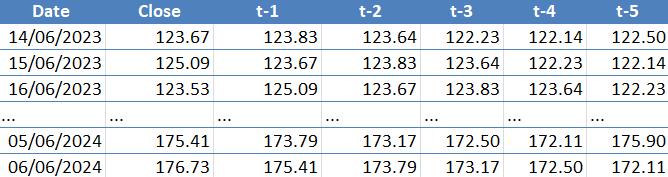
### 

### **Elaborarea modelelor de deep learning:**

1) Prelucrarea Datelor:

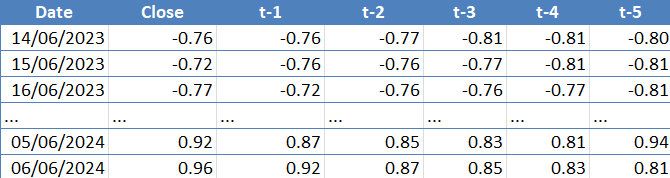
Această etapă are adesea cel mai mare impact asupra calității previziunii, necesitând aplicarea unei abordări cât mai robuste și relevante pentru seria de tip cu care se lucrează. În cadrul acestui am încercat să păstrez un nivel de complexitate cât mai mic pentru această etapă, dată fiind natura limitată a prognozei univariată, mai ales în contextul deep learning-ului.

Pentru început am realizat o funcție care să convertească observațiile setului de date în secvențe care conțin valoarea curentă și ultimele 5 valori.



Tabelul 4.6: Set de date secvențial - prelucrare personală

Odată ce am creat noul set de date, l-am supus la procesul de normalizare pentru a trata mai bine fluctuațiile/modificările față de perioada de start. Tehnica de normalizare aleasă a fost minmax cu valori cuprinse în intervalul [-1;1].



Tabelul 4.7: Setul de date după normalizare - prelucrare personală

Am continuat prin a realiza împărțirea setului de date, în antrenament și testare, dar și pentru variabilele indpendente ( valorile anterioare de la t-1 până la t-5 ) și cea dependentă ( Close ).

A urmat apoi împărțirea în cate 5 batch-uri de antrenat a fiecărei variabile declarate.

2) Definirea rețelelor neuronale

În cadrul lucrării am ales să folosesc cele 3 tipuri de rețele neuronale despre care am vorbit la capitolul 2, mai exact RNN, GRU și LSTM.

Pentru a limita cât de mult se poate diferențele dintre ele, le-am proiectat pe toate cu structura de forma: 1 neuron input => 4 straturi ascunse => 1 neuron output.

De precizat că ponderile inițiale sunt setate ca 0.

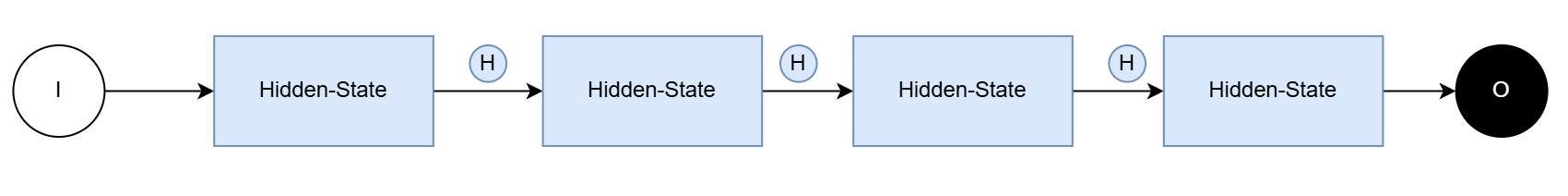


Figura 4.12: Structura rețelelor neuronale - prelucrare personală în draw.io

Caracteristicile meta parametrilor:

* Numărul de epoci de antrenament este 30
* Rata de învățare setată este 0.001
* Funcția de pierdere este redată de MSE
* Optimizatorul folosit este ADAM

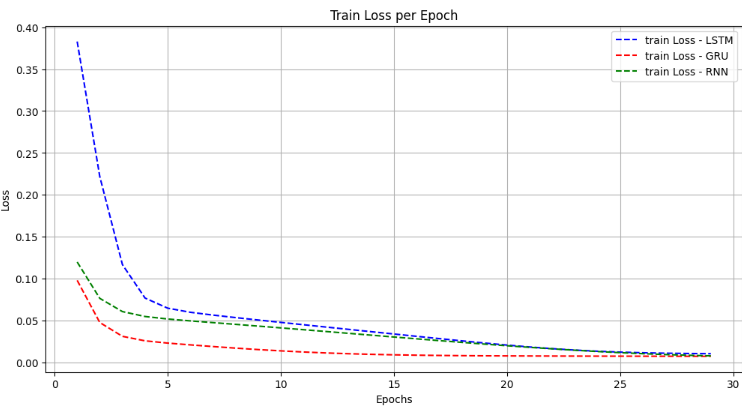


Figura 4.13: valoare funcției de pierdere în timpul antrenamentului - prelucrare personală în python

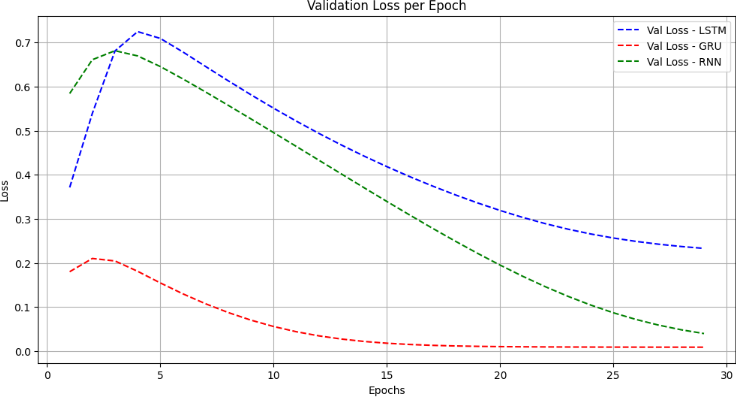


Figura 4.14: valoare funcției de pierdere în timpul validării - prelucrare personală în python

Se poate observa ca toate cele 3 modele au performanță similară în timpul antrenamentului, însă în perioada validării, GRU pare să învețe caracteristicile modelului mai bine față de LSTM și RNN.

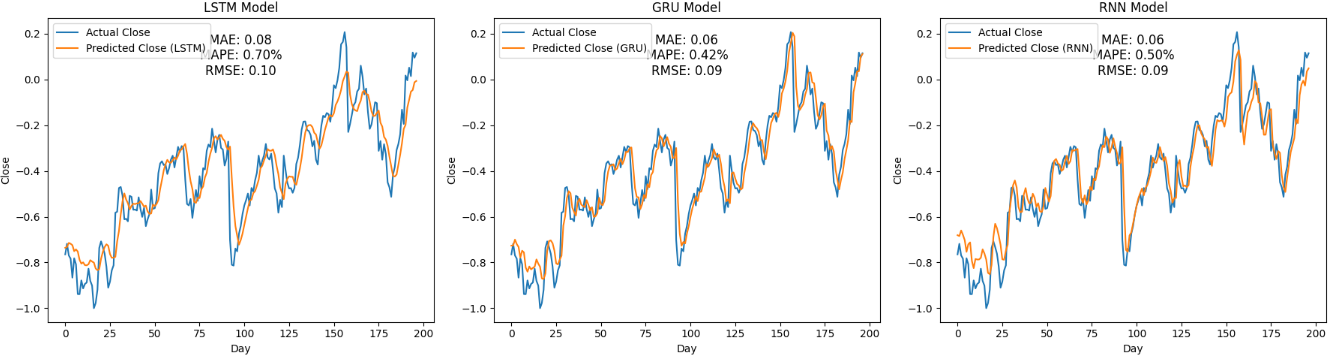


Figura 4.15: Comparație între acuratețea celor 3 modele pe datele de antrenament - prelucrare personală în python

Performanța rețelei de tip GRU este evidențiată clar în figura 4.15, find cea mai apropiată de valoarea reală și având cele mai mici erori de previziune.

Pentru a analiza mai amănunțit intervalul de previziuni făcut de fiecare rețea în parte am realizat o serie de histograme și boxplot-uri.

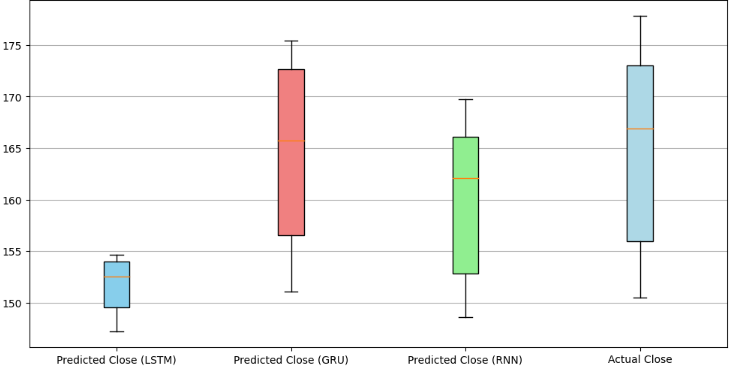


Figura 4.16: Boxploturi ale valorilor prezise de rețelele neuronale - prelucrare personală în python

În figură se poate observa o diferență foarte pronunțată între raza de valori pe care LSTM-ul a prezis-o față de cea a celorlalte rețele, de unde și performanța mai slabă de care a dat dovadă.

De asemenea, niciuna dintre rețele nu a prezis o valoare de tip outlier.

Rețeaua de tip GRU are pe departe cea mai similară distribuție cu cea a valorilor reale.

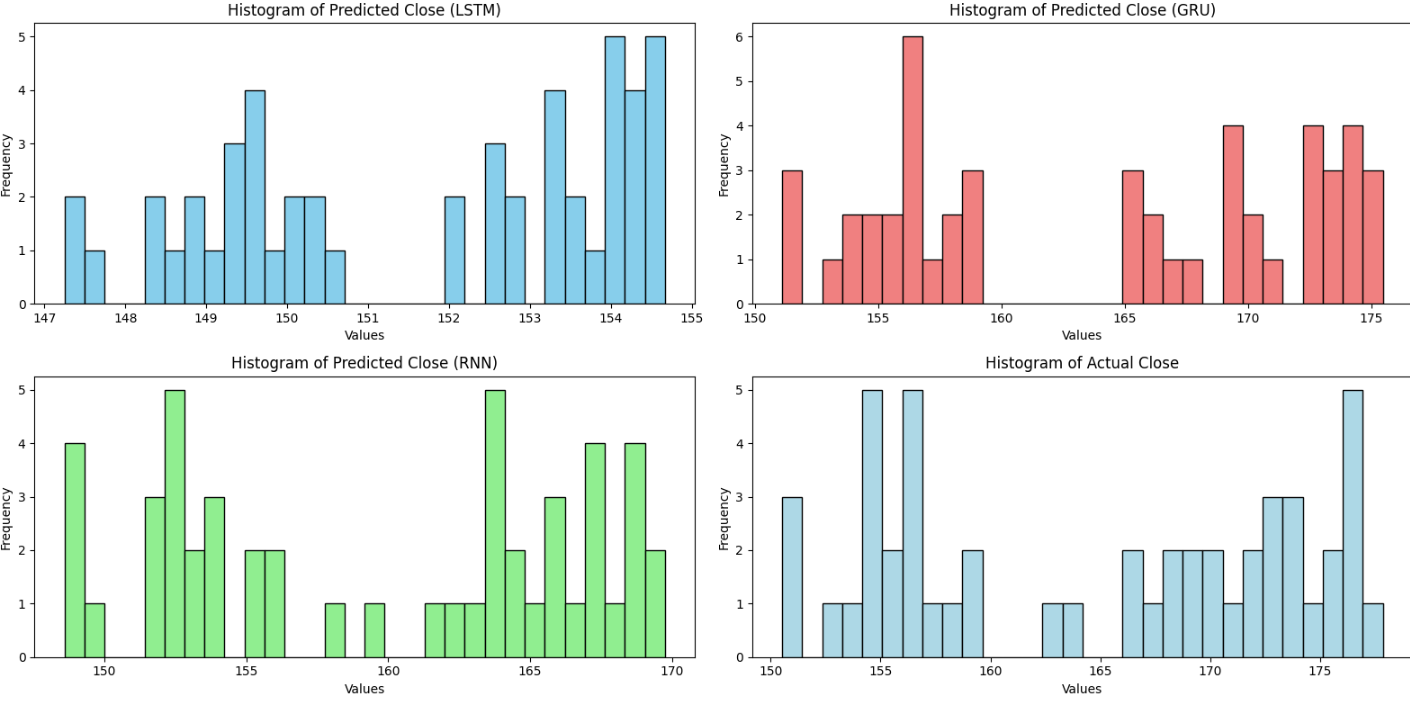


Figura 4.17: Histograme ale valorilor prezise de rețelele neuronale - prelucrare personală în python

În figură se poate observa ca, în timp ce toate seturile de date prezintă distribuții distincte, toate tipurile de rețele neuronale identifică corect că valorile apropiate de medie sunt rare spre inexistente, principala problemă a valorilor prezise este ca rețelele nu au reușit să identifice corect tipul de asimetrie pe care îl prezintă datele originale, modelul cel mai apropiat de acest obiectiv, find GRU.

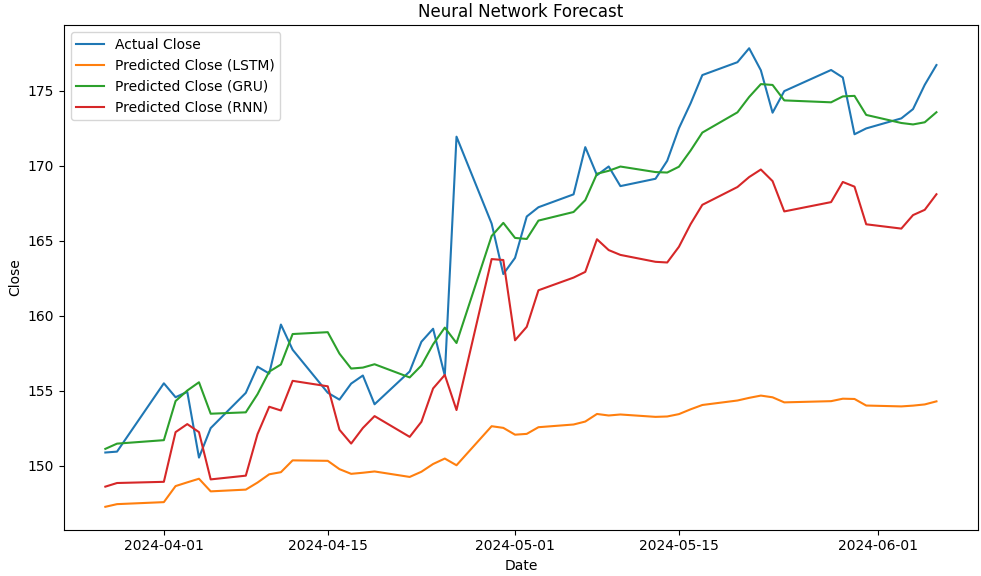


Figura 4.18 Comparație a prognozelor realizate de cele 3 rețele neuronale - prelucrare personală în python

## **2. Rezultate**

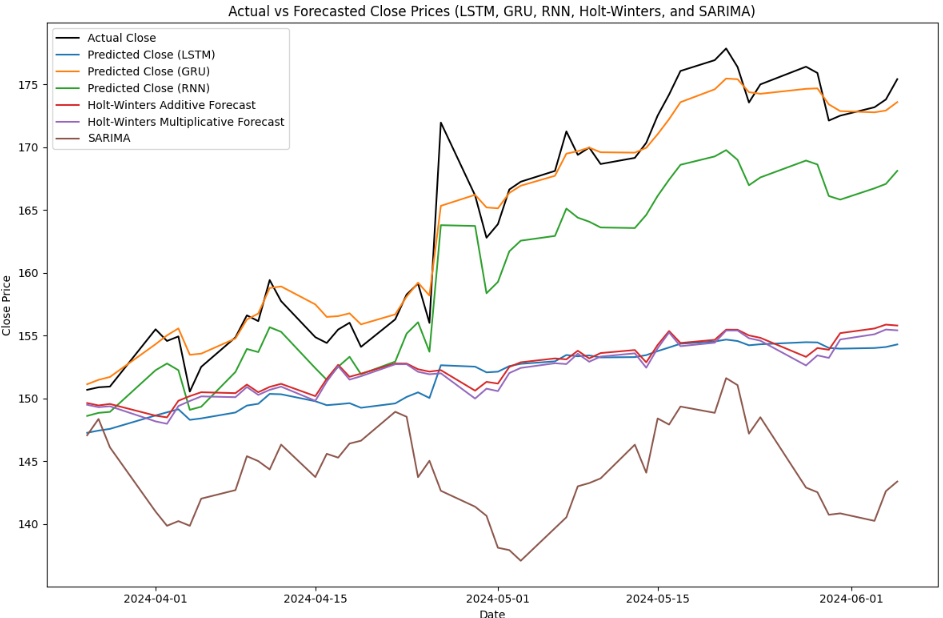
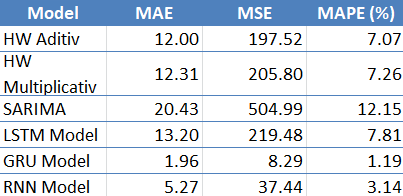


Figura 4.19: Comparație între toate modelele pentru previziune univariată - prelucrare personală în python

Date fiind figurile pentru previziunile finale, cât și tabelul cu erori, putem deduce cu certitudine că rețelele neuronale au avut parte de o performanță superioară față de modelele statstice, în special rețeaua GRU.



Tabelul 4.8: Erorile de previziune

3) Testarea modelului cel mai eficient pe alte seturi de date

Pentru a vedea într-adevăr cât de eficient este modul GRU pe care l-am antrent, l-am folosit pe datele acțiunilor NFLX, TSLA și META.

După cum se poate vedea în figura 4.19, modelul GRU pe care l-am proiectat anterior a avut parte de un succes surprinzător în previziunea prețului META și NFLX, însă a întâmpinat câteva dificultăți în prognoza prețului TSLA, lucru cel mai probabil dat de diferența semnificativă între distribuția prețului TSLA și cel de GOOGL.

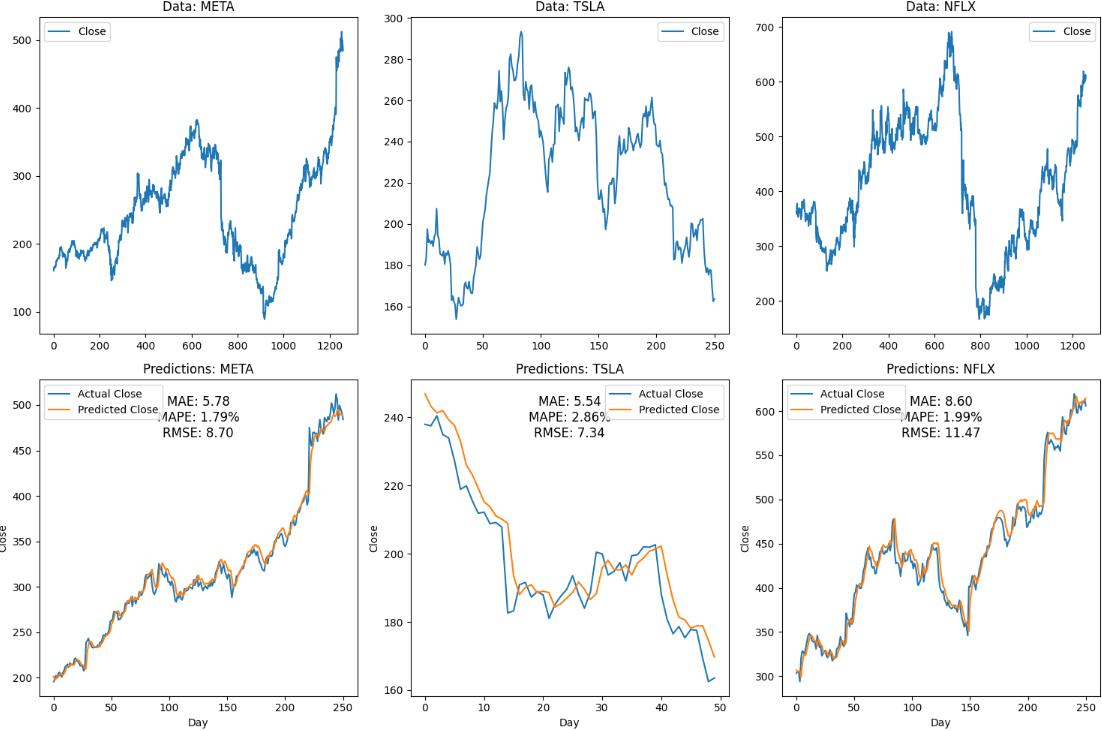


Figura 4.20: Testarea modelului GRU pe datele altor companii - prelucrare personală în python

Limitări ale studiului:

Cu toate că acuratețea modelelor de deep learning pare foarte ridicată, nu se poate negă impactul semnificativ al fenomenului de overfitting asupra acestui tip de previziune. Modelul a învățtat doar pe baza valorilor istorice ale prețului de închidere, ceea ce nu este nici pe departe suficient pentru a previziona pe termen lung (chiar și scurt) un indicator atât de volatil la schimbări economice și de natură socială.

Concluzii:

Raportându-ne strict la contextul previziunii univariate, am reușit să dovedesc cât de mult potențial are deep learning-ul pentru a învăța traiectoria istorică a prețului acțiunilor, făcând comparații strict cu trei metode statistice de previziune univariată.

# **Aplicatia 1.2: prognoza prețului BITCOIN prin intermediul unei combinații între ARCH-GARCH și rețele neuronale**

## **1. Descrierea Problemei**

Acestă extindere a primului studiu de caz are ca scop modelarea mai detaliată a datelor cu caracter financiar, axându-se pe îmbunătățirea performanței unui algoritm de deep learning prin combinarea sa cu un model statistic destinat predicției volatilității.

## **2. Metodologie**

Setul de date ales pentru exemplificarea acestei problematici, este prețul bitcoin în dolari americani, datorită volatilității sale renumite. Acesta a fost preluat de pe yahoo finance și cuprinde observații înregistrate în perioada 07.06.2022 - 07.06.2024.



Figura 4.21: Seria de timp a prețului de închidere BITCOIN în USD - prelucrare personală în python

### **Elaborarea modelului ARCH-GARCH:**

I Pregătirea rentabilitățiilor:

Pentru început, am calculat rentabilitățiile efectuând procesul de diferențiere pe seria de timp logartimată.

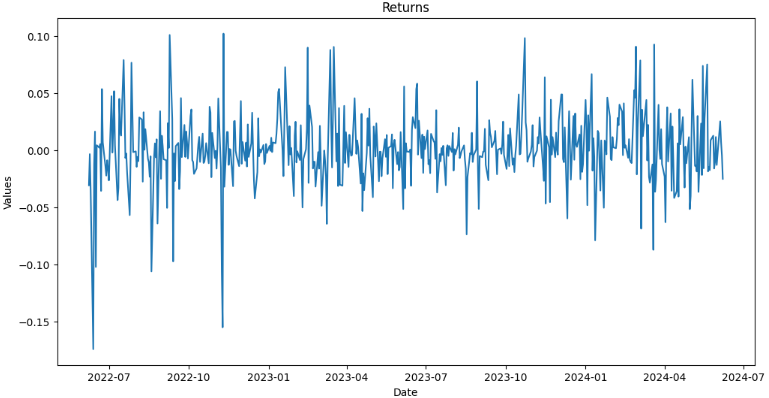


Figura 4.22: Seria de timp a rentabilitățiilor - prelucrare personală în python

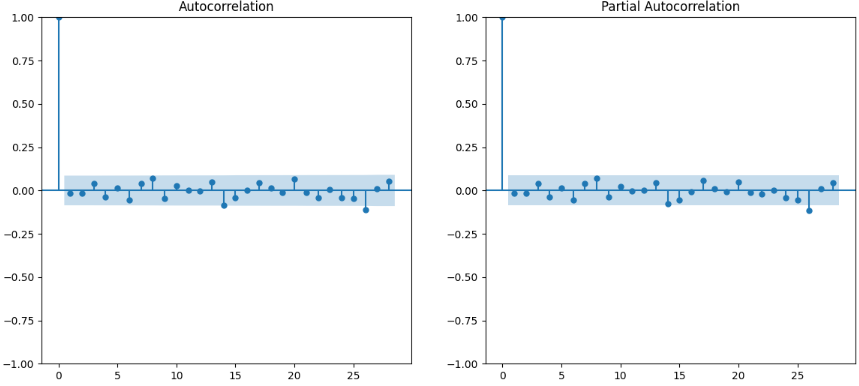


Figura 4.23: ACF și PACF rentabilități - prelucrare personală în python

Seria de timp încă pare nestaționară în dispersie, chiar și după logaritmare și diferențiere, dat find acest lucru, am decis să aplic testele de staționaritate.

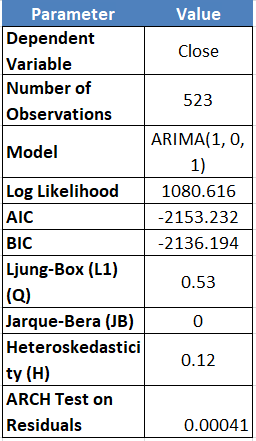
| ADF | 0 |
| --- | --- |
| Ljung Box Test | 0.58 |
| Box-Pierce Test | ~0 |
| Phillips-Perron | 0 |
| KPPS | 0.1 |

Tabelul 4.9: Teste de staționaritate pentru rentabilități

Toate testele indică că seria rentabilitățiilor este staționară, se poate trece la următorul pas.

II. Estimarea modelului

Pentru a putea determina parametrii optimi/admisibili pentru ARCH-GARCH, trebuie mai întăi sa se realizeze o estimare pe baza rezidurilor modelului ARIMA(1,0,1).



Tabel 4.10: Statistici ale modelului ARIMA(1,0,1) - prelucrare personală în python

Din Tabelul 4.10, rezultă ca modelul ARIMA conceput este semnificativ statistic, lucru determinata de valoarea foarte ridicată a parametrului Log Likehood și valoarea extrem de redusă a testelor AIC și BIC, iar testul Ljung-Box indică ca nu există autocorelație puternică între laguri, în schimb valoarea mai mică decat 0,1 a testului Jarque-Bera, dar și figura 4.22, indică că rezidurile nu au o distribuție normală.

De asemenea, rezidurile prezintă heteroskedasticitate, lucru care rezultă atât din figura, cât și din tabel. Un alt detaliu important, il reprezintă prezenta efectelor arch, testul ARCH returnând o valoare mult mai mică decât 0,05, așadar, se poate merge la pasul următor.

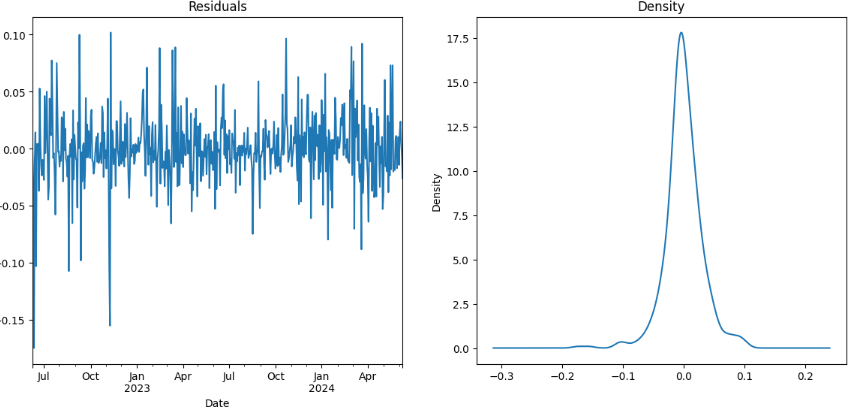


Figura 4.24: Diagnostic reziduri ARIMA(1,0,1) - prelucrare personală în python

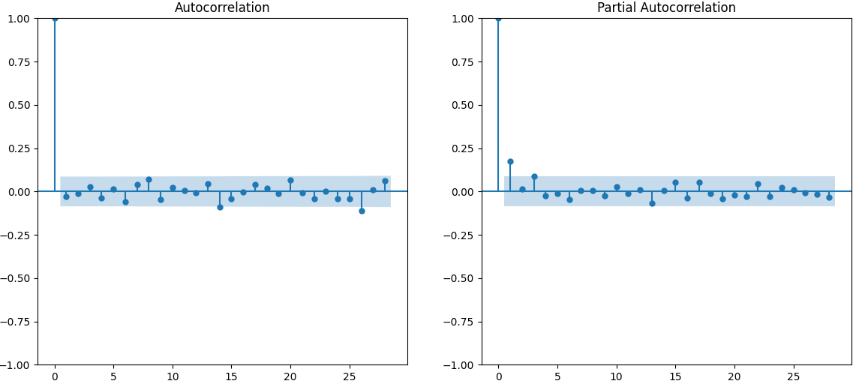
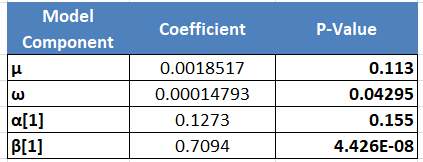


Figura 4.25: ACF și PACF reziduri ARIMA(1,0,1) - prelucrare personală în python

Pe baza graficului PACF, inclus în figura 4.25, se poate determina ordinul maximal al parametrilor ARCH-GARCH, având în vedere că in PACF, primele două laguri prezintă autocorelare, parametrul pentru ARCH poate să ia maxim valoarea 2.

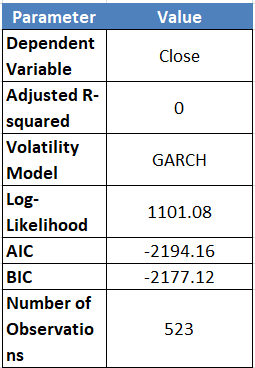
III. Crearea modelului



Tabelul 4.11: Coeficienții modelului

În tabelul de mai sus sunt trecuți indicatorii care formează ecuația volatilității conditționate.

Coeficienții μ și α[1] au o valoare p mai mare decât 0,1, așadar nu sunt semnificativi statistici, prin urmare nu se poate respinge ipoteza nulă că media ar fi diferită de 0 și respectivă că socurile din trecut nu au un efect semnificativ asupra modelului, în schimb ω și β sunt semnificativi din punct de vedere statistic, prin urmare, demonstrează existența unui nivel de volatilitate de bază și respectiv ca volatilitatea din trecu o afectează pe cea din prezent.



Tabelul 4.12: Statistici ale modelului GARCH(1,1)

În cadrul tabelului 4.12, sunt incluse informații generale despre starea modelului, mai exact log likelihood arată cât de semnificativ/adecvat este modul pentru seria de timp asupra căreia a fost aplicat, iar testele AIC și BIC, având în vedere valorile lor foarte mici, indică, de asemenea că modelul este unul robust și adecvat.

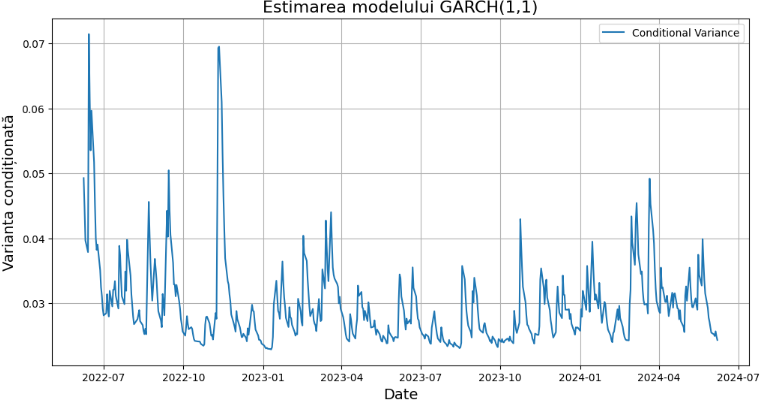


Figura 4.26: Volatilitatea condiționată esimată de modelul GARCH(1,1) - prelucrare personală în python

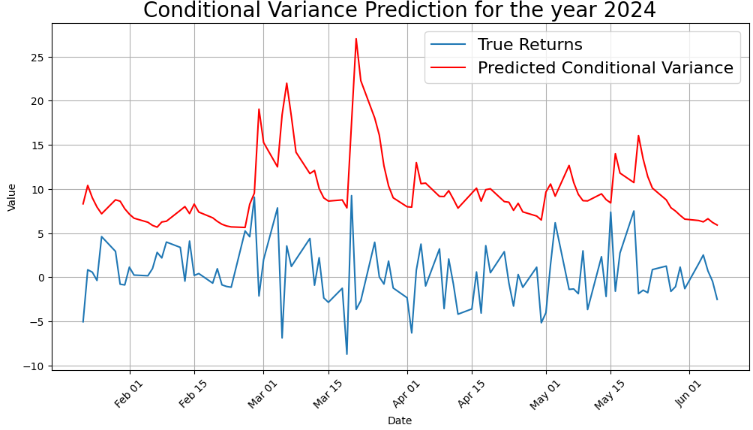


Figura 4.27: Volatilitate conditionata estimată comparată cu rentabilității din 2024 - prelucrare personală în python

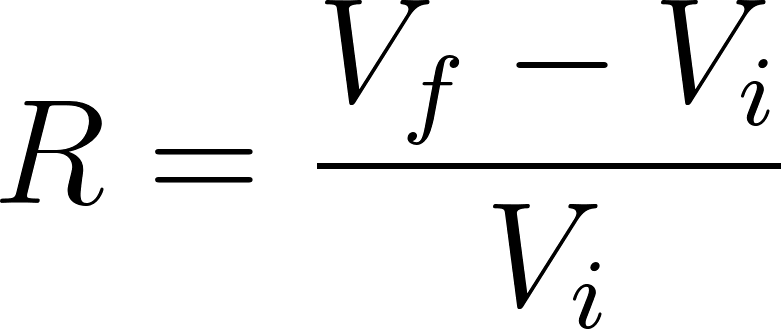
După cum se poate observa în figura 4.27, cu toate că previziunea volatilității nu e chiar perfectă, ea face o treabă destul de bună în a prezice perioadele în care o să aibă loc oscilații ale prețului.

### **Elaborarea unui model de deep learning care se folosește de volatilitatea estimată**

Modelele de deep learning de la studiul de caz anterior aveau o limitare mult prea, find foarte predispuse la overfitting, de data aceasta am procedat diferit, facând modele să ia în considere alte variabile decât prețul istoric.

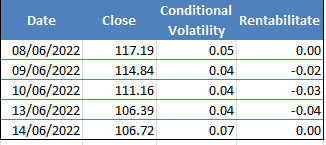
1) Prelucrarea Datelor:

Am început prin a calcula rentabilitățiile după următoare formulă:

[](https://www.codecogs.com/eqnedit.php?latex=R%20%3D%20%5Cfrac%7BV_f%20-%20V_i%7D%7BV_i%7D#0)

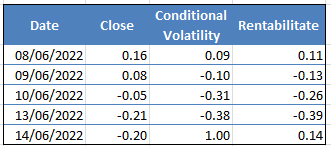
Unde Vf reprezintă valoare finală, iar Vi, cea inițială.

Cea de-a doua variabilă independentă utilizată în prognoza prețului de închidere este volatilitate condiționată estimată de către modelul GARCH(1,1).



Tabelul 4.13: Setul de date realizat pentru antrenarea rețelelor neuronale

Ca și data trecută, am decis să procedez prin normalizarea datelor cu funcția minmax cu valori în intervalul [-1;1] și am împărțit datele în felul următor: primele 80% observații în setul de antrenament și ultimele 20% în setul de testare.



Tabelul 4.14: Setul de date după procedura de normalizare

Odată normalizate, datele sunt convertite în secvențe de câte 10 observații și sunt impărțite în batch-uri de câte 10.

2) Definirea rețelelor neuronale

În acest studiu de caz am sacrificat simplitate pentru a obține o eficiență sporită:

rețeaua are 3 neuroni în stratul de intrare, fiecare corespunde uneia dintre cele 3 caracteristice ale setului de date ( prețul, voltatilitatea condiționată și rentabilitatea ), urmați apoi de două straturi ascunse a câte 120 de neuroni fiecare, în cele din urmă rezultând într-o singură valoare de ieșire.

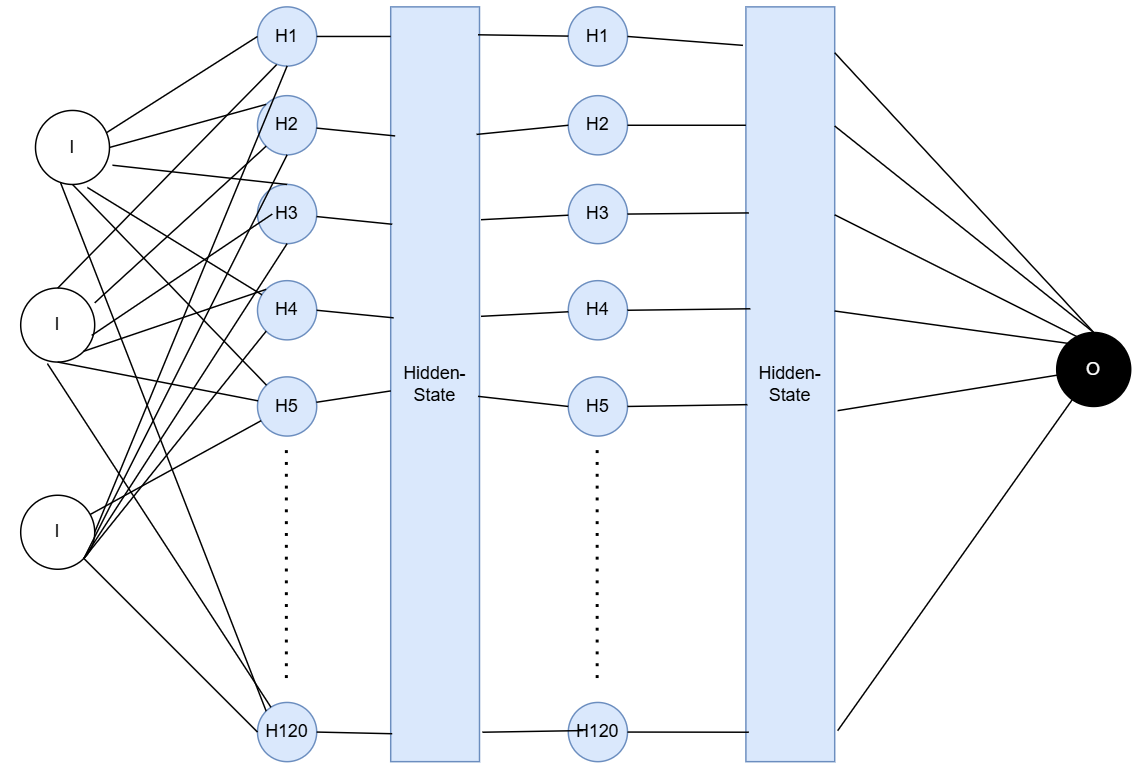
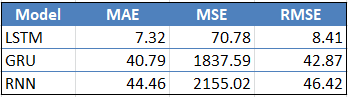


Figura 4.28: Reprezentare grafică a rețelei neuronale concepute pentru garch - prelucrare personală în draw.io

Caracteristicile meta parametrilor:

* Numărul de epoci de antrenament este 10
* Rata de învățare setată este 0.001
* Funcția de pierdere este redată de MSE
* Optimizatorul folosit este ADAM
* Dropout implementat cu o rată de 10%

3) Antrenarea modelului



Tabelul 4.15: Erori generate de fiecare model în previziunea BTC

În tabel se poate observa diferența masivă între LSTM și celelalte modele, cel mai aprent în cazul MSE, unde eroarea modelului GRU este de peste 20 de ori mai mare decât cea a LSTM, iar eroarea modelului RNN de peste 30 de ori mai mare decât cea LSTM.

De asemenea, este aparent și faptul că performanța rețelelor GRU și RNN este relativ similară

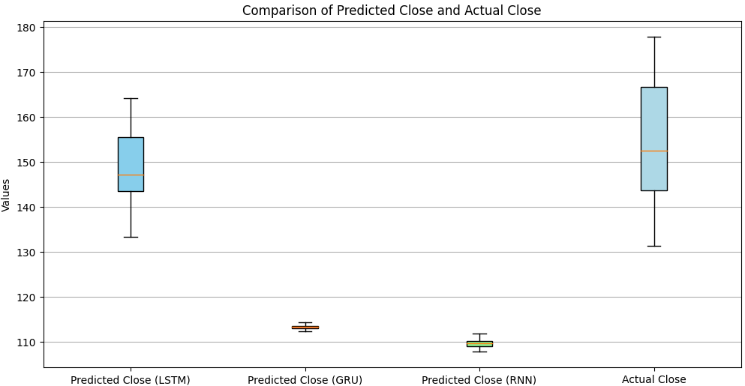


Figura 4.29: Boxploturi ale prețului BTC previzionat - prelucrare personală în python

Diferența masivă a calității previziunilor LSTM fațp de cele GRU și RNN este la fel de aparentă în cadrul figurii de mai sus, GRU și RNN nu reușesc să se încadreze deloc în raza de valori ale prețurile actuale, în schimb LSTM este destul de aproape doar ca în general valorile prezise sunt mai mici decât cele reale.

De asemenea se remarcă că nu există outlieri în cadrul previziunilor.

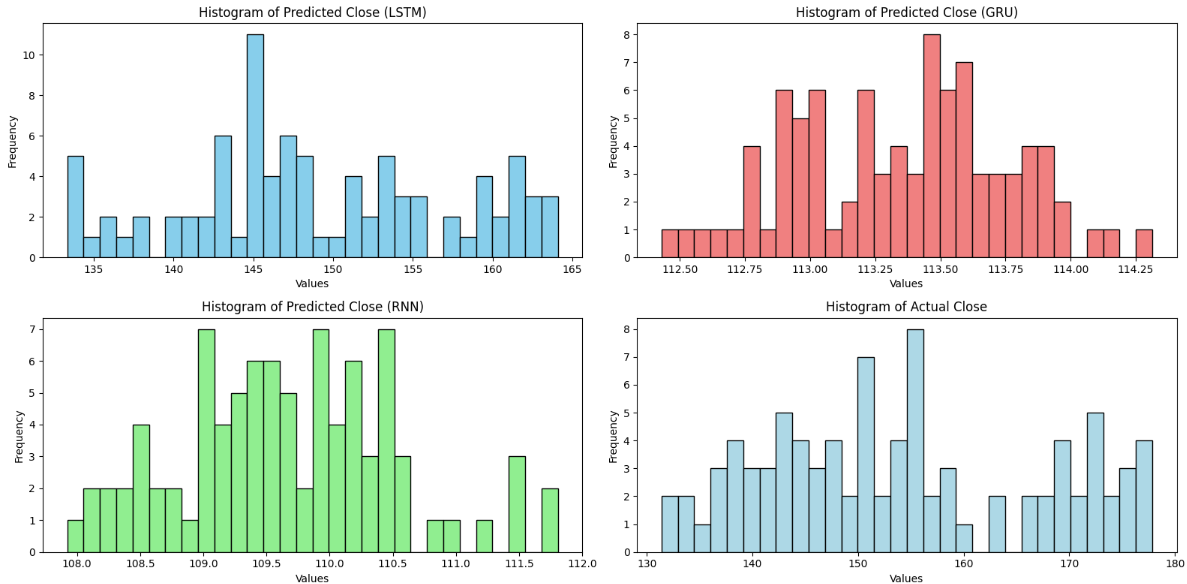


Figura 4.30: Histograme ale prețului BTC previzionat - prelucrare personală în python

Un lucru interesant care reiese din figura de mai sus, este că fiecare set de date are o distribuție relativ distinctă, se pare că cea mai mare greșeală făcută de modelul lstm, este alocarea frecvențelor, această find considerabil diferită față de cea a prețurilor reale.

## **3. Rezultate**

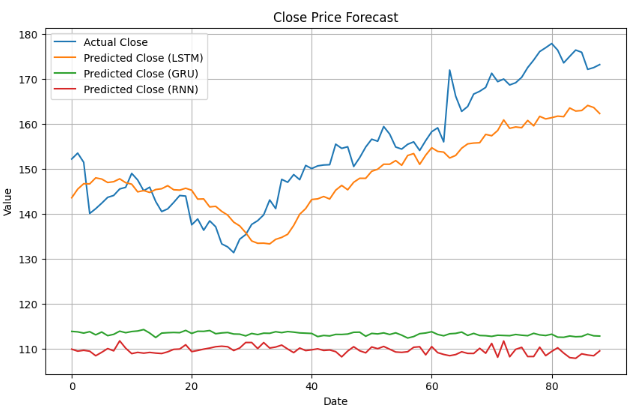


Figura 4.31: Prognoza comparativă prețului BTC cu rețele neuronale - prelucrare personală în python

Pe baza figurii 4.31, se poate vedea că modelul LSTM s-a descurcat remarcabil de bine în a previziona prețul BTC, cu toate că nu a reușit să capteze toate momente când volatilitate prețului a crescut, în schimb modelele GRU și RNN au parte de o performanța foarte slabă, se remarcă anumite tentative de ajustare în funcție de volatilitatea percepută în cazul RNN, însă GRU practic nu a reușit să capteze deloc caracteristicile modelului.

Limitări:

Cu toate că tehniciile de previziune din cadrul acestei extinderi a primului studiu de caz sunt mult mai practice și reziste la overfitting, natura datelor cu caracter financiar rămâne tot la fel de dificil de prezis, fiind necesară abordarea unei metodologii mult mai complexe și luare în considerare a anumitor factori sociali.

Concluzii:

Colaborarea dintr-un model GRU și unul de rețea neuronală se poate dovedi benefică în previziuena datelor cu caracter financiar și care prezintă un nivel înalt de volatilitate.

De asemenea, se remarcă că rețelele de tip LSTM au performanță mai bună decât cele de tip GRU și RNN când vine vorba de previziuni care implică un nivel de complexitate mai ridic