



Academia de Studii Economice, București
Facultatea de Cibernetică, Statistică și Informatică Economică
Specializarea: Informatică Economică

Proiect de practică

***Tema proiectului: Prognoza indicatorilor economici folosind rețele
neuronale LSTM***

Cadrul didactic coordonator: Cristina Rodica Boboc

Student: Charyyeva Yazgul

București

2024

Cuprins

Introducere	4
Metodologia	5
<i>Descrierea tehnicilor si metodelor de analiza folosite.</i>	<i>5</i>
Descrierea Bazei de date.....	7
Aplicarea modelului si rezultate	8
<i>Univariate Stacked LSTM.</i>	<i>8</i>
<i>Multivariate LSTM.....</i>	<i>13</i>
Concluzii	16
Bibliografie	17

Introducere

Stagiul meu de practică a fost susținut la Institutul National de Statistica.

Institutul National de Statistica este principala autoritate de stat in domeniul statisticilor din Romania, avand sediul central in Bucuresti. Rolul sau este de a colecta , analiza si disemina date statistice care reflecta dinamica economica, sociala, demografica si culturala a tarii. Aceste informatii sunt esentiale pentru planificarea si implementarea politicilor guvernamentale, precum si pentru evaluarea progresului social si economic al Romaniei. Institutul este organizat in diverse departamente specializate care se ocupa de diferite sectoare statistice, de la economie la demografie si de la statistici sociale la cele regionale. Aceasta structura permite INS sa ofere date precise si relevante pentru o gama larga de utilizatori, inclusiv guvernul , mediul academic, afacerile si publicul larg.

Am decis sa particip pentru programul de practica la Institutul National de Statistica din dorinta de a imbunatati si aprofunda cunostintele mele in domeniul Statisticilor si Data Science. Participarea in acest cadru, unde sa valorifica datele pentru a informa si modela politici publice, m – a atras in mod special, deoarece sunt pasionat de modul in care analiza statistica poate influenta dezvoltare si optimizarea proceselor in diferite sectoare. De asemenea, posibilitatea de a lucra si invata alaturi de profesionisti experimentati in cadrul INS reprezinta o oportunitate unica de a –mi extinde abilitatile analitice si de a contribui efectiv la proiecte de amploare nationala.

In timp de practica proiectul pe care am creat – o vizeaza dezvoltarea unui model predictiv folosind rețelele neuronale LSTM pentru a prognoza indicatorii economici, in acest caz, Produsul Intern Brut(PIB). Acest model isi propune sa demonstreze cum datele istorice pot fi utilizate pentru a anticipa tendintele economice, oferind astfel o baza solida pentru luarea deciziilor economice si financiare.

In contextul actual, predictibilitatea indicatorilor economici joaca un rol crucial in planificarea si strategia economica la nivel macro si micro. Utilizarea tehnologiilor de invatare automata, precum LSTM, permite analiza si interpretarea complexitatilor dinamice ale datelor economice in timp real.

Metodologia

Descrierea tehnicilor si metodelor de analiza folosite.

Am folosit rețele neuronale LSTM, cunoscute pentru eficiența lor în modelarea seriilor de timp datorită capacității de a păstra informații din trecut pentru o perioadă lungă. Acest lucru este esențial pentru a capta tendințele și ciclicitatea datelor economice. Continuând descrierea modelelor, putem vizualiza compartamentul fiecărui model printr – un grafic al pierdilor pe parcursul antrenării și printr – o diagramă care compară predicțiile modelului cu valorile reale. Modele LSTM utilizate univariate:

1. *Stacked LSTM* : A fost utilizat pentru a îmbunătăți acuratețea modelului prin adăugarea mai multor straturi LSTM una peste alta. Acest lucru permite modelului să învețe o gamă mai largă de abstracții ale datelor. Stacked LSTM este deosebit de util pentru modelarea relațiilor complexe și a interdependentelor din serie. Formula generală a modelului Stacked LSTM:

$$h_t = f(W \cdot [h_{t-1}, x_t] + b)$$

unde h_t este starea ascunsă la timpul t , x_t este intrarea la timpul t , W și b sunt parametrii modelului, și f este funcția de activare.

2. *Vanilla LSTM*: Acest model simplu a servit ca punct de plecare, oferind o bază de comparație pentru celelalte modele mai complexe. Chiar și în forma sa cea mai simplă, Vanilla LSTM poate capta secvențialitatea datelor și poate produce predicții rezonabile. Am folosit Vanilla LSTM pentru a stabili o linie de bază a performanței, verificând cât de bine poate modelul să prezică PIB – ul fără ajustări suplimentare sau optimizări complexe.
3. *Bidirectional LSTM*. Formula:

$$\begin{aligned} h_t &= f(W \cdot [h_{t-1}, x_t] + b) \\ h_t \leftarrow &= f \leftarrow (W \cdot [h_{t+1}, x_t] + b) \quad h_t = f(W \cdot [h_{t+1}, x_t] + b) \\ y_t &= g(W \cdot [h_t \rightarrow, h_t \leftarrow] + b) \quad y_t = g(W \cdot [h_t, h_t] + b) \end{aligned}$$

Prin procesarea datelor atât în direcția trecut – către – viitor, cât și invers, acest model poate înțelege contextul într – un mod mai complet, ceea ce îl face ideal pentru serii de timp care pot beneficia de la interpretări din ambele perspective temporale.

4. *CNN LSTM*: Integrarea straturilor convolutive înainte blocurilor LSTM permite acestui model să extragă și să interpreteze de timp la nivel mai înalt. Este o abordare eficientă când datele conțin pattern-uri temporale subtile care trebuie extrase. Utilizarea CNN LSTM a facilitat modelarea detaliilor fine ale fluctuațiilor economice sezoniere captând caracteristici importante ale datelor care ar putea fi omise de un model LSTM simplu.
5. *ConvLSTM*: Acest model este ideal pentru date care au atât componente temporale cât și spațiale, oferind o capacitate unică de a procesa informații pe mai multe axe. ConvLSTM a fost explorat ca o metodă de a integra informații economice suplimentare, cum ar fi distribuția geografică a activității economice, pentru a vedea cum acestea influențează PIB-ul național

Modele LSTM utilizate multivariate:

1. *Multiple input series LSTM* : Formula de configurarea a intrărilor multiple:

$$\begin{aligned} X_t &= [x_{1t}, x_{2t}, \dots, x_{nt}] \\ h_t &= \text{LSTM}(X_t) \\ y_t &= \sigma(W \cdot h_t + b) \end{aligned}$$

unde X_t reprezintă vectorul de intrări la timpul t .

2. *Random Forest Regressor* este o tehnică robustă de învățare automată care folosește multiple arbori de decizie pentru a realiza predicții, având capacitatea de a gestiona eficient serii de timp multivariate. Acest model este adesea preferat în scenarii economice complexe deoarece oferă o interpretare clară a importanței variabilelor și este capabil să capteze relații non-liniare și interacțiuni între caracteristici fără necesitatea unei configurații laborioase. Prin adăugarea acestor detalii, secțiunea despre modelele pentru serii de timp multivariate devine mai completă și mai informativă, subliniind capacitatea Random Forest de a complementa modelele LSTM în analiza datelor economice complexe.

Descrierea Bazei de date

Datele pentru a realiza analizele economice propuse in acest proiect, am utilizat o baza de date colectata de la TEMPO, Institutul National de Statistica, care include indicatori economici relevanti. Aceasta contine seriile de timp ale Produsului Intern Brut(PIB), un indicator esential pentru masurarea performantei economice a unei tari. Baze de date cuprinde observatii trimestriale ale PIB – ului , inregistrate pe parcursul ultimilor 5 ani. Fiecare inregistrare corespunde unui trimestru specific, reflectand astfel dinamica economica intr – un interval bine definit. Datele sunt structurate intr – un format tabelar cu aproximativ 40 de linii reprezentand trimestrele , si 5 coloane. Variabile pentru setul de date :

1. *Anul* : Aceasta indica anul in care datele au fost inregistrate si este crucial pentru analiza trendurilor pe termen lung.
2. *Trimestrul* : Aceasta variabila reprezinta anul in patru parti egale, oferind o granularitate mai fina pentru observatii si analiza.
3. *PIB*: Valoarea Produsului Intern Brut, exprimata in milioane de unitati monetare locale, este variabila dependenta in modelul nostru. Aceasta reprezinta suma tuturor bunurilor si serviciilor produse intr – o economie si este ajustata sezonier pentru a reflecta variatiile economice naturale de – a lungul anului.
4. *Rata de angajare* : Variabila reprezinta procentul persoanelor angajate din populatia activa disponibil si este un indicator vital al sanatatii economice. Rata de angajare influenteaza consumul si, implicit , cresterea economica, deoarece un numar mai mare de persoane angajate inseamna o capacitate crescuta de cheltuieli si investitie.
5. *Cheltuielile totale*: Acestea include suma totala a cheltuielilor de consum si a investitiilor realizate in economie intr – o perioada determinata. Cheltuielile totale sunt direct corelate cu cresterea economica, deoarece reflecta activitatea economica generala si capacitatea de consum a populatiei.

Inainte de a fi folosite in modelare , datele au fost supuse unor procese de prelucrare si curatare. Datele legate de PIB au fost normalizate folosind MinMaxScaler pentru a facilita procesarea de catre modelele de invatare automata. Acest metodul se foloseste pentru a asigura ca modelul nu va fi partinitor catre valori anormal de mari sau mici. Similar cu PIB – ul , variabilele

Rata de Angajare si Cheltuielile Totale au fost normalizate pentru a asigura comparabilitatea si pentru a evita bias – urile in modelare datorate scalarii diferite a valorilor.

Aplicarea modelului si rezultate

Dupa ce am incarcat setul de date , am normalizat datele intre 0 si 1 . Aceasta este o practica standard in prelucrarea seriilor de timp pentru rețele neuronale, deoarece ajuta la accelerarea convergentei in timpul antrenamentului. Datele sunt impartite in seturi de antrenament si de testare, cu 80% din date alocate pentru antrenament si restul de 20% pentru testare. Aceasta permite evaluarea modelului pe date nevazute pentru a verifica performanta generala.

```
def create_dataset(dataset, time_steps=1):  
    X, Y = [], []  
    for i in range(len(dataset)-time_steps-1):  
        a = dataset[i:(i+time_steps), 0]  
        X.append(a)  
        Y.append(dataset[i + time_steps, 0])  
    return np.array(X), np.array(Y)
```

Figure 1

Aceasta functie “*create_dataset*” este utilizata pentru a restructura datele intr – un format adecvat pentru antrenarea rețelei LSTM . Ea creeaza secvente de intrare “*X*” si etichete “*Y*” pe baza unui numar de pasi temporal(“*time_steps*”). Dupa am redimensionat datele pentru a se potrivi cerintelor de intrare ale modelului LSTM , care necesita datele de intrare sa fie formatul(*samples, time steps, features*). Apoi am construit modelul univariate :

Univariate Stacked LSTM.

```
model = Sequential()  
model.add(LSTM(units=50, return_sequences=True, input_shape=(1, time_steps)))  
model.add(LSTM(units=50))  
model.add(Dense(units=1))
```

Figure 2

Apoi am compilat modelul cu optimizatorul “*adam*” si functia de pierdere “*mean_squared_error*”, care sunt adecvate pentru probleme de regresie. Modelul este antrenat

pentru 100 de epochi, iar rezultatele arata pierderea la fiecare epoca, indicand cum modelul invata si se adapteze la date.

```
Epoch 92/100
14/14 - 0s - loss: 0.0111 - 57ms/epoch - 4ms/step
Epoch 93/100
14/14 - 0s - loss: 0.0109 - 112ms/epoch - 8ms/step
Epoch 94/100
14/14 - 0s - loss: 0.0110 - 84ms/epoch - 6ms/step
Epoch 95/100
14/14 - 0s - loss: 0.0109 - 62ms/epoch - 4ms/step
Epoch 96/100
14/14 - 0s - loss: 0.0115 - 51ms/epoch - 4ms/step
Epoch 97/100
14/14 - 0s - loss: 0.0113 - 52ms/epoch - 4ms/step
Epoch 98/100
14/14 - 0s - loss: 0.0110 - 50ms/epoch - 4ms/step
Epoch 99/100
14/14 - 0s - loss: 0.0112 - 51ms/epoch - 4ms/step
Epoch 100/100
14/14 - 0s - loss: 0.0121 - 55ms/epoch - 4ms/step
```

Figure 3

Mai jos avem si predictia pentru urmatorul 4 ani:

```
[[2.023000e+03 4.000000e+00 4.159069e+05]
 [2.027000e+03 7.000000e+00 5.917351e+05]]
```

Figure 4

Vanilla LSTM. Aici datele sunt redimensionate pentru a se potrivi cu cerintele modelului LSTM, care necesita inputeri de form[*samples, time_steps, features*]. Modelul Vanilla LSTM este definit cu un singur strat LSTM urmat de un strat dens pentru predictia output – ului scalar.

```
def create_sequences(data, sequence_length):
    x, y = [], []
    for i in range(len(data) - sequence_length):
        x.append(data[i:i+sequence_length])
        y.append(data[i+sequence_length])
    return np.array(x), np.array(y)

sequence_length = 3
X, y = create_sequences(data['PIB'].values, sequence_length)
```

Figure 5

Modelul este antrenat folosind setul de date , pe durata a 100 epoci, cu un batch size de 1.

```
17/17 - 0s - loss: 0.0063 - 60ms/epoch - 4ms/step
Epoch 92/100
17/17 - 0s - loss: 0.0065 - 60ms/epoch - 4ms/step
Epoch 93/100
17/17 - 0s - loss: 0.0062 - 89ms/epoch - 5ms/step
Epoch 94/100
17/17 - 0s - loss: 0.0066 - 87ms/epoch - 5ms/step
Epoch 95/100
17/17 - 0s - loss: 0.0066 - 83ms/epoch - 5ms/step
Epoch 96/100
17/17 - 0s - loss: 0.0064 - 71ms/epoch - 4ms/step
Epoch 97/100
17/17 - 0s - loss: 0.0064 - 52ms/epoch - 3ms/step
Epoch 98/100
17/17 - 0s - loss: 0.0063 - 84ms/epoch - 5ms/step
Epoch 99/100
17/17 - 0s - loss: 0.0062 - 60ms/epoch - 4ms/step
Epoch 100/100
17/17 - 0s - loss: 0.0066 - 58ms/epoch - 3ms/step
```

Figure 6

Apoi am calculat eroare pătratică medie, care este $RMSE = 13751.719326895127$ sugerează că, în medie, predicțiile modelului deviază cu aproximativ 13751.719 unități de valorile reale.

Bidirectional LSTM. După ce am definit modelul Bidirectional LSTM am compilat și am antrenat la fel ca alte tipuri de LSTM.

```

Epoch 92/100
1/1 [=====] - 0s 19ms/step - loss: 0.0079
Epoch 93/100
1/1 [=====] - 0s 14ms/step - loss: 0.0079
Epoch 94/100
1/1 [=====] - 0s 16ms/step - loss: 0.0079
Epoch 95/100
1/1 [=====] - 0s 18ms/step - loss: 0.0078
Epoch 96/100
1/1 [=====] - 0s 18ms/step - loss: 0.0078
Epoch 97/100
1/1 [=====] - 0s 15ms/step - loss: 0.0079
Epoch 98/100
1/1 [=====] - 0s 17ms/step - loss: 0.0079
Epoch 99/100
1/1 [=====] - 0s 16ms/step - loss: 0.0079
Epoch 100/100
1/1 [=====] - 0s 17ms/step - loss: 0.0079

```

Figure 7

Aceasta configuratie utilizeaza straturi LSTM in ambele directii, procesand infomatiile din trecut spre viitor si din viitor spre trecut simultan. După antrenarea modelului, este util să vizualizăm pierderea (*loss*) pe parcursul epocilor pentru a evalua progresul și performanța modelului:

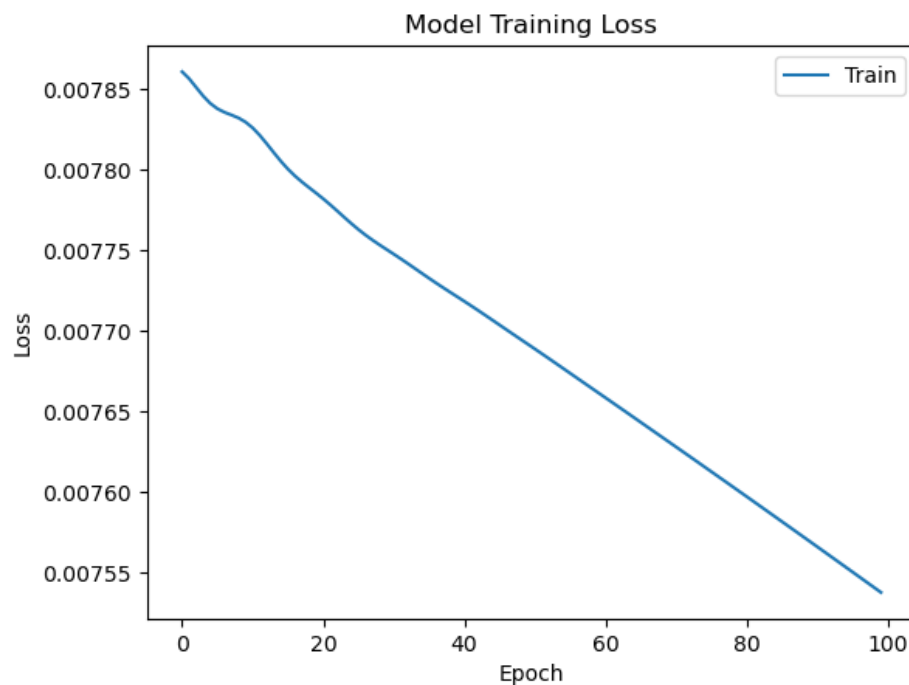


Figure 8

Putem folosi ultimele valori ale seriei de timp pentru a face predicții. Adică, 391468.3, 402454.9, 415906.9.

```
yhat = model.predict(x_input, verbose=0)

print(yhat)

[[4.650444]]
```

Figure 9

Rezultatul predicției, afișat de *print(yhat)*, este *[[4.650444]]*. Aceasta este predicția modelului pentru următoarea valoare a PIB-ului, bazată pe cele trei valori anterioare furnizate.

CNN LSTM

Modelul folosește o arhitectură *TimeDistributed* pentru a aplica convoluții pe fiecare segment temporal al seriei de date, urmat de straturi de pooling și flattening pentru a reduce dimensiunea datelor și a extrage caracteristici relevante.

Apoi, datele procesate sunt trimise printr-un strat LSTM pentru a capta dependente temporale.

```
1/1 [=====] - 0s 14ms/step - loss: 0.0111
Epoch 93/100
1/1 [=====] - 0s 14ms/step - loss: 0.0111
Epoch 94/100
1/1 [=====] - 0s 14ms/step - loss: 0.0110
Epoch 95/100
1/1 [=====] - 0s 13ms/step - loss: 0.0109
Epoch 96/100
1/1 [=====] - 0s 14ms/step - loss: 0.0108
Epoch 97/100
1/1 [=====] - 0s 13ms/step - loss: 0.0107
Epoch 98/100
1/1 [=====] - 0s 13ms/step - loss: 0.0106
Epoch 99/100
1/1 [=====] - 0s 13ms/step - loss: 0.0105
Epoch 100/100
1/1 [=====] - 0s 12ms/step - loss: 0.0104
```

Figure 10

Antrenamentul a fost efectuat pe 100 de epoci, cu pierderea (*loss*) scăzând progresiv de la 0.3233 la 0.0104. Aceasta indică faptul că modelul a învățat eficient din date pe următorul antrenament, adaptându-și ponderile pentru a minimiza eroarea de predicție. Predicția modelului pentru următoarea valoare în seria aceasta 1.090356.

ConvLSTM


Antrenamentul se desfășoară pe 100 de epoci, cu pierderi (*loss*) care scad progresiv de la 0.3397 la 0.0177. Acest lucru indică faptul că modelul învață eficient și se adaptează pentru a minimiza eroarea de predicție.

```
1/1 [=====] - 0s 20ms/step - loss: 0.0192
Epoch 93/100
1/1 [=====] - 0s 21ms/step - loss: 0.0190
Epoch 94/100
1/1 [=====] - 0s 19ms/step - loss: 0.0188
Epoch 95/100
1/1 [=====] - 0s 20ms/step - loss: 0.0185
Epoch 96/100
1/1 [=====] - 0s 19ms/step - loss: 0.0183
Epoch 97/100
1/1 [=====] - 0s 17ms/step - loss: 0.0182
Epoch 98/100
1/1 [=====] - 0s 24ms/step - loss: 0.0180
Epoch 99/100
1/1 [=====] - 0s 22ms/step - loss: 0.0178
Epoch 100/100
1/1 [=====] - 0s 19ms/step - loss: 0.0177
```

Figure 11

La finalul antrenamentului o predicție pentru următoarea valoare a PIB – ului.

```
yhat_original = scaler.inverse_transform(yhat)
print("Predicted GDP value:", yhat_original)
```



```
Predicted GDP value: [[0.99466413]]
```

Figure 12

Multivariate LSTM

Pentru analiza multivariate am adaugat variabile “Rata angajare” si “Cheltuiile totale” apoi am facut Multiple Input Series si Random Forest Regressor. Dupa ce am incarcat datele actualizate, cream secvente “*in_seq1*”, “*in_seq2*”, “*out_seq*” pentru modelul nostru. Fiecare

dintre aceste secvențe este redimensionată într-un format de (numărul_de_înregistrări, 1) pentru a le pregăti pentru a fi stivuite orizontal. Acest format este necesar pentru unele modele de machine learning care necesită o dimensiune specifică a intrării.

```
def split_sequences(sequences, n_steps):
    x, y = list(), list()
    for i in range(len(sequences)):
        # Find the end of this pattern
        end_ix = i + n_steps
        # Check if we are beyond the dataset
        if end_ix > len(sequences):
            break
        # Gather input and output parts of the pattern
        seq_x, seq_y = sequences[i:end_ix, :-1], sequences[end_ix-1, -1]
        x.append(seq_x)
        y.append(seq_y)
    return array(x), array(y)

x, y = split_sequences(dataset, n_steps)
print(x.shape, y.shape)

(18, 3, 2) (18,)
```

Figure 13

split_sequences este o funcție care ia un dataset complet și creează perechi de intrare-țintă pe baza unui număr specificat de pași (*n_steps*). Fiecare secvență de intrare este formată din *n_steps* rânduri consecutive din dataset, minus ultima coloană care este considerată variabila țintă (PIB-ul în acest caz).

```

[7.608656e+06 3.306100e+03]] 346335.5
[[7.439170e+06 3.267710e+03]
[7.608656e+06 3.306100e+03]
[7.552003e+06 3.562130e+03]] 355617.5
[[7.608656e+06 3.306100e+03]
[7.552003e+06 3.562130e+03]
[7.493612e+06 3.662180e+03]] 363828.5
[[7.552003e+06 3.562130e+03]
[7.493612e+06 3.662180e+03]
[7.378396e+06 3.702120e+03]] 384075.6
[[7.493612e+06 3.662180e+03]
[7.378396e+06 3.702120e+03]
[7.417382e+06 3.675080e+03]] 391468.3
[[7.378396e+06 3.702120e+03]
[7.417382e+06 3.675080e+03]
[7.455072e+06 4.024730e+03]] 402454.9
[[7.417382e+06 3.675080e+03]
[7.455072e+06 4.024730e+03]
[7.466183e+06 4.135840e+03]] 415906.9

```

Figure 14

[[7.892464e+06 2.347000e+03] [8.142162e+06 2.413050e+03] [8.154113e+06 2.570570e+03]] 267966.3 arată o secvență de trei pași de timp unde modelul ia valorile “*Rata angajare*” și “*Cheltuielile_total*” ca input și ‘267966.3’ ca output, adică valoarea ‘PIB’ care trebuie prezisă. Această structură este utilă pentru modelele de timp, precum rețelele neuronale LSTM, care învață să prezică valori viitoare din serii de timp bazate pe secvențe de date istorice.

Apoi am aplicat modelul Random Forest Regressor, am calculat,

1. Eroare absolută medie: 7580.220999999947.
2. Eroare pătratică medie rădăcină: 8953.360579273027.

Un MAE de 7580.22 înseamnă că, în medie, predicțiile modelului deviază de la valorile reale ale PIB-ului cu aproximativ 7580.22 unități.

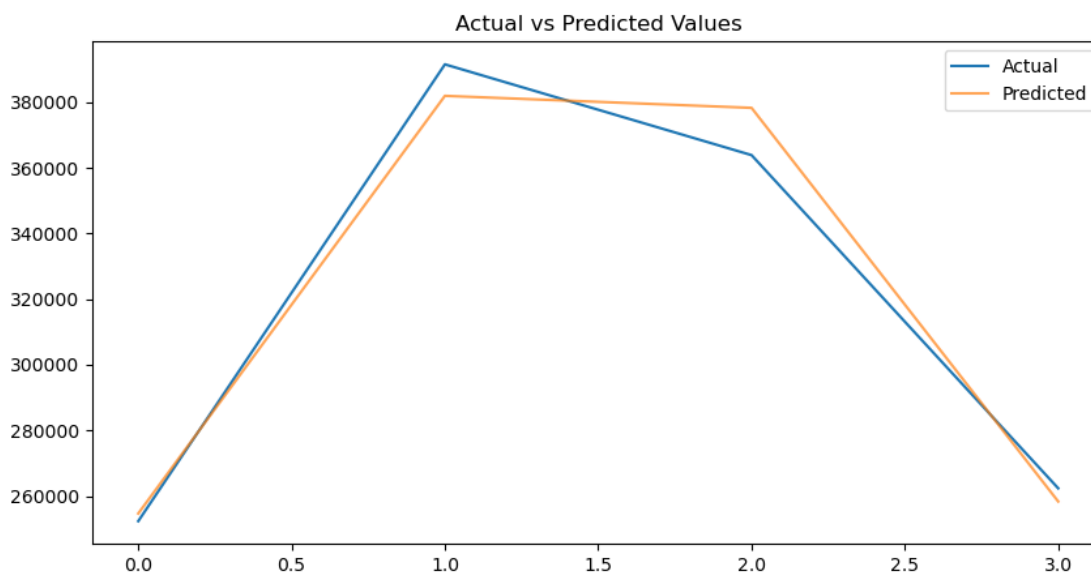


Figure 15

Linia *Predicted* urmărește destul de fidel linia *Actual*, ceea ce indică că modelul a reușit să captureze tendințele generale ale datelor PIB-ului destul de bine.

Punctele de Divergență: Există o suprapunere aproape perfectă în prima jumătate a graficului, dar în a doua jumătate, în special după punctul 1.5 pe axa x, se observă o discrepanță mai mare între cele două linii. Aceasta ar putea indica faptul că modelul are dificultăți în a modela anumite condiții sau schimbări din date care apar în acea parte a setului de date.

Concluzii

Proiectul a fost realizat în cadrul Institutului Național de Statistică, utilizând datele colectate de INS. Această colaborare a permis accesul la seturi de date actualizate și relevante, ceea ce a îmbunătățit calitatea și relevanța predicțiilor modelului. Experiența la INS a fost esențială pentru aplicarea practică a modelelor LSTM în contextul real al datelor economice naționale. Studiul a confirmat că modelele LSTM sunt extrem de eficiente în modelarea și predicția seriilor de timp, captând complexitățile și dinamica datelor economice pe termen lung furnizate de INS. Prin testarea diverselor configurații ale rețelelor LSTM, inclusiv Vanilla, Stacked, Bidirectional, CNN și ConvLSTM, proiectul a explorat cum diferitele arhitecturi se pot adapta specific la caracteristicile datelor INS, optimizând astfel predicțiile. Integrarea modelului Random Forest ca metodă

complementară a demonstrat utilitatea combinației între învățarea automată tradițională și cea profundă, îmbunătățind acuratețea predicțiilor economice pe baza datelor .

Bibliografie

1. INS <http://statistici.insse.ro:8077/tempo-online/>
2. Long Short-Term Memory Networks With Python Develop Sequence Prediction Models With Deep Learning by Jason Brownlee
3. <https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/>
4. "Deep Learning" Ian Goodfellow, Yoshua Bengio and Aaron Courville