EACH

Escola de Artes, Ciências e Humanidades
da Universidade de São Paulo

SIN5006 - Inteligência Computacional

Trabalho de avaliação da disciplina (entrega parcial)

1o sem 2025 - aluno Ivan F. Gancev (16298145)

Professora Patrícia R. Oliveira

EACH

Escola de Artes, Ciências e Humanidades
da Universidade de São Paulo

CONTEXTO

Dados do artigo selecionado

Título: Stock Price Manipulation Detection using Variational Autoencoder and Recurrence Plots

Autores:

- *Khaled Safa*
  - Department of Computer Science, NTIC Faculty University of Constantine2, Abdelhamid Mehri, Constantine, Algeria
- *Ammar Belatreche*
  - Department of Computer and Information Sciences, Fac. of Engineering and Environment, Northumbria University, Newcastle Upon Tyne, UK
- *Salima Ouadfel*
  - Department of Computer Science, NTIC Faculty University of Constantine2, Abdelhamid Mehri, Constantine, Algeria

Publicação: 2024 International Joint Conference on Neural Networks (IJCNN)

Link: https://doi.org/10.1109/IJCNN60899.2024.10650234

Objetivo do artigo

O mercado de ações demanda constante necessidade de aprimoramento dos mecanismos que identificam cenários de manipulação ilícitas em seus preços. Isso interfere diretamente na confiança dos investidores e atualizações de normas feitas por seus reguladores.

O artigo escolhido desenvolve um método de identificação de manipulações em preços de ações usando *Recurrence Plots* e *beta Variational Autoencoders*, usando redes neurais convolucionais. Este modelo demonstrou alta eficácia na identificação de manipulações usando dados do projeto LOBSTER, que reuniu dados oficiais da NASDAQ das empresas Amazon, Google, Intel, Microsoft e Apple.

Conjuntos de dados

O projeto LOBSTER (Limit Order Book System - The Efficient Reconstructor) possui um conjunto de dados contendo o **livro de ofertas** e o **histórico de ordens** de 5 ativos da bolsa de NASDAQ. Os dados do artigo são os de 1o nível de 21/jun/2012. Os ativos são:

- Amazon: **AMZN**
- Apple: **AAPL**
- Google: **GOOG**
- Intel: **INTC**
- Microsoft: **MSFT**

Ref: https://lobsterdata.com/info/DataSamples.php

EACH

Escola de Artes, Ciências e Humanidades
da Universidade de São Paulo

ENTREGA PARCIAL

1) Definição do problema

Identificação de manipulações ilícitas em preços de 5 ações de Nasdaq, usando os dados de 21/jun/2012 disponíveis no projeto LOBSTER.

2) Métodos utilizados

Recurrence Plots

- Introduzido por **Eckmann** e outros em 1987
- É uma técnica avançada de análise de dados não lineares. É uma visualização (ou um gráfico) de uma matriz quadrada, na qual os elementos da matriz correspondem aos momentos em que um estado de um sistema dinâmico se repete.
- Tecnicamente, o RP revela todos os momentos em que a trajetória do espaço de fase do sistema dinâmico visita aproximadamente a mesma área no espaço de fase.
- Ref: http://www.recurrence-plot.tk/glance.php

Variational Autoencoder

- Por **Diederik P. Kingma** e **Max Welling**
- É um modelo gráfico probabilístico composto por uma rede de codificadores, uma rede de decodificadores e uma função de perda. O codificador mapeia os dados de entrada em uma distribuição com uma média e um desvio padrão, então a variável latente é amostrada dessa média e desvio padrão e o decodificador aprende a reconstruir os dados através da representação oculta por *backpropagation* dos erros da função de perda.
- Ref: https://arxiv.org/pdf/1312.6114

2) Métodos: Recurrence Plots

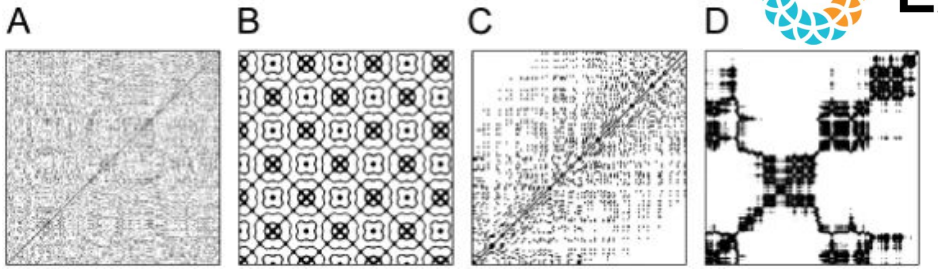
Hipótese

- Quando estamos no verão, em um dia de muito calor, muita umidade e surgem nuvens escuras. Sabemos que haverá uma pancada de chuva.
- Isso acontece porque somos capazes de observar a recorrência deste evento.

O que são Recurrence Plots?

- Recurrence plots visualizam padrões de repetição em sistemas dinâmicos, mostrando quando o sistema retorna a estados passados similares.
- Através dessa ferramenta gráfica, é possível visualizar recorrências em sistemas dinâmicos e suas séries de dados, como distribuição assintótica, comportamento de órbitas ou séries temporais.

2) Recurrence Plots



Características

- Criação de uma matriz de recorrências que avalia a combinação de 2 valores e indica se é uma recorrência ou não (0 para não recorrente e 1 para recorrente) de acordo com um limite estabelecido (*threshold*).

$$\mathbf{R}_{i,j}(\varepsilon) = \Theta(\varepsilon - \| \vec{x}_i - \vec{x}_j \|), \quad i,j = 1,\ldots,N,$$

(i.e. $\Theta(x) = 0$, if $x < 0$, and $\Theta(x) = 1$ otherwise)

- O limite parametrizado é importante por determinar a tolerância do modelo. Um valor elevado encontrará recorrências demais e um valor baixo não trará resultados.
- A Dimensão determina a dimensionalidade do espaço no qual a trajetória do sistema é imersa.
- O *Time delay* é o intervalo de tempo considerado entre os componentes dos vetores de estado reconstruídos.
- **A**: Homogêneo; **B**: periódico/harmônico; **C**: Drift (linear com desvios); **D**: Disruptivo

3) Entrega parcial (30/abr/25)

Previsto:

Criar a implementação para *Recurrence Plots* com os dados usados no artigo

Realizado:

Código desenvolvido e impressão dos gráficos de recorrência realizada com sucesso.

Dificuldade: Interpretação dos gráficos de recorrência

> A seguir demonstraremos o código implementado em Python e alguns exemplos de *recurrence plots*

3) Entrega: Código usado

Exemplo do arquivo de ordens

best_ask: Melhor preço de venda
best_ask_size: volume de ações
best_bid: Melhor preço de compra
best_bid_size: volume de ações

	best_ask	best_ask_size	best_bid	best_bid_size
0	5859400	200	5853300.0	18.0
1	5859100	18	5853300.0	18.0
2	5859200	18	5853300.0	18.0
3	5859300	100	5853300.0	18.0
4	5859300	100	5853600.0	18.0
5	5859300	100	5857300.0	20.0
6	5857400	40	5857300.0	20.0
7	5857500	82	5857300.0	20.0
8	5857500	57	5857300.0	20.0
9	5857500	57	5857300.0	19.0
10	5857500	57	5857300.0	9.0
11	5857500	32	5857300.0	9.0
12	5857500	27	5857300.0	9.0
13	5857500	20	5857300.0	9.0
14	5857800	45	5857300.0	9.0

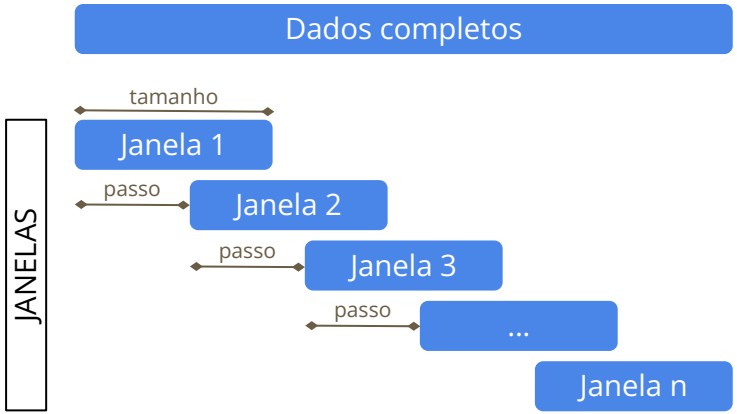
1 passo: Carga do arquivo

```python
# ------------------------------------------------------------------------------
# 1. Carga do Livro de Ofertas (Level 1)
# ------------------------------------------------------------------------------

def carregar_livro_ordens(arquivo_orderbook):
    """
    Carrega o livro de ordens a partir de um arquivo CSV de acordo com o layout LOBSTER.
    Args:
        arquivo_orderbook (str): Caminho para o arquivo CSV do livro de ordens.
    Returns:
        pd.DataFrame: DataFrame contendo o livro de ordens.
    """

    livro_ordens = pd.read_csv(arquivo_orderbook, header=None)
    livro_ordens.columns = ['best_ask', 'best_ask_size','best_bid', 'best_bid_size' ]
    #livro_ordens['best_ask'] = livro_ordens['best_ask'].astype(float)
    #livro_ordens['best_bid'] = livro_ordens['best_bid'].astype(float)

    return livro_ordens
```

3) Entrega: Código usado

Janelas deslizantes

Contém as colunas 'best_bid' e 'best_ask', o **tamanho** determina quantas linhas terá cada janela e o **passo** é a distância em linhas do 1o elemento de cada janela. Este passo serve para reduzir o total de dados em janelas analisáveis



2o passo: Geração das janelas deslizantes

```python
# ----------------------------------------------------------------------------
# 2. Geração de Janelas Deslizantes
# ----------------------------------------------------------------------------

def criar_janelas_deslizantes(dados, tamanho_janela, passo):
    """
    Cria janelas deslizantes a partir dos dados.
    Args:
        dados (pd.DataFrame): DataFrame contendo a série temporal.
        tamanho_janela (int): Tamanho de cada janela.
        passo (int): Passo entre as janelas.
    Returns:
        list: Lista de arrays NumPy representando as janelas.
    """
    janelas = []
    num_pontos = len(dados)
    for i in range(0, num_pontos - tamanho_janela + 1, passo):
        janela = dados[i:i + tamanho_janela].to_numpy()
        janelas.append(janela)
    return janelas
```

3) Entrega: Código usado

3o passo: Geração dos recurrence plots

Recurrence plots

De acordo com o limite estipulado e o percentual de tolerância, para cada janela é gerada uma matriz que marca as recorrências de cada valor ao longo do conjunto de dados

```
[[1. 1. 1. ... 0. 0. 0.]
 [1. 1. 1. ... 0. 0. 0.]
 [1. 1. 1. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 1. 1. 1.]
 [0. 0. 0. ... 1. 1. 1.]
 [0. 0. 0. ... 1. 1. 1.]]
```

Cada valor na matriz indica se aquele valor é recurrente ou não

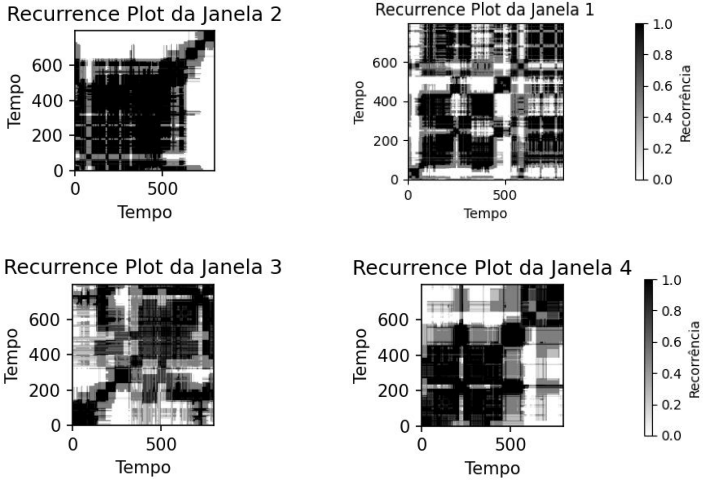
```python
# ------------------------------------------------------------------------------
# 3. Geração de Recurrence Plots
# ------------------------------------------------------------------------------

def gerar_recurrence_plots(janelas, time_delay, dimension, threshold, percentage):
    """
    Gera Recurrence Plots para uma lista de janelas deslizantes.
    Args:
        janelas (list): Lista de arrays NumPy representando as janelas.
        time_delay (int): Atraso para o embedding.
        dimension (int): Dimensão do embedding.
        threshold (str or float): Limiar para recorrência.
        percentage (float): Percentual para o limiar pointwise.
    Returns:
        list: Lista de arrays NumPy representando os Recurrence Plots.
    """
    recurrence_plots = []
    rp = RecurrencePlot(time_delay=time_delay, dimension=dimension, threshold=threshold, percentage=percentage)
    for janela in janelas:
        # RecurrencePlot espera uma entrada 2D: [n_samples, n_timestamps]
        # Se a janela tiver múltiplas features, precisamos decidir como processar
        # Aqui, vamos gerar um RP para cada feature separadamente e armazenar
        rps_janela = []
        if janela.ndim > 1:
            for feature in range(janela.shape[1]):
                rp_feature = rp.fit_transform(janela[:, feature].reshape(1, -1))[0]
                rps_janela.append(rp_feature)
            recurrence_plots.append(np.mean(rps_janela, axis=0) if rps_janela else None) # Média dos RPs das features
        elif janela.ndim == 1:
            recurrence_plot = rp.fit_transform(janela.reshape(1, -1))[0]
            recurrence_plots.append(recurrence_plot)
        else:
            recurrence_plots.append(None)
    return [rp for rp in recurrence_plots if rp is not None]
```

3) Entrega: Código usado

4o passo: Visualização dos recurrence plots

Visualização

Seguindo os valores das matrizes, é impresso um gráfico que gera uma visualização das recorrências



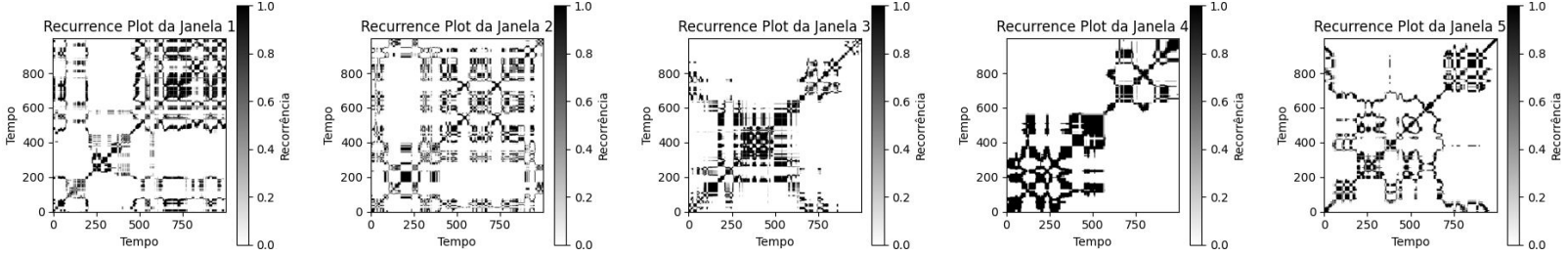

```python
# ----------------------------------------------------------------------
# 4. Impressão dos Gráficos de Recurrence Plots
# ----------------------------------------------------------------------

def visualizar_recurrence_plots(recurrence_plots, num_graficos=5):
    """
    Visualiza os Recurrence Plots gerados.
    Args:
        recurrence_plots (list): Lista de arrays NumPy representando os RPs.
        num_graficos (int): Número de gráficos para exibir.
    """
    num_plots = min(num_graficos, len(recurrence_plots))
    plt.figure(figsize=(15, 5 * num_plots))
    for i in range(num_plots):
        plt.subplot(num_plots, 1, i + 1)
        plt.imshow(recurrence_plots[i], cmap='binary', origin='lower')
        plt.title(f'Recurrence Plot da Janela {i + 1}')
        plt.xlabel('Tempo')
        plt.ylabel('Tempo')
        plt.colorbar(label='Recorrência')
    #plt.tight_layout()
    plt.show()
```

Janela: 1000
Passo: 500

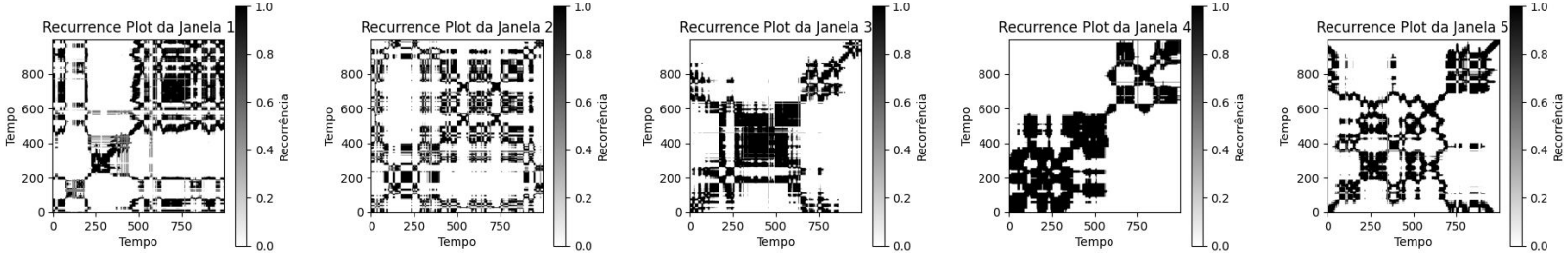
EACH

4) Experimentos: Preços de ações Amazon (AMZN)

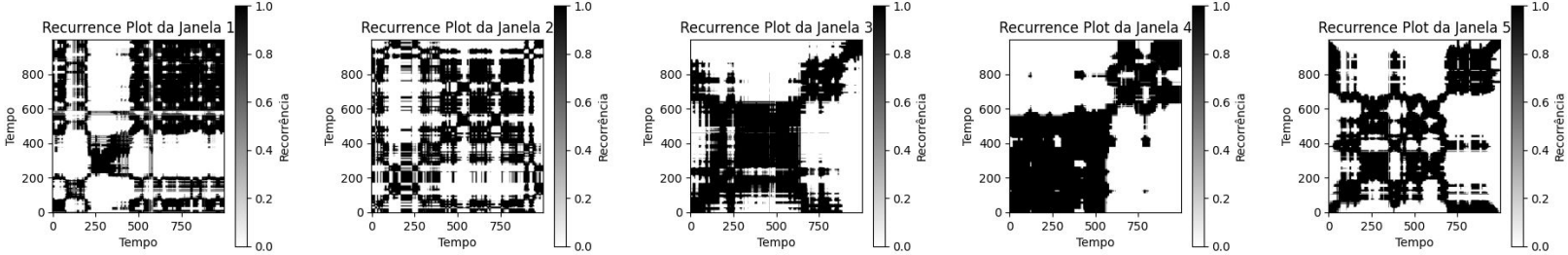
Threshold **5%**:



Threshold **10%**:



Threshold **20%**:

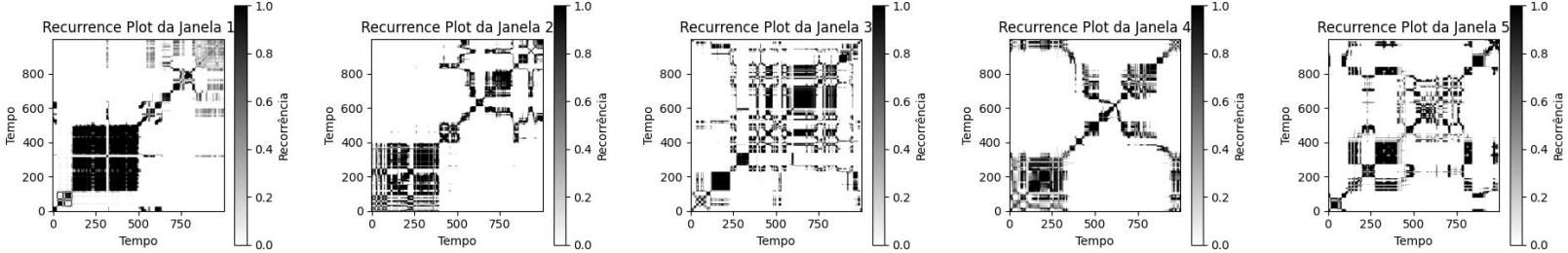


Janela: 1000
Passo: 500

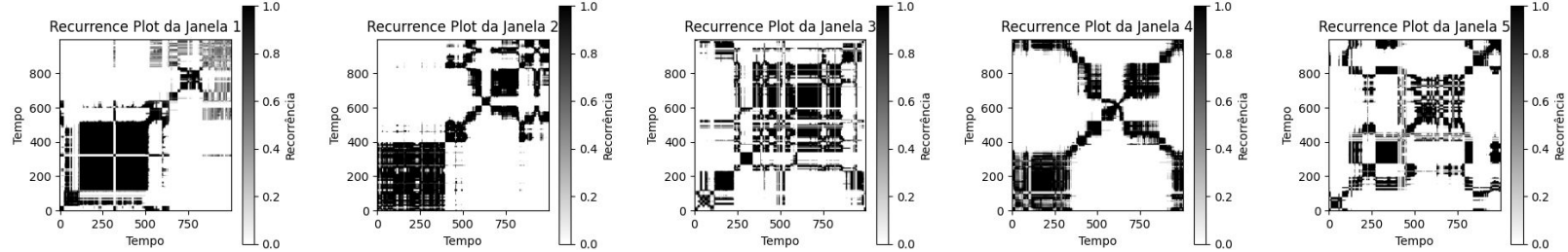
EACH

4) Experimentos: Preços de ações Apple (AAPL)

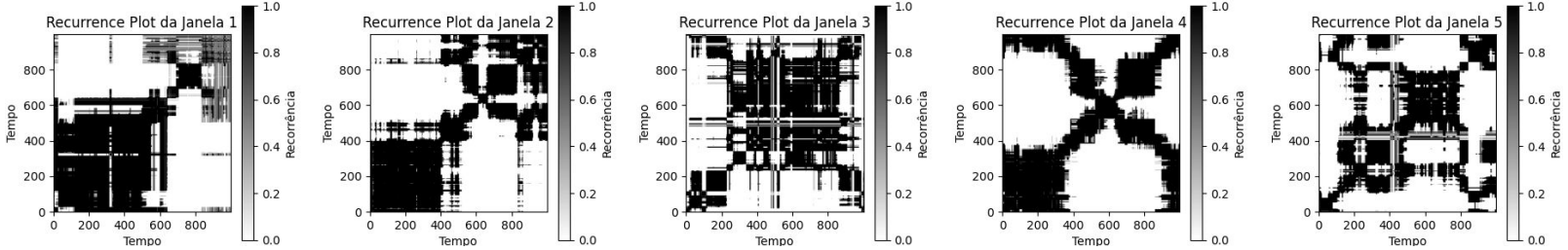
Threshold **5%**:



Threshold **10%**:



Threshold **20%**:

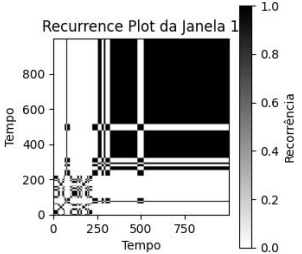


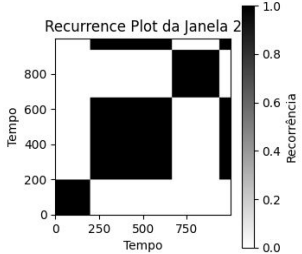
Janela: 1000
Passo: 500

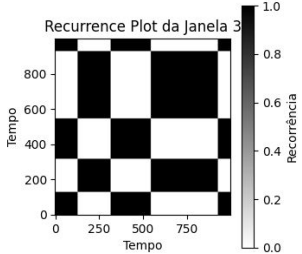
EACH

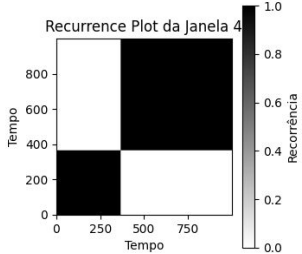
4) Experimentos: Preços de ações Intel (INTC)

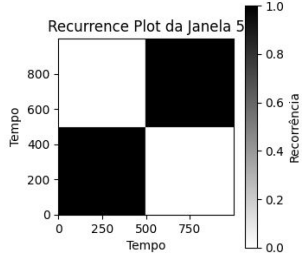
Threshold **5%**:



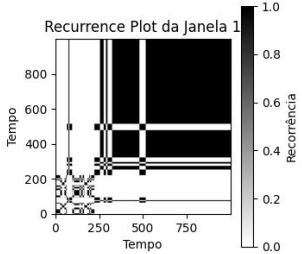


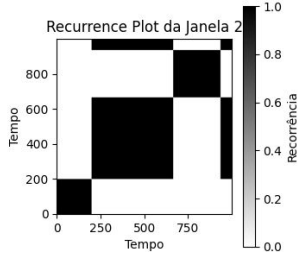


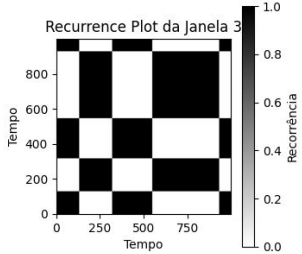


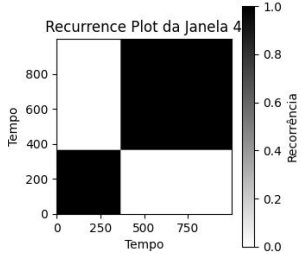


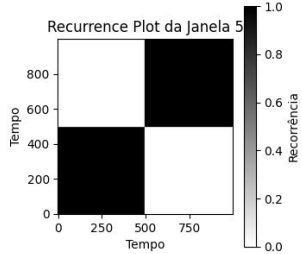
Threshold **10%**:



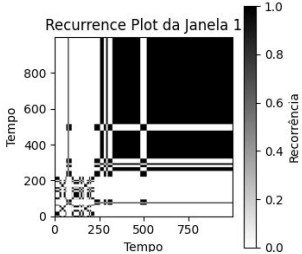


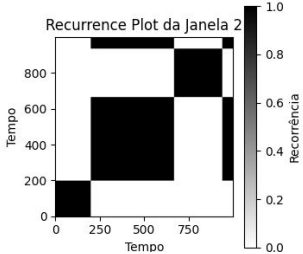


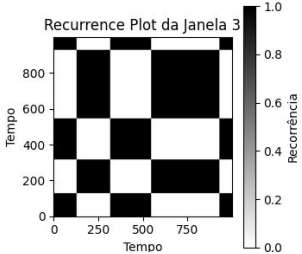


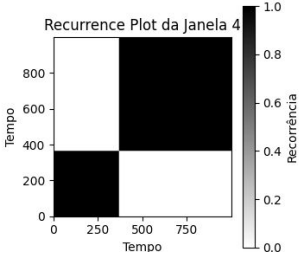


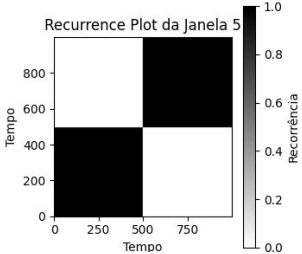
Threshold **20%**:









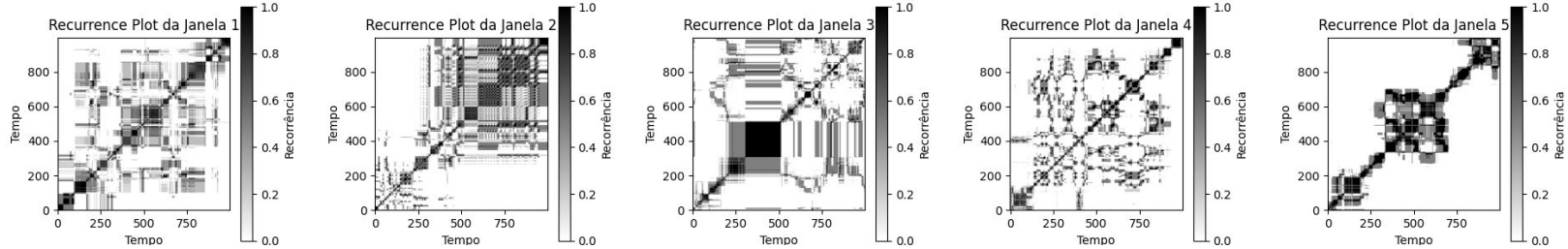


Janela: 1000
Passo: 500

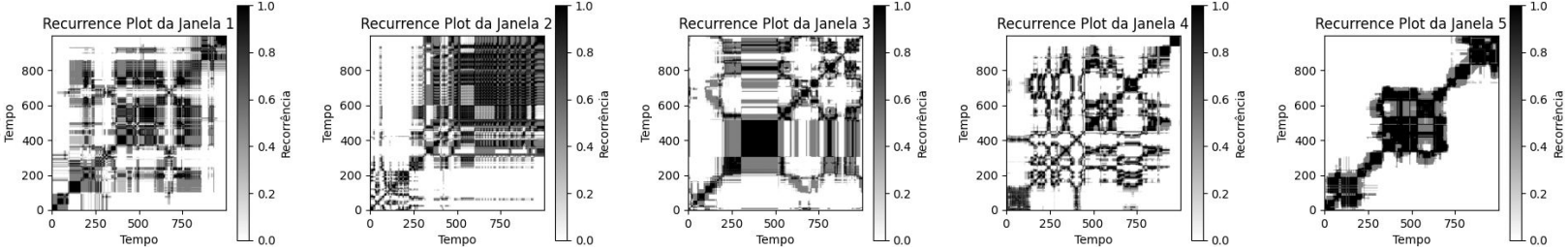
EACH

4) Experimentos: Preços de ações Google (GOOG)

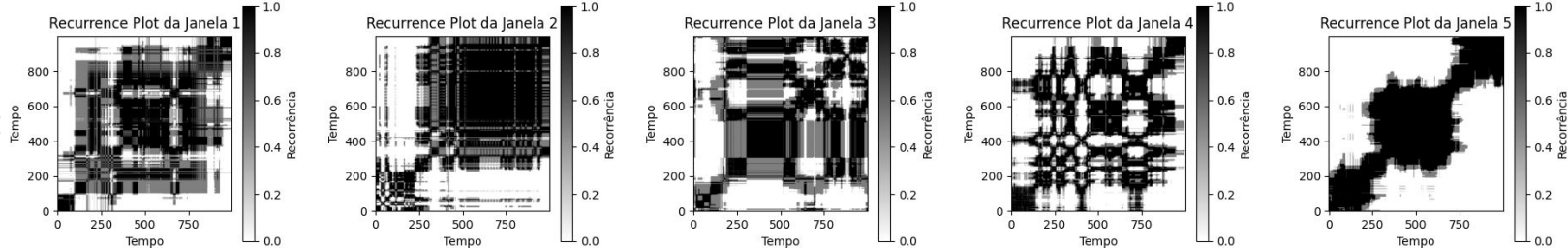
Threshold **5%**:



Threshold **10%**:



Threshold **20%**:

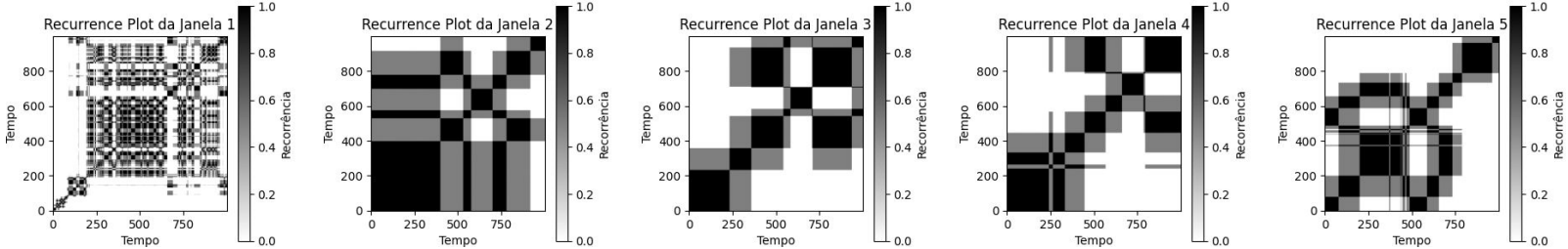


Janela: 1000
Passo: 500

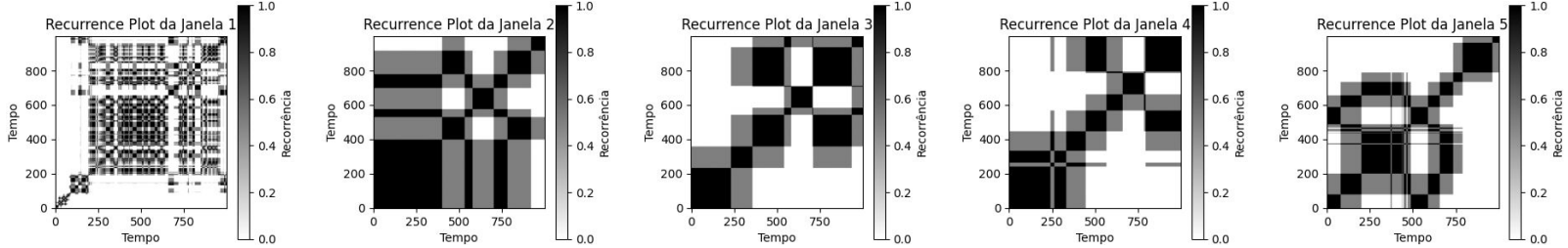
EACH

4) Experimentos: Preços de ações Microsoft (MSFT)

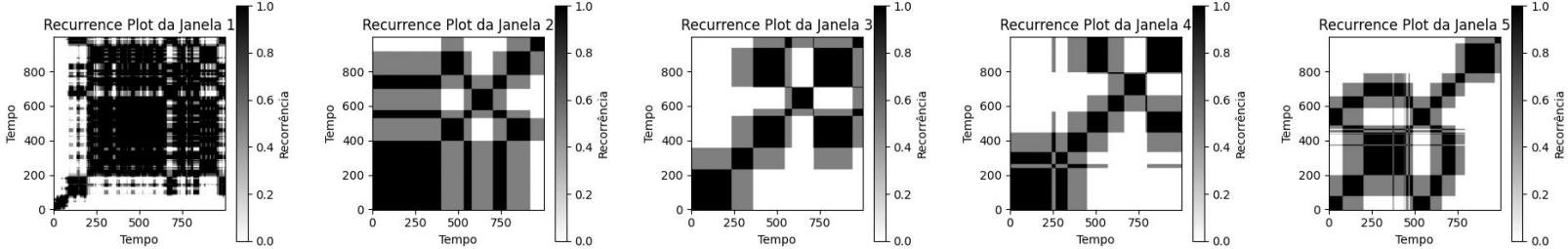
Threshold **5%**:



Threshold **10%**:



Threshold **20%**:



4) Seleção de hiper-parâmetros e medida de avaliação

- Para esta entrega parcial, não foi utilizada nenhuma técnica de seleção de hiper-parâmetros ou de medidas de avaliação. Isto será realizado durante os treinamentos aplicados para a **entrega final**.

5) Discussão dos resultados

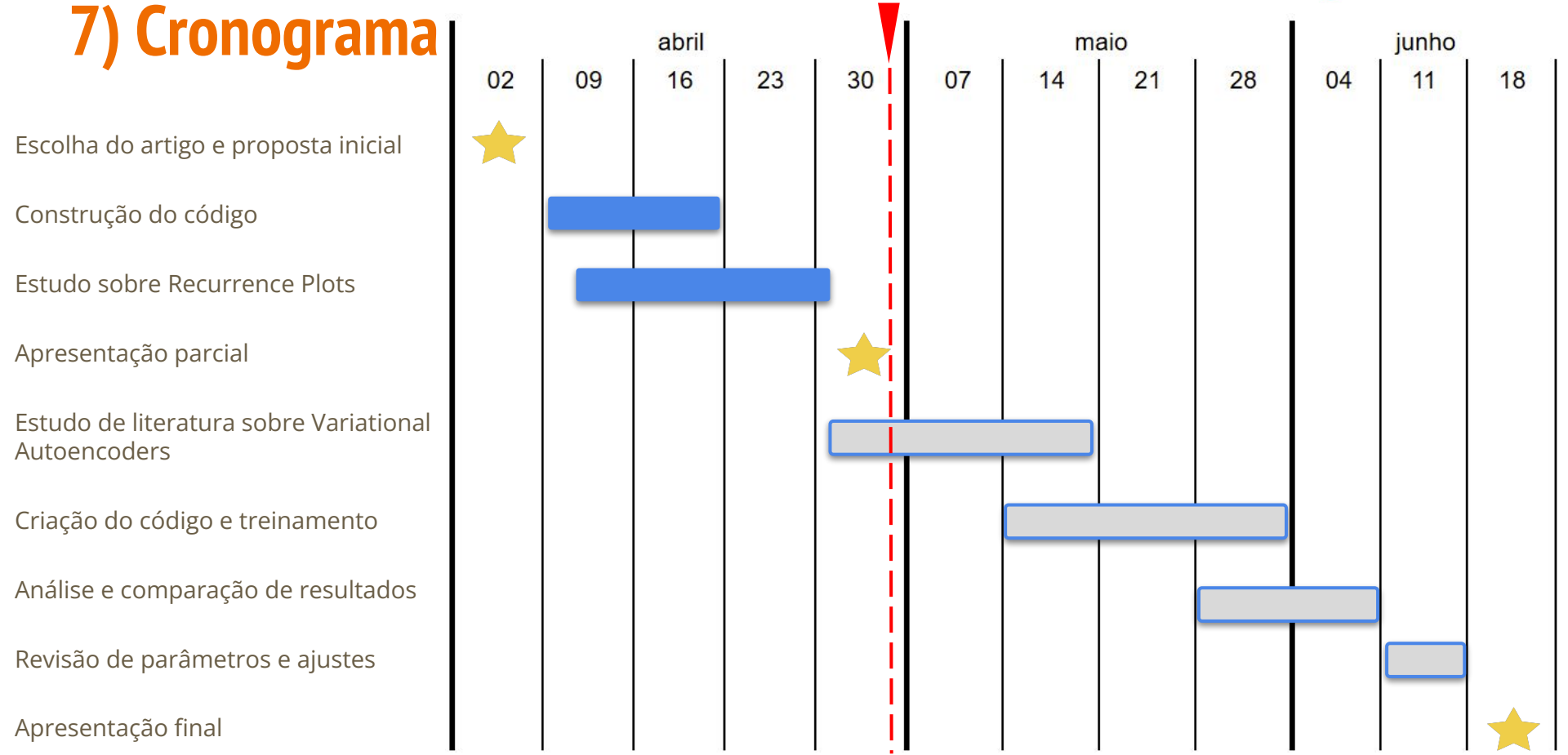
- Os resultados até o momento mostraram diferentes características e distribuição dos dados para cada uma das ações (entradas) disponíveis.
- A escolha do método de *recurrence plots* permite identificar valores sem recorrências dentro das janelas de tempo e esse indicador será usado para apontar potencial manipulação ilícita nos preços.
- Nos próximos passos, esses gráficos servirão de entrada para processamento

6) Próxima entrega do projeto

Apresentação final (18/jun):

- Criar a implementação para *Variational Autoencoders* para o mesmo conjunto de dados
- Comparar os resultados obtidos com os resultados demonstrados no artigo

7) Cronograma



Referências

- Eckmann. “Recurrence plots”, http://www.recurrence-plot.tk/glance.php
- Marwan, Norbert. “*Recurrence plots for the analysis of complex systems*”, jan 2007, https://doi.org/10.1016/j.physrep.2006.11.001
- Safa, Khaled. “*Stock Price Manipulation Detection using Variational Autoencoder and Recurrence Plots*”, jul 2024, https://doi.org/10.1109/IJCNN60899.2024.10650234

EACH

Obrigado!