# Previsão de Ativos Financeiros com Deep Learning

## Comparativo: CNN, LSTM e Transformer

Breno de Lima Galves

18 de dezembro de 2025

# Agenda

# Objetivo e Escopo

## Objetivo

Prever o preço de fechamento (*Close*) do índice S&P 500 (ĜSPC) com horizonte de previsão de 1 dia (*Next Day Prediction*).

- **Dados:** Histórico diário (Open, High, Low, Close, Volume).
- **Desafio:** Capturar dependências temporais em séries financeiras não-lineares.
- **Abordagem:** Comparar três arquiteturas modernas de Deep Learning:
  1. **CNN 1D** (Convolutional Neural Networks)
  2. **LSTM** (Long Short-Term Memory)
  3. **Transformer** (Attention Mechanism)

# Trabalhos Relacionados

## Advanced Stock Market Prediction Using Long Short-Term Memory Networks: A Comprehensive Deep Learning Framework (2025)

**Autor:** Rajneesh Chaudhary (Supervisão: Dr. Arun Kumar)
**Instituição:** Indian Institute of Information Technology and Management (IIITM), Gwalior
**Fonte:** arXiv:2505.05325v1 [cs.CE] (Maio 2025)

## Stock Market Prediction Using Machine Learning (2018)

**Autores:** Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena et al.
**Instituição:** National Institute Of Technology (NIT), Hamirpur
**Conferência:** 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)

*Estes trabalhos fundamentam a escolha das arquiteturas recorrentes e o pré-processamento de dados financeiros utilizados neste projeto.*

# Divisão Temporal dos Dados

O conjunto de dados foi dividido estritamente por tempo para evitar *data leakage* (vazamento de dados).

| **Conjunto** | **Início** | **Fim** | **Propósito** |
|---|---|---|---|
| Treino | 01/01/2015 | 31/12/2021 | Aprendizado dos pesos |
| Validação | 01/01/2022 | 31/12/2022 | Ajuste de Hiperparâmetros |
| Teste | 01/01/2023 | 31/12/2024 | Avaliação Final (Inédita) |

# Pré-processamento e Janelamento

**Normalização:**

- Uso de `MinMaxScaler`.
- Ajustado apenas no Treino, aplicado na Validação/Teste.

**Sliding Window:**

- Janelas testadas: $[10, 20, 30, 40, 50]$ dias.
- Target: Preço de fechamento do dia $t + 1$.[9]

```
1 def create_windowed_data(data, window,
      horizon):
2     X, y = [], []
3     # ... loop ...
4     window = data[i : i + window]
5     target = data[i + window + horizon
      - 1, 3]
6     X.append(window)
7     y.append(target)
8     return np.array(X), np.array(y)
9
```

Snippet de Janelamento

# 1. TimeSeries CNN

Focada em extrair padrões locais e invariantes no tempo.

## Estrutura

- **Entrada:** (Batch, Features, Sequence Length)
- **Camadas:** 2 Blocos de Convolução 1D + ReLU + MaxPool.
- **Regularização:** Dropout ($p = 0.1$).
- **Head:** Camadas Lineares para regressão final.

*Espera-se que seja capaz de capturar tendências de curto prazo.*

# 2. TimeSeries LSTM

Rede Recorrente desenhada para evitar o problema do *Exploding/Vanishing Gradient*.

## Estrutura

- **Camadas Recorrentes:** 2 Camadas LSTM empilhadas.
- **Dimensões Ocultas:** 64 $\rightarrow$ 32 neurônios.
- **Intermediário:** Camada densa com ReLU entre as LSTMs.
- **Saída:** Apenas o último estado oculto é usado para previsão.

*Espera-se que seja capaz de capturar dependências sequenciais de longo prazo.*

# 3. TimeSeries Transformer

Uso de mecanismos de atenção (Self-Attention) para ponderar a importância de cada passo de tempo.

## Componentes Chave

- **Positional Encoding:** Adiciona informação de ordem temporal (seno/cosseno).
- **Encoder Layer:** Multi-head Attention ($n = 4$) + Feed Forward.
- **Output:** Flatten $\rightarrow$ Linear Decoder.

*Espera-se que seja capaz de capturar equilibrar tendências de curto e longo prazo.*

# Hiperparâmetros e Treinamento

- **Hardware:** GPU (CUDA) se disponível, caso contrário CPU.
- **Otimizador:** Adam (`lr=0.001`).
- **Função de Perda:** MSE Loss (Mean Squared Error).
- **Epochs:** 20 por configuração.
- **Batch Size:** 32.

**Critério de Seleção:** O modelo "Vencedor"é aquele que apresentar o menor **MAPE** (Mean Absolute Percentage Error) no conjunto de *Validação*.

# Figuras de Mérito

Para garantir a interpretabilidade, os dados são desnormalizados (`inverse_transform`) antes do cálculo:

## RMSE (Root Mean Squared Error)

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Penaliza erros grandes (outliers).

## MAPE (Mean Absolute Percentage Error)

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Erro percentual médio, mais fácil de interpretar.

# Pipeline de Seleção Automática

O código executa um *Grid Search* simplificado:

1. Itera sobre janelas: 10, 20, 30, 40, 50.
2. Para cada janela, treina CNN, LSTM e Transformer.
3. Calcula MAPE de Validação.
4. Armazena o melhor modelo.

## Resultado Final (Output)

O script imprime o modelo vencedor e seu desempenho final no conjunto de **Teste** (2023-2024), garantindo uma estimativa realista da performance em produção.

# Resultados Consolidados (Iterações = 5)

**Tabela Final de Resultados (MAPE)**

| Modelo | Tamanho da Janela | Figuras de Mérito (MAPE) |
|---|---|---|
| Baseline | - | 2.65% |
| **CNN** | **10** | **1.06% ± 0.17%** |
| Transformer | 10 | 2.42% ± 0.25% |
| LSTM | 10 | 4.41% ± 0.65% |

## Comentários e Análise

- **CNN (Vencedor):** O erro de **1.06%** demonstra que a convolução capturou eficientemente padrões locais e tendências de curto prazo nos preços.
- **Transformer vs Baseline:** O Transformer teve desempenho marginalmente melhor que o Baseline (2.42% vs 2.65%), sugerindo que o mecanismo de atenção precisa de mais dados ou janelas maiores.

# Próximos Passos

- **Visualização:** Plotar Gráfico "Real vs Previsto"do modelo vencedor.
- **Feature Engineering:** Adicionar indicadores técnicos (RSI, MACD) além dos dados brutos.
- **Tuning:** Aumentar número de epochs e implementar *Early Stopping*.
- **Backtesting:** Simular uma estratégia de trading baseada nas previsões.

# Obrigado!

Dúvidas?