CSCD94 Week 3 Review

Dominik Luszczynski

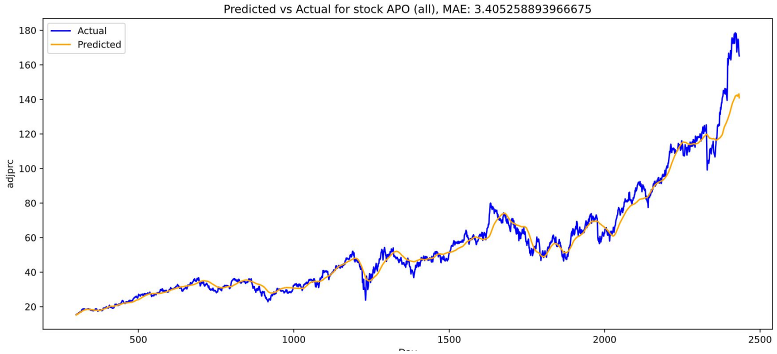
Problems

- When doing a white box adversarial attack, the gradients are necessary for generating an adversarial example.
  - This requires the use of model(x) where x is a tensor with require_grad_ set to True.

- Problem 1: If we only perturb adjprc, then we need to keep gradients active during feature generation.

Solution: Convert the Model Preprocessing from Pandas to Pytorch

- Implement rolling means and standard deviations in pytorch.
- Implement exponential moving averages (with the definition from pandas https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.ewm.html)
- Implement the log returns, ROC.
- Manually do a standardization (since the TimeSeriesDataset does it for you)
- Retrain the NHiTs model with the new features for consistency.



Predicted vs Actual for stock APO (all), MAE: 3.405258893966675

Problem 2: The Payload

- Normally, when you do model(x), you input x which is a Tensor, but PyTorch Forecasting does not accept a Tensor, but a dictionary which is derived from the TimeSeriesDataset (the dataset from which you create your dataloader from).
- However, you cannot convert the attack input into a TimeSeriesDataset because you will break the computational graph (break the gradients).

Solution: Make a Dictionary For the Payload

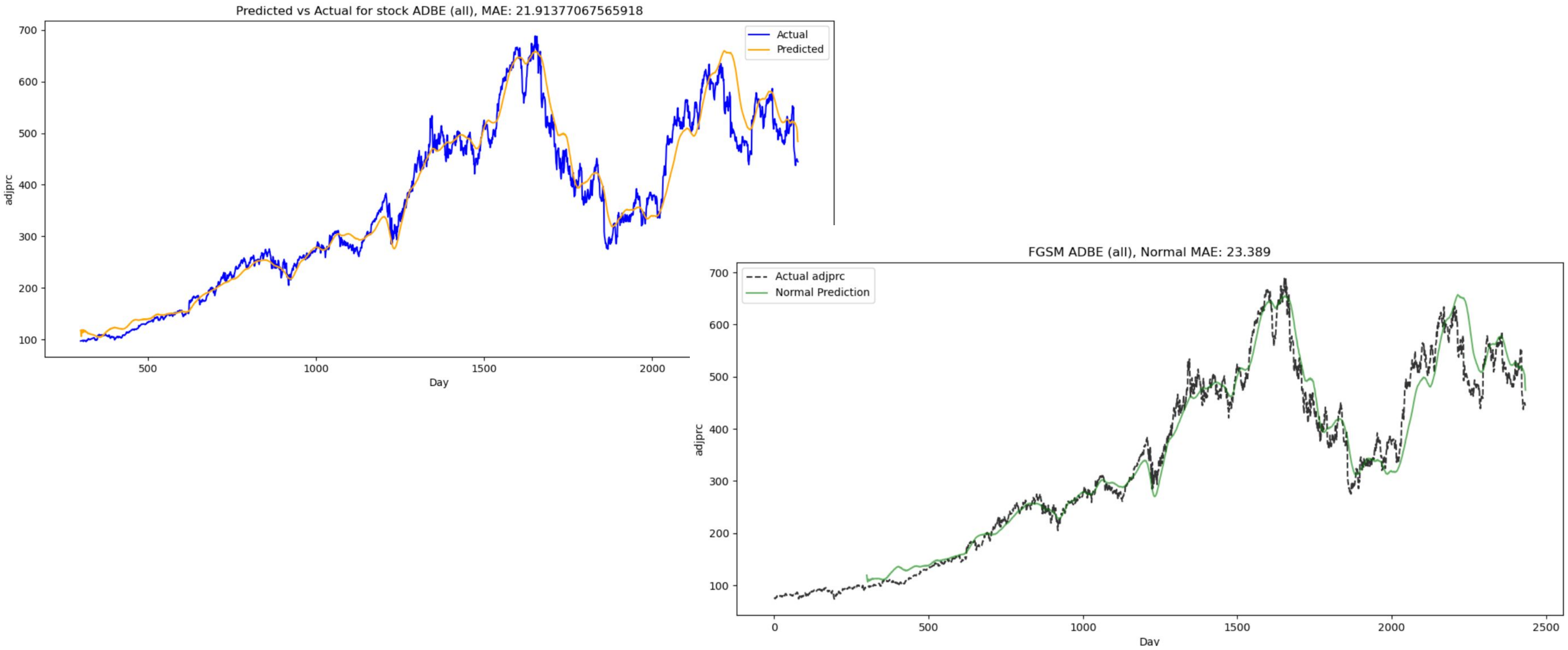
payload: (dictionary)

1. encoder_cat -> Your categorical features for your input, size: (B, 300, 1) where 300 is the lookback time
2. encoder_cont -> Your continuous features, size: (B, 300, 13)
3. encoder_lengths -> Tensor with the length of the encoder
4. encoder_target -> Unscaled adjprc for the encoder (the lookback period of 300 days)
5. decoder_cat -> Your categorical features for your output, size: (B, 50, 1) where 50 is the forecast length (next 50 days)
6. decoder_cont -> Your continuous features, size: (B, 50, 13)
7. decoder_lengths -> Tensor with the length of the decoder (length of the forecast)
8. decoder_target -> Unscaled adjprc for the decoder (the forecast period of 50 days)
9. target_scale -> the center and scale of the RobustScaler (which is what I used) where the center is the median and the scale is (q75 - q25) **/ 2**

Problem 3: The Output

- Normally when you perform model(x) you get the predictions back, but not with PyTorch Forecasting!
- Instead you get a tuple of 4 tensors:
  - 1) The predictions….of the quantiles (because I used QuantileLoss)
  - 2) The “Backcast”
  - 3) The “Block Forecasts”
  - 4) The “Block Backcasts”
- Solution: The actual predictions are the 0.5th quantile, so we take the 4th index of the quantile predictions (0.001, 0.01, 0.05, 0.5, 0.95, 0.99, 0.99)

Problem 4: Predictions With .predict() Differ From the Payload Predictions



Solution: Compare White Box Attacks with Payload Version

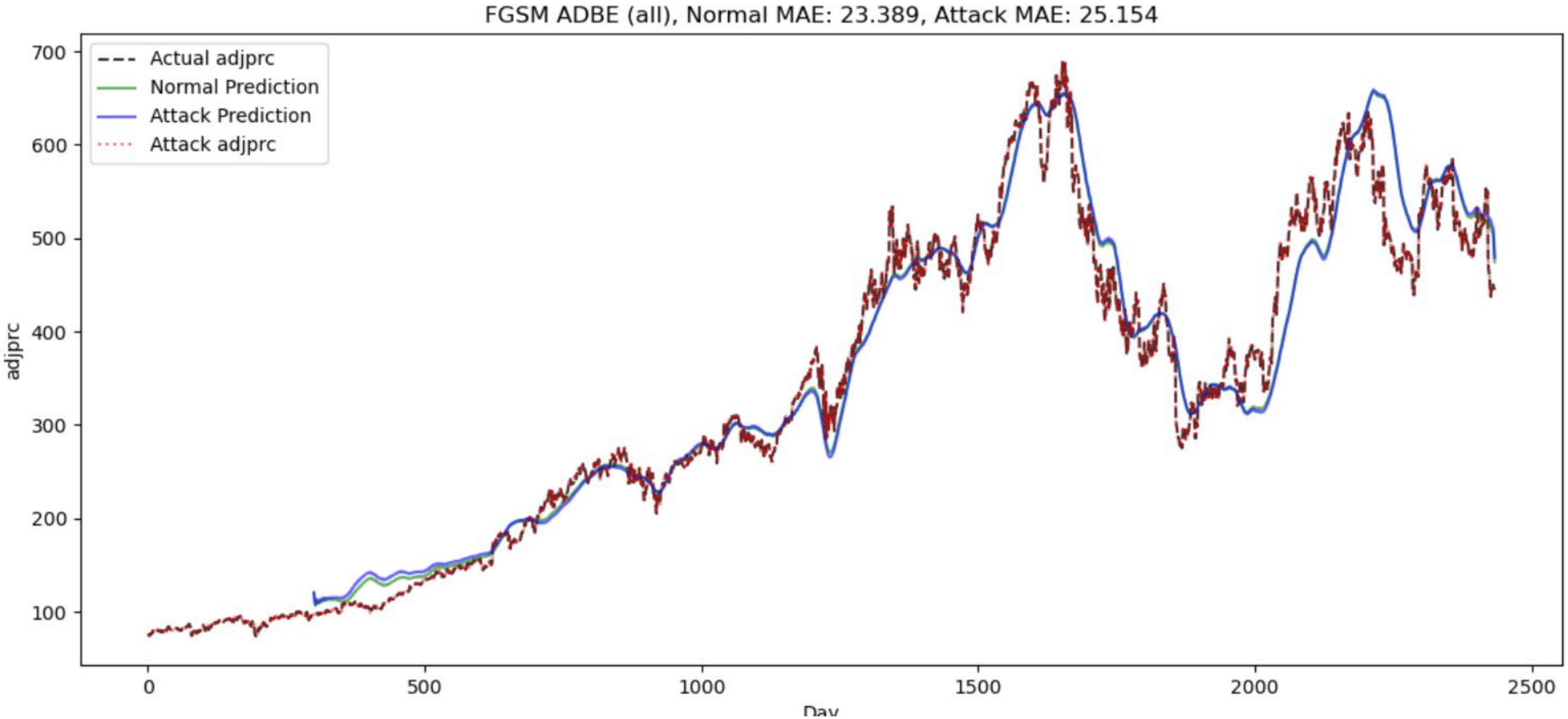
Attack: Fast Gradient Sign Method

$$adv_x = x + \varepsilon * sign(\nabla_x J(\theta, x, y)$$ [1]

Steps of attack [2]:

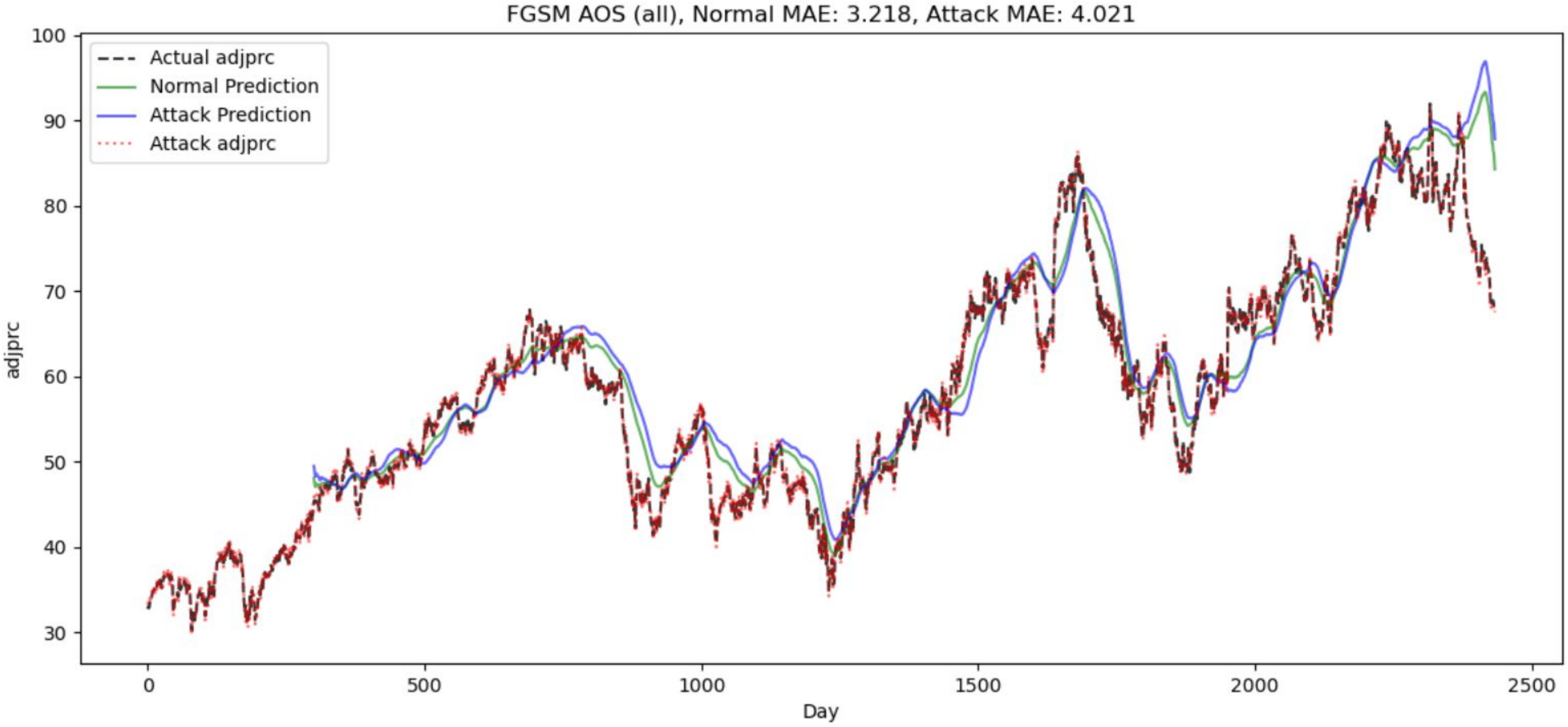
1) Call model(payload) with the normal adjprc to get predictions
2) In my case, take the average prediction for each time step of the 0.5 quantile (used scatter_add to preserve gradients)
3) Compute the loss function (whatever loss you want - I chose MAE)
4) Compute the gradient (model.zero_grad() -> loss.backward() -> grad = adjprc.grad.data)
5) New attack_adjprc = adjprc + eps * sign(grad), where eps=0.5

Examples

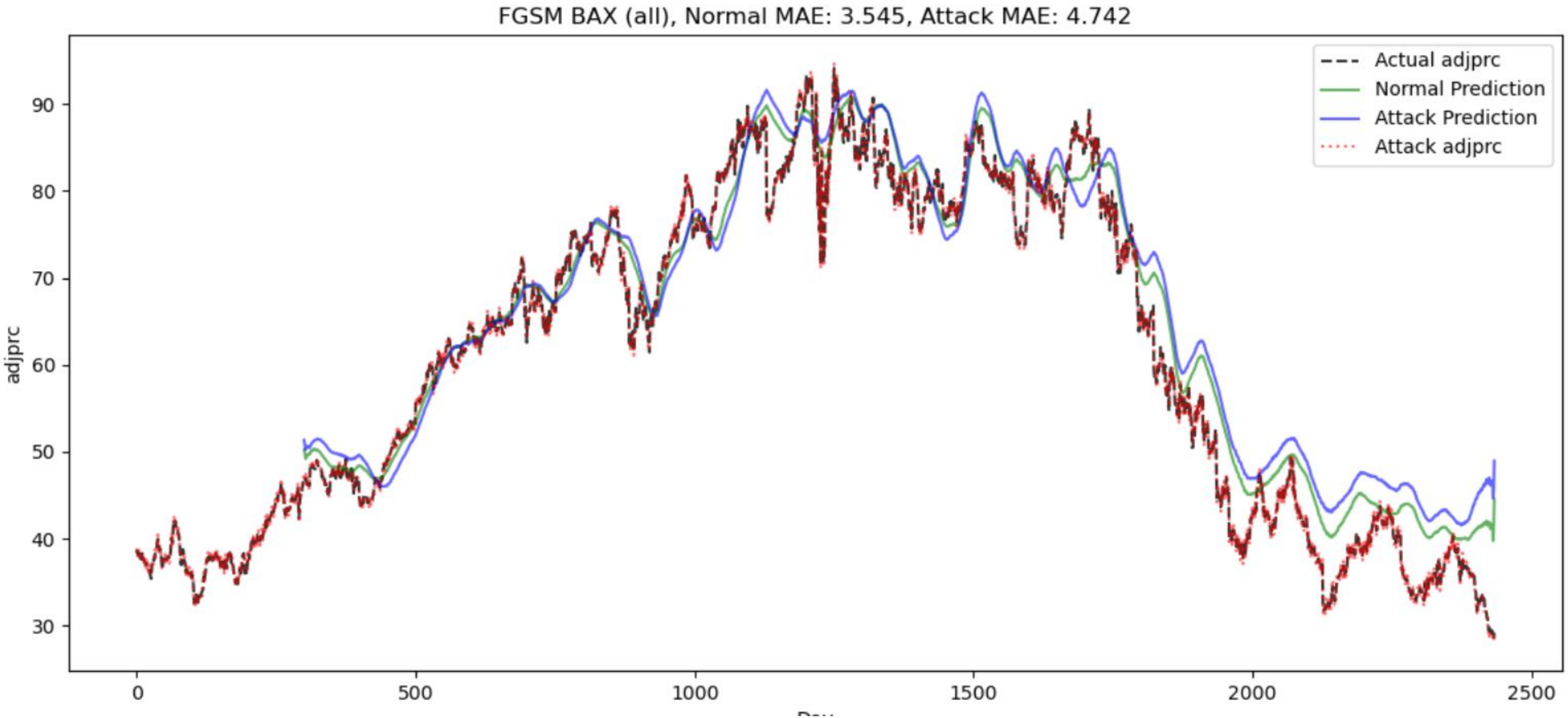


FGSM ADBE (all), Normal MAE: 23.389, Attack MAE: 25.154
Actual adjprc
Normal Prediction
Attack Prediction
Attack adjprc
adjprc
700
600
500
400
300
200
100
0
500
1000
1500
2000
2500
Day

Examples


FGSM AOS (all), Normal MAE: 3.218, Attack MAE: 4.021
Actual adjprc
Normal Prediction
Attack Prediction
Attack adjprc
adjprc
100
90
80
70
60
50
40
30
0
500
1000
1500
2000
2500
Day

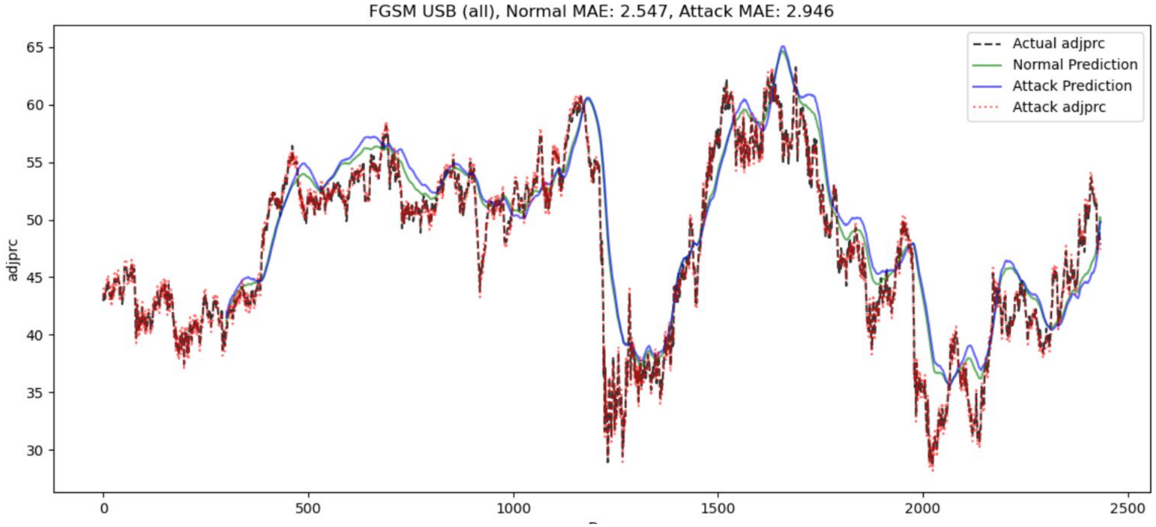
Examples


FGSM BAX (all), Normal MAE: 3.545, Attack MAE: 4.742
Actual adjprc
Normal Prediction
Attack Prediction
Attack adjprc
adjprc
90
80
70
60
50
40
30
0
500
1000
1500
2000
2500
Day

Overall Performance

Average MAE: 7.0966805555555545

Attack MAE: 8.024208333333334



Next Steps

- Experiment with the value of eps:
  - Ideally it should be dynamic, as the lower the adjprc the more noticeable a perturbation will be.
- Implement the FGSM on smaller windows (rather than the full recording).
- Implement other white box attacks:
  - Iterative methods
  - A paper titled “Temporal characteristics-based adversarial attacks on time series forecasting” implemented a targeted white box attack where they choose the direction of the attacked projections (published this year) [3].
- Start basic GAN implementation.

Citations

[1] J. Sen and S. Dasgupta, “Adversarial Attacks on Image Classification Models: FGSM and Patch Attacks and their Impact,” 2023, doi: 10.48550/arxiv.2307.02055.

[2] M. Gallagher, N. Pitropakis, C. Chrysoulas, P. Papadopoulos, A. Mylonas, and S. Katsikas, “Investigating machine learning attacks on financial time series models,” Computers & security, vol. 123, pp. 102933-, 2022, doi: 10.1016/j.cose.2022.102933

[1] Z. Shen and Y. Li, “Temporal characteristics-based adversarial attacks on time series forecasting,” Expert systems with applications, vol. 264, pp. 125950-, 2025, doi: 10.1016/j.eswa.2024.125950.