Proiect Procesarea Semnalelor
Detecția anomaliilor în serii de timp

Petre-Șoldan Adela, Cătălina Nica, Alexandru Miclea
(adela.petre-soldan, maria-catalina.nica,
alexandru.miclea)@s.unibuc.ro

Facultatea de
Matematică şi Informatică
Universitatea din Bucureşti

UNIVERSITY OF
BUCHAREST
VIRTUTE ET SAPIENTIA

Facultatea de Matematică și Informatică
Universitatea din București

16.01.2025

Cuprins

Problemă

Detectia anomaliilor pe serii de timp reprezinta identificarea valorilor neobisnuite intr o secventa de date temporale. Aceste anomalii pot semnala evenimente critice, probleme tehnice sau schimbari neasteptate in comportamentul unui sistem.

În cadrul acestui proiect am folosit dataset-ul NVidia - Stock Data - Latest and Updated pentru a ne desfășura experimentele.

Justificare

Detectarea acestor anomalii poate fi critica in contextul prevenirii evenimentelor neplacute (deteriorari ale sistemelor, atacuri cibernetice, etc). De asemenea in industrie, finante sau domeniul sanatatii intarzierile in identificarea anomaliilor pot duce la pierderi financiare semnificative, avarii sau pot risca sanatatea oamenilor. Prin urmare dezvoltarea unui sistem automatizat de detectarea a erorilor care sa poata face fata volumului mare de date este indispensabil.

Biblioteci folosite

- **PyTorch & CUDA & Tensorflow** - Backend pentru modelele de AI
- **Keras** - API pentru dezvoltare
- **Matplotlib** - Prezentare de grafice cu observațiile făcute
- **NumPy** - Bibliotecă pentru prelucrat colecții de date matematic
- **Pandas** - Librarie de manipulat date în format CSV
- **Scikit-Learn** - Funcții de preprocesare a datelor (e.g. StandardScaler)

Modele folosite

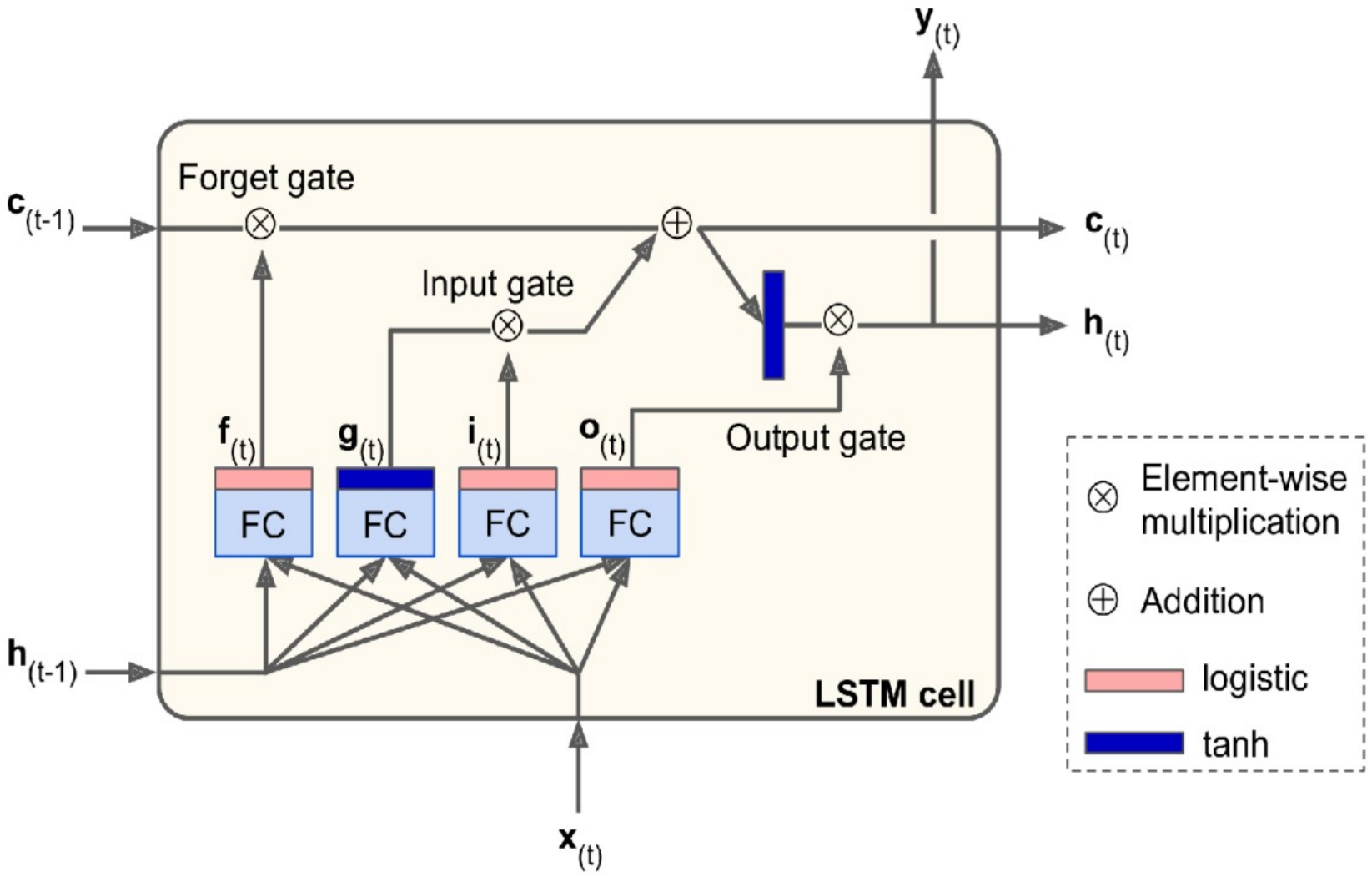
Abordarea noastră a constat în implementarea a patru modele de Machine Learning care să găsească puncte aferente unor anomalii:

- LSTM
- Autoencoders
- Isolation Forests
- Transformers

Long Short-Term Memory

Am utilizat o rețea LSTM, deoarece aceasta poate modela dependențele pe termen lung și tiparele complexe din seriile de timp. Modelul a fost antrenat pe date din trecut pentru a realiza predicții în viitor. Am calculat eroarea (loss-ul) dintre predicții și valorile reale, iar dacă aceasta depășea un prag predefinit, punctul respectiv era considerat anomalie.

Long Short-Term Memory



Long Short-Term Memory

$C_{(t-1)}$ reprezintă starea pe termen lung (*long-term state*), $h_{(t-1)}$ starea pe termen scurt (*short-term state*), iar $x_{(t)}$ inputul curent. Poarta de uitare ($f(t)$) decide ce informații din trecut sunt eliminate, contribuind la actualizarea memoriei. Funcția $g(t)$ analizează inputul curent împreună cu starea pe termen scurt anterioară, iar poarta de intrare controlează cât din acest input este adăugat la memoria pe termen lung. Poarta de ieșire ($o(t)$) determină ce parte din memoria pe termen lung, acum actualizată, contribuie atât la outputul curent, cât și la actualizarea stării pe termen scurt.

Long Short-Term Memory - Descriere Arhitectura

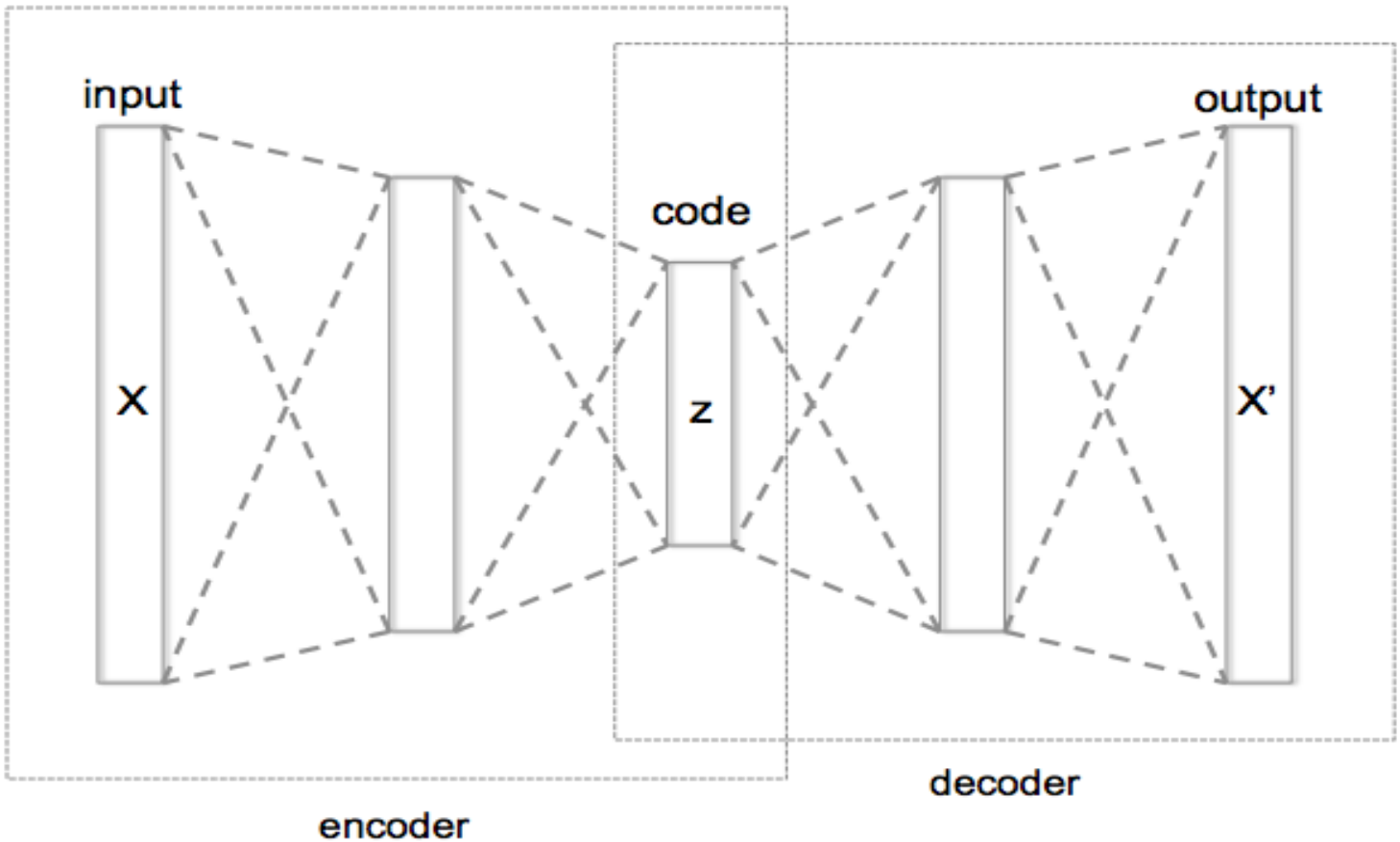
Modelul creat are stratul de input alcatuit din 64 celule LSTM, iar stratul de output este fully connected de dimensiune 1. Modelul contine un strat LSTM ascuns (hidden layer) de dimensiune 32. Pentru functia de pierdere am folosit MSE, iar pentru functiile de activare tanh:

$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Autoencoders

Autoencoderele reprezintă o tehnică de învățare nesupervizată. Rolul acestora este de a deduce o reprezentare mai compactă sau mai dispersată a datelor de intrare. Autoencoderele sunt formate din 2 componente: Encoder și Decoder. Modul prin care știm că autoencoderele învață este prin minimizarea costului de reconstrucție (ex: Squared Error Loss: $L(x, \hat{x}) = \|x - \hat{x}\|^2$). Eroarea de reconstrucție reprezintă eroarea între multimea de antrenare și cea de validare. În cazul Autoencoderelor, mulțimea de testare este aceeași cu cea de validare. Există multe arhitecturi de Autoencodere în literatură. Cea încercată în acest proiect este Convolutional Autoencoder.

Autoencoders



[1]

[1] https://en.wikipedia.org/wiki/Autoencoder

Autoencoders - Rundown al implementării

- Dataset - normalizat și spart în ferestre de 12 saptamâni full consecutive de trading (de ex. pe test din (166, 1) ma duc in (27, 60))
- Antrenez Autoencoderul
- Fac diferenta in modul intre x si $\hat{x}$ (pe datele normalizate)
- Clasific punctele ca fiind anomalii dacă diferența în modul se află la o distribuție standard de medie

Isolation Forests

Isolation Forest este folosit pentru identificarea tranzacțiilor bancare sau a operațiunilor cu carduri de credit suspecte. Detectarea comportamentelor financiare atipice, cum ar fi retrageri masive sau plăți neobișnuite. Este folosit si in Cybersecurity. Isolation forest este un algortim de invatare nesupervizată, avand ca scop principal detectia anomaliilor. Se bazeaza pe principiul izolarii, punctele care sunt anomalii fiind in general izolate mai usor, prin mai putine operatii. Punctele normale fac parte din aglomerari dense, au valori care nu ies din tipare. Foloseste partitionarea binara si recursiva pentru a crea o padure de arbori.

Isolation Forests

Crearea arborilor se face recursiv. La fiecare pas alegem aleator o caracteristica a seriei de timp si o valoare tot aleatoare a acesteia. Esantioanele vor fi impartite in 2 subsecvente. Cele care au valoarea caracteristicii mai mica sau egala decat valoarea aleasa vor deveni fiul stang, iar celelalte fiul drept. Un nod din arbori retine caracteristica aleasa, valoarea sa si cei 2 subarbori. Exista o valoare maxima pentru adancimea unui arbore, pentru a nu izola variatiile minore si pentru a preveni suprainvatarea detaliilor nesemnificative, cum ar fi zgomotul. Limitarea adancimii permite modelului sa se concentreze pe tipare generale si nu pe detalii specifice. Pentru o subsecventa de dimensiune n, adancimea optima este aproximativ $log_2(n)$.

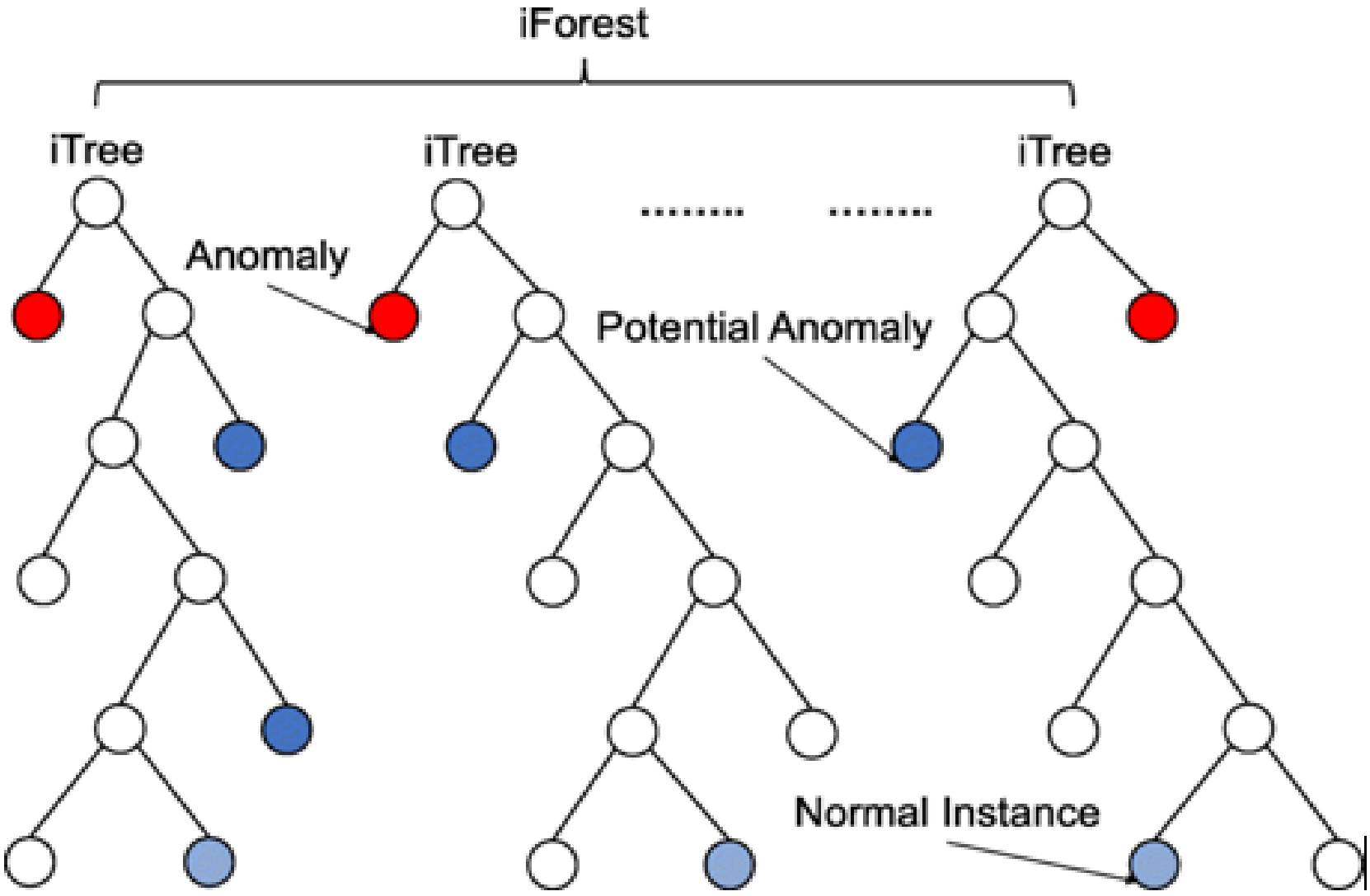
Isolation Forests

Pentru a stabili care esantioane reprezinta anomalii, avem nevoie sa calculam ceoficientul de anomalie, care este legat de cat de usor a fost de izolat esantionul in toti arborii din care face parte:
$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$

$E(h(x))$ = media adancimii la care se afla in fiecare arbore $C(n)$ = constanta de normalizare; Aceasta are mai multe formule echivalente sau aproximative, toate fiind legate de lungimea asteptata a caii pentru un punct izolat intr un arbore binar cu N puncte. Scopul constantei este de a normaliza $E(h(x))$. Formula clasica a constantei: $c(n) = 2H(n-1) - n^2(n-1)$

$H(n–1)$ = suma armonica = $ln(N–1) + 0.577215$

Isolation Forests

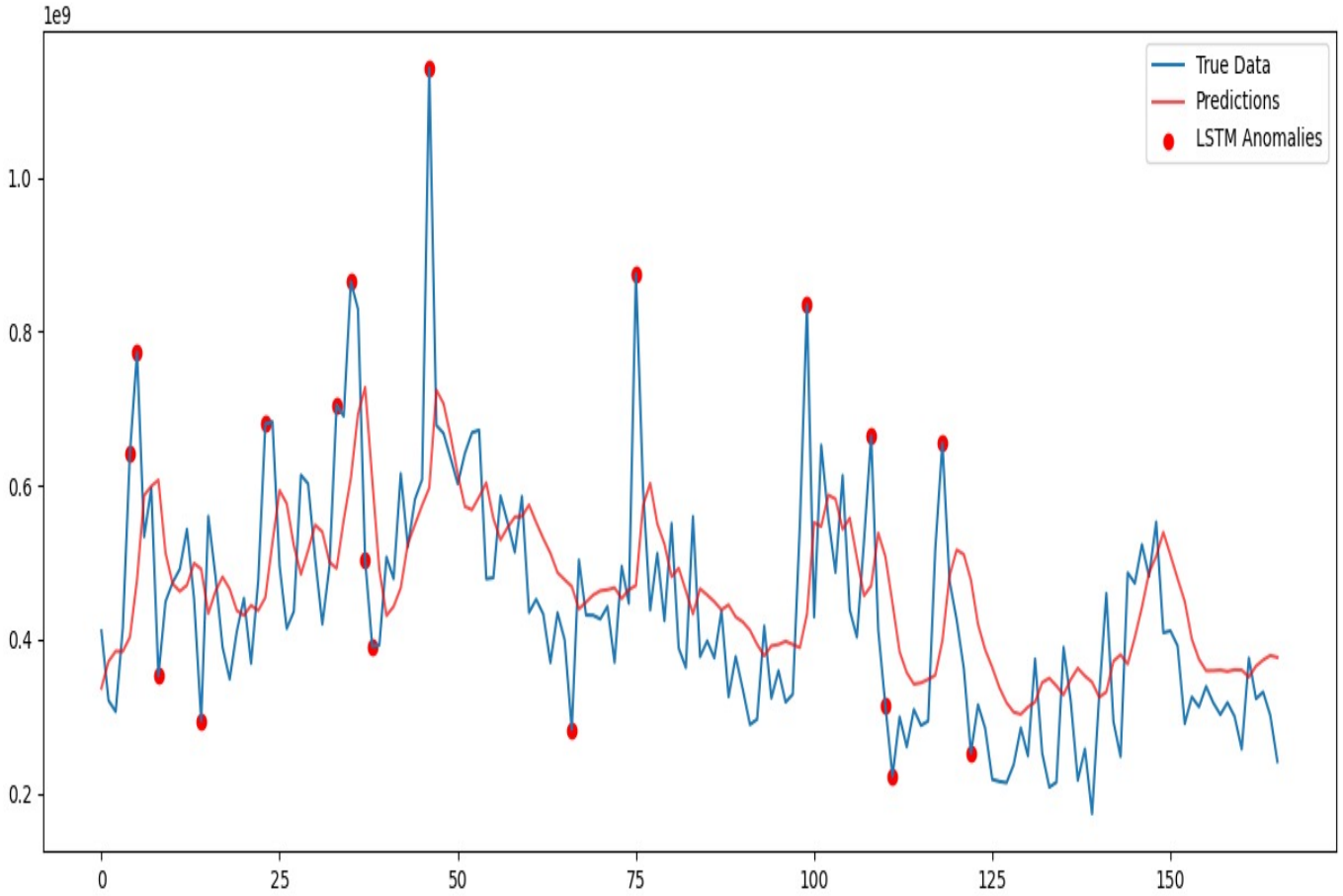


Transformers

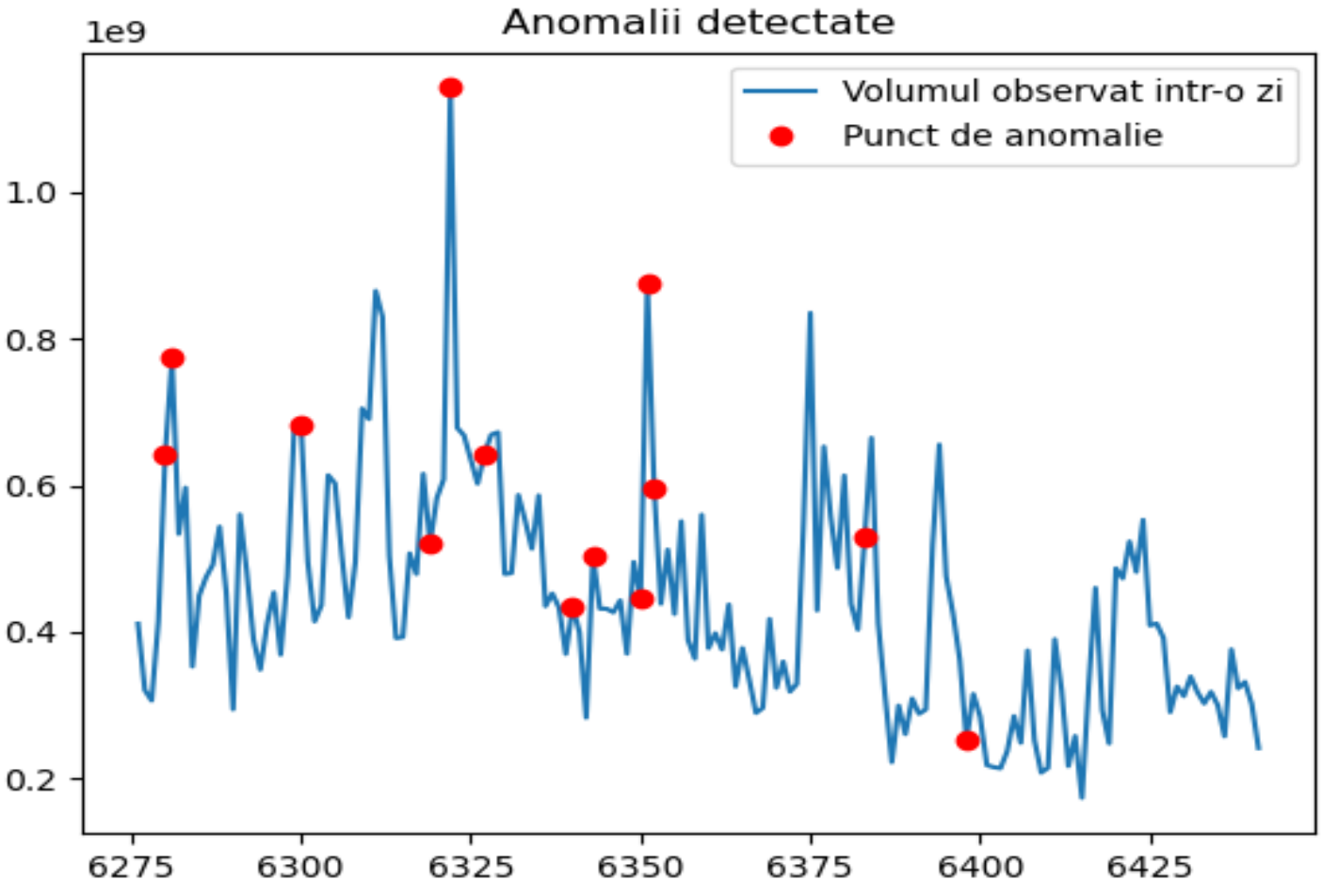
Modelul transformer este folosit in procesarea secventiala a datelor, cum ar fi limbajul natural, traducerea automata si generarea de text. Un transformer este o retea neuronala cu urmatoarele componente de baza:

- **Self Attention**: permite esantioanelor sa interactioneze unul cu celalalt si sa obtina informatii de la fiecare daca are nevoie
- **Positional Encoding**: Operatiile unui transformer sunt realizate in mod paralel, nu secvential, deci e nevoie de o dimensiune noua pentru a retine pozitiile fiecarui element.
- **Feed Forward**: Este o retea neuronala cu un singur sens folosit dupa fiecare apel Self Attention, cu rol de a aprofunda si mai mult informatia.
- **Encoder**: Contine layer de self_attention, urmat de normalizare si layer de feed_forward.

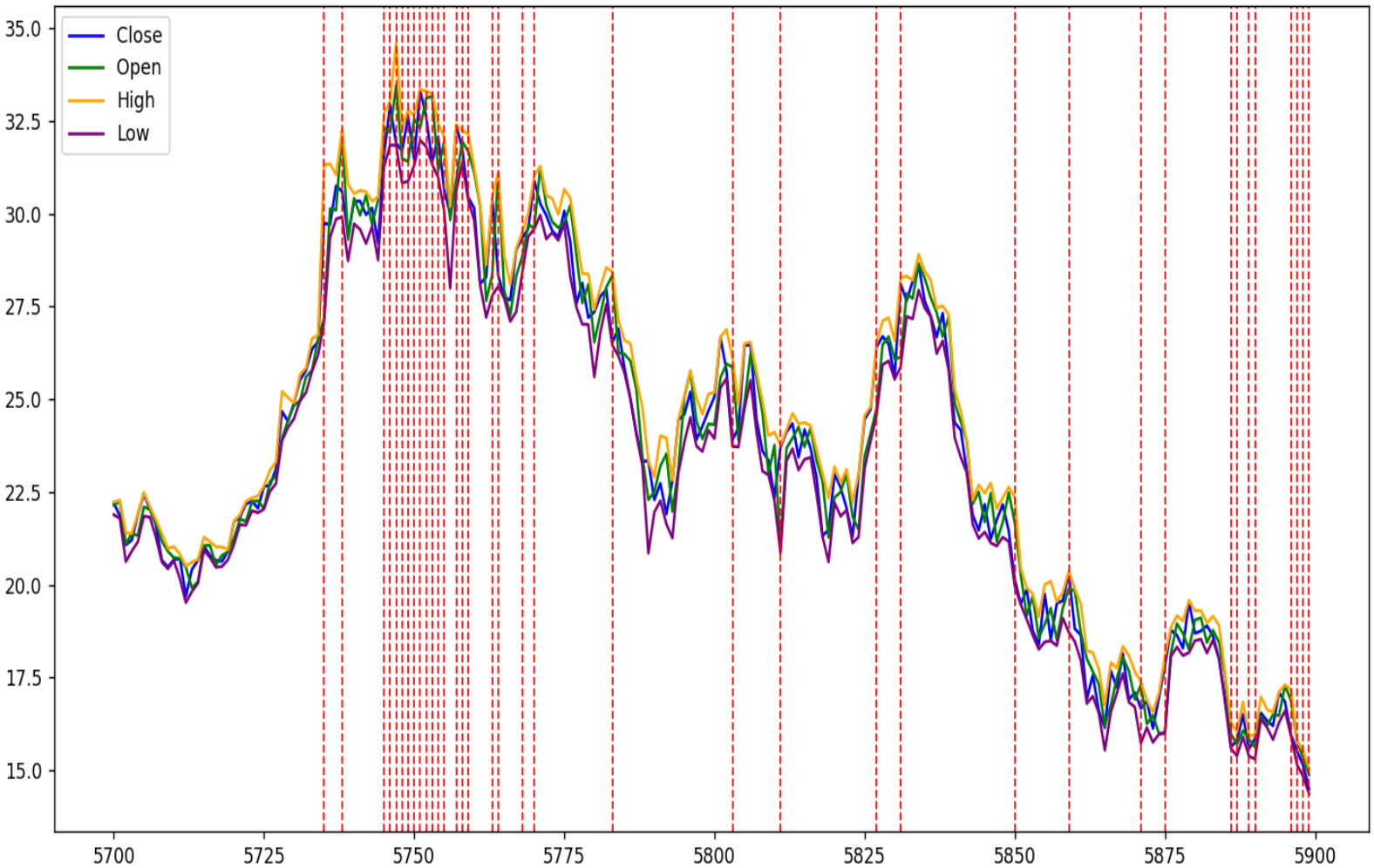
Rezultate obtinute - LSTM



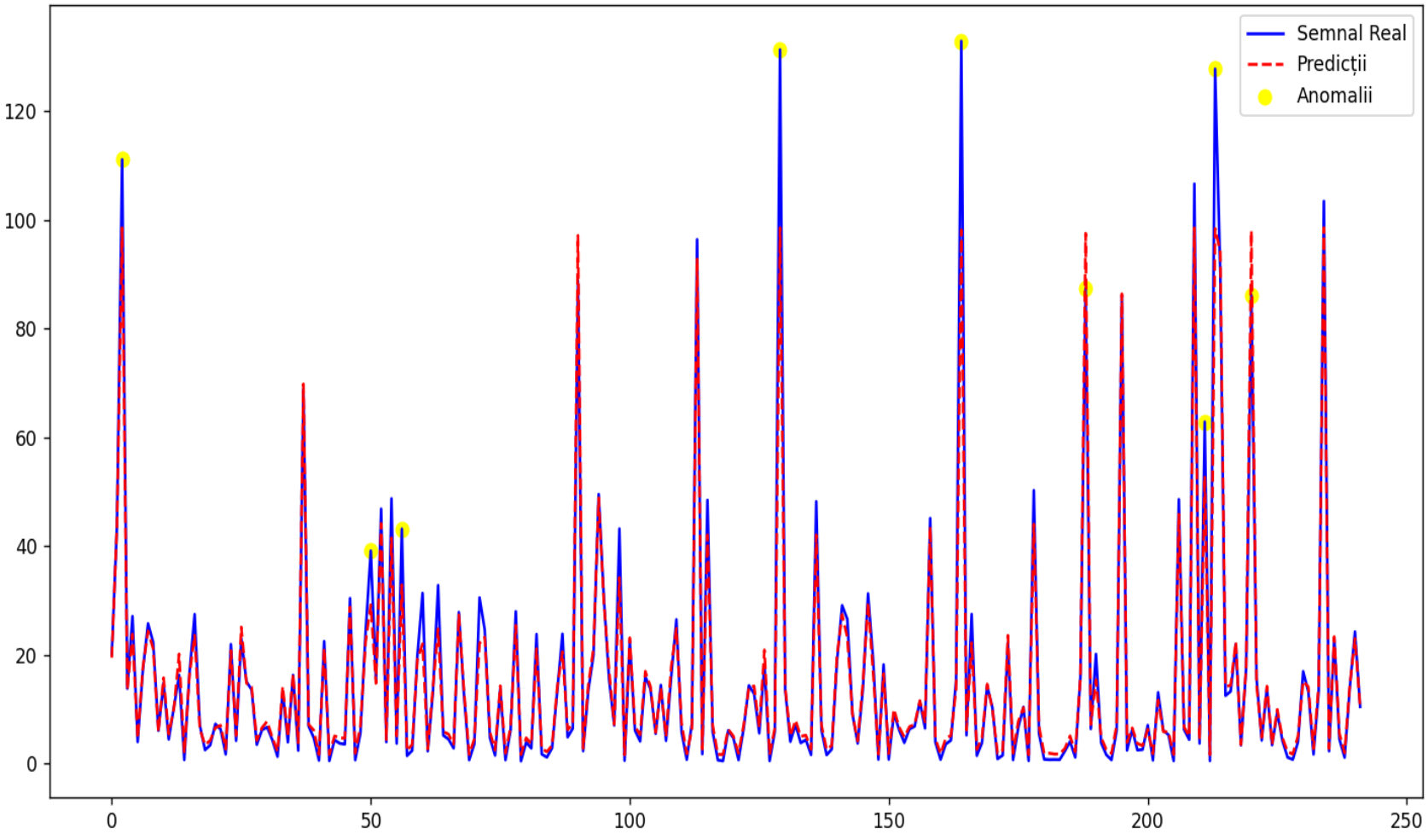
Rezultate obtinute - Autoencoders



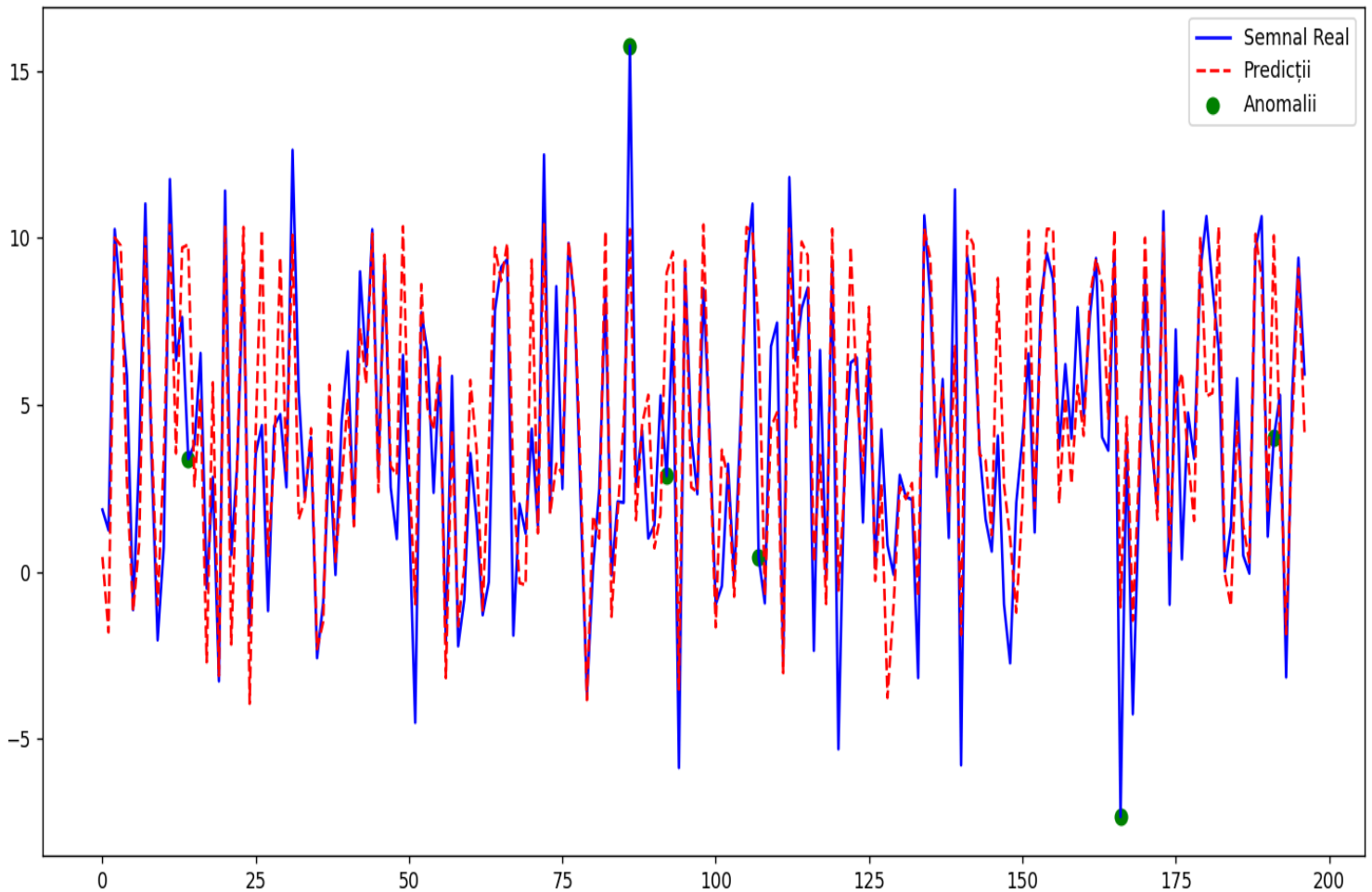
Rezultate obtinute - Isolation Forest



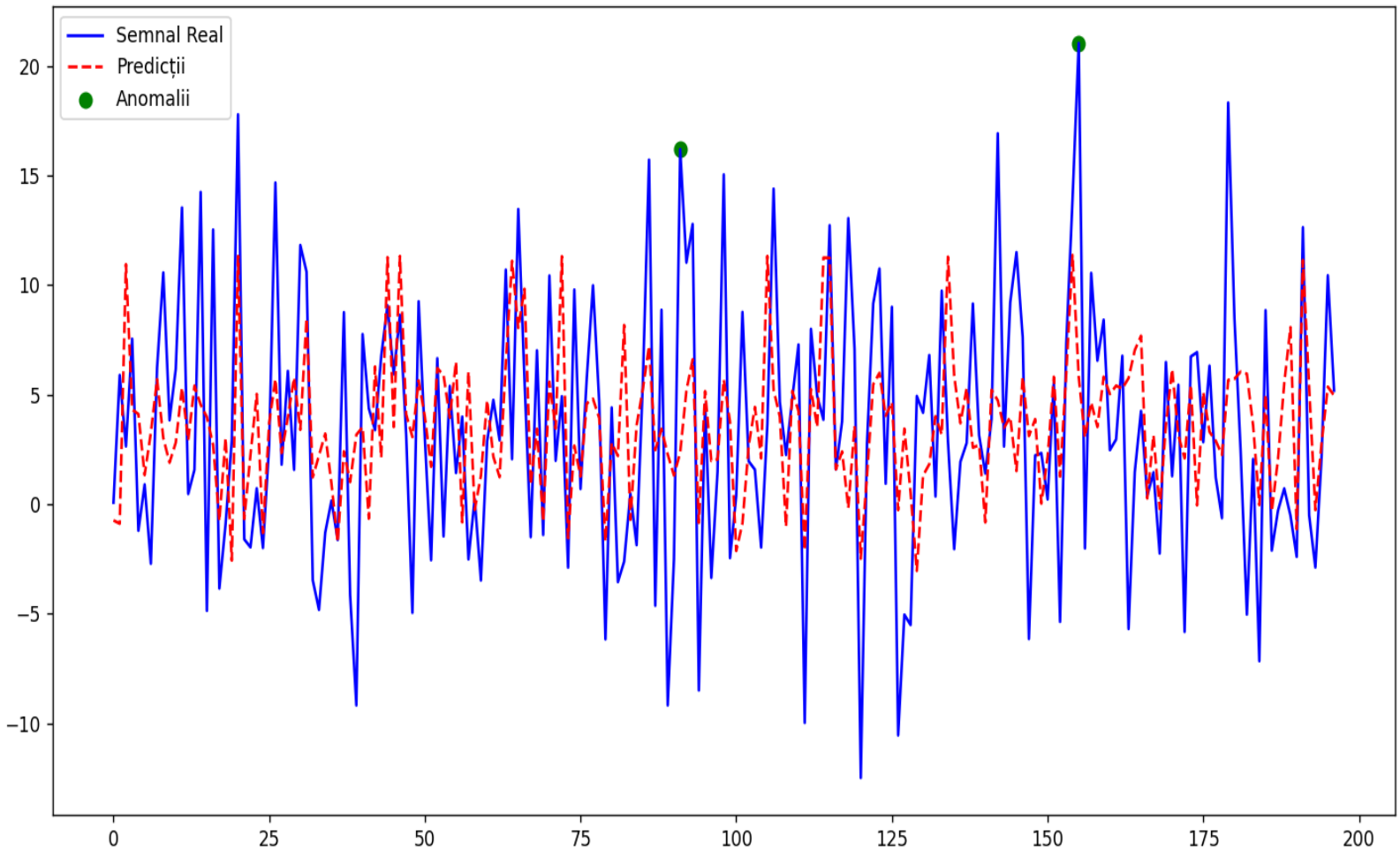
Rezultate obtinute - Transformers 1



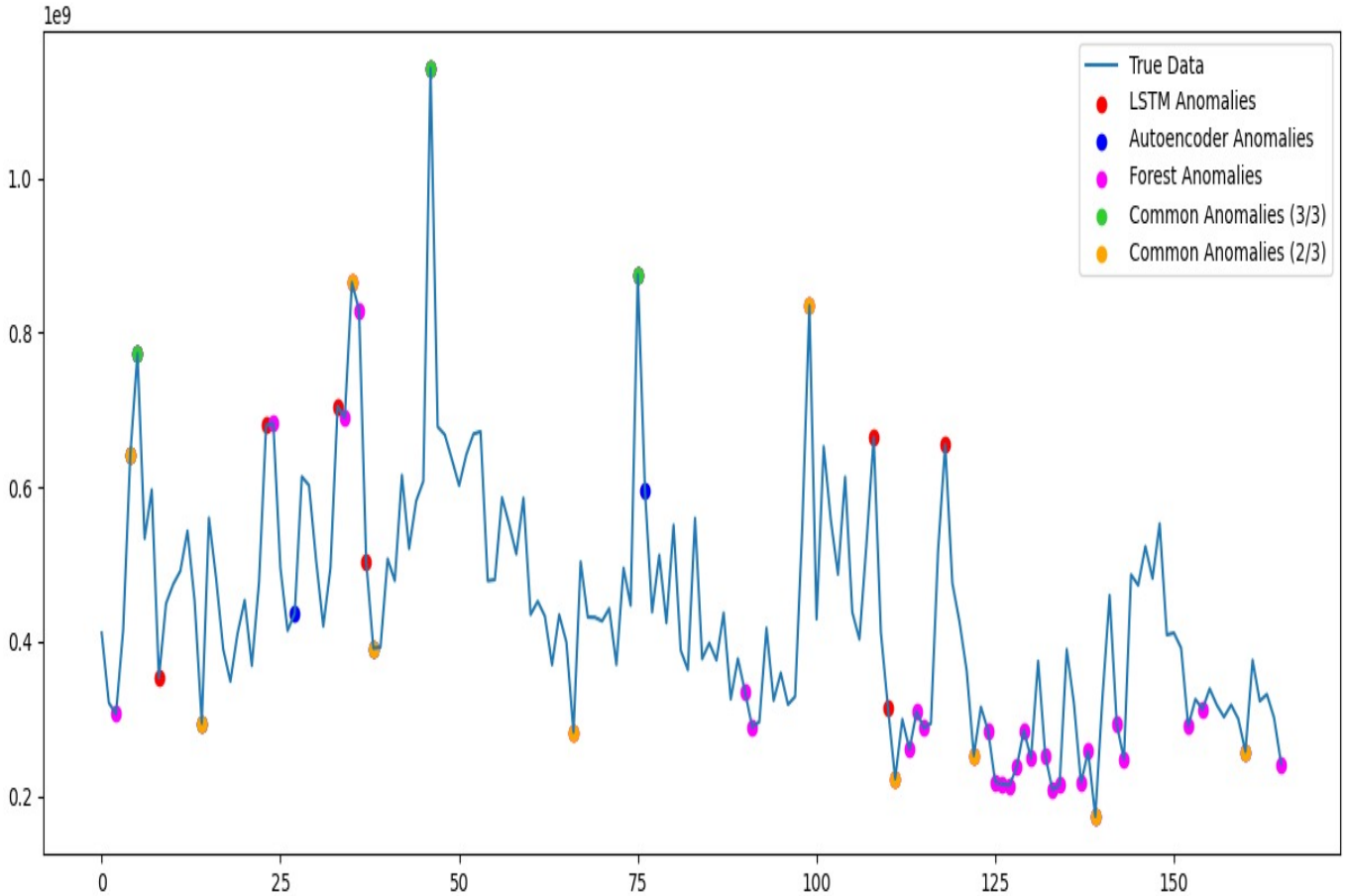
Rezultate obtinute - Transformers 2



Rezultate obtinute - Transformers 3



Rezultate obtinute - Comparație



Concluzii

- Transformer - se bazeaza pe Self Attention pentru a detecta relatiile dintre esantioane pe perioada foarte mari. Poate intelege dependente complexe. De aceea este bun pentru limbaj natural, pentru ca percepe intregul context, nu doar cuvintele din vecinatate.
- Isolation Forest - detecteaza anomalii atât in serii unidimensionale cat si serii cu mai multe dimensiuni, fara a fi nevoie de ajustari majore.
- Autoencoders - pot fi folosite pentru a detecta anomalii pe diferite tipuri de serii de timp (periodice e.g. ECG sau non-periodice e.g. volumul unei acțiuni).
- LSTM - prin arhitectura lor, reusesc sa identifice anomaliile prin modelarea comportamentului normal al datelor si detectarea deviatiilor semnificative fata de aceste tipare

Ce ne-am dori să mai facem

- Autoencoders
  - transpus datele pentru a folosi convoluția cum trebuie
  - explorat diferite arhitecturi de Autoencoder
- General
  - experimentat pe alte serii de timp

Întrebări?

Mulțumim pentru atenția acordată!

Bibliografie

- Keras - Timeseries Anomaly Detection using Autoencoders
- YouTube (Markus Thill) - Temporal Convolutional Networks
- GeeksForGeeks - How Isolation Forests work
- (Link Amazon) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 2nd Edition, Aurélien Géron
- (Medium) Transformer — A detailed explanation from perspectives of tensor shapes and PyTorch implementation.

Ce haz aveam prin anul I de facultate

Transport	FKTransportor
345	10
346	10

Transportor	Nume
10	'Fast Fourier Transport'

Table: Forma Normala

Transport	Nume
345	'Fast Fourier Transport'
346	'Fast Fourier Transport'

Table: Forma Denormalizata