Relatório 2 - Redes neurais do tipo LSTM para detecção de pontos de mudança em séries temporais de alta frequência

ALUISIO AMORIM C.[1] and MENDONCA, J. RICARDO G. *[1]

[1]EACH - USP

Setembro 2024

Resumo

Este relatório documenta o desenvolvimento e a implementação de um modelo de rede neural do tipo LSTM para a previsão em tempo real de preços de ações no mercado financeiro brasileiro. O principal objetivo, em relação a primeira etapa do trabalho, foi aumentar a frequência da série temporal, com o intuito de permitir a detecção de pontos de mudança em tempo teal. Para isso, o modelo foi integrado com as APIs do Yahoo Finance para obter dados intradiários, e o MongoDB foi utilizado para o armazenamento eficiente desses dados. Além disso, foi implementada uma estratégia de retreinamento em tempo real, que permite ao modelo atualizar-se automaticamente à medida que novos dados são recebidos. Contudo, os resultados indicam que, embora o modelo se ajuste bem aos dados históricos, ele apresenta limitações ao prever pontos de mudança abrupta, replicando movimentos passados em vez de antecipá-los com precisão. Essas observações sugerem que os hiperparâmetros e a estratégia de retreinamento contínuo não são os mais adequados para lidar com séries financeiras, que são não estacionárias. Para mitigar esses problemas, futuras melhorias poderão incluir um retreinamento completo periódico e a

*Orientador

inclusão de indicadores financeiros padrão no modelo. Todo o código produzido está disponível no repositório: https://github.com/Amorim33/lastim-cast. Recomendamos a consulta a este repositório no GitHub para uma visão detalhada do progresso do projeto e acesso ao código-fonte.

Abstract

This report documents the development and implementation of an LSTM neural network model for real-time stock price prediction in the Brazilian financial market. The main objective, in relation to the first stage of the project, was to increase the time series frequency with the aim of enabling real-time detection of change points. To achieve this, the model was integrated with Yahoo Finance APIs to obtain intraday data, and MongoDB was used for efficient data storage. Additionally, a real-time retraining strategy was implemented, allowing the model to automatically update as new data is received. However, the results indicate that, although the model fits well with historical data, it has limitations in predicting abrupt change points, replicating past movements instead of accurately anticipating them. These observations suggest that the hyperparameters and the continuous retraining strategy are not the most suitable for dealing with financial series, which are non-stationary. To mitigate these issues, future improvements could include periodic full retraining and the inclusion of standard financial indicators in the model. All the code produced is available in the repository: https://github.com/Amorim33/lastim-cast. We recommend consulting this GitHub repository for a detailed view of the project's progress and access to the source code.

1. Introdução

Um preditor em tempo real é um sistema ou modelo de previsão capaz de analisar dados em tempo real e fornecer previsões instantâneas ou com um atraso mínimo. Ele recebe dados contínuos ou frequentes de uma fonte e gera previsões ou *insights* imediatamente à medida que os dados são processados. No mercado financeiro, a previsão de preços de ações representa um desafio essencial, visto que decisões de compra e venda em frações de segundo podem impactar significativamente os lucros. A habilidade de prever esses movimentos de forma precisa e em tempo real é de grande valor para *traders* e investidores.

As redes neurais do tipo LSTM, conforme discutido na primeira etapa do trabalho [1] e demonstrado por [2], mostram capacidade para prever, com razoável acurácia, os preços de ações no mercado financeiro. Essas redes são particularmente eficazes para previsões em tempo real pelos seguintes motivos:

1. **Memória de longo e curto prazo**: LSTMs capturam dependências de longo prazo, essenciais para dados sequenciais.
2. **Processamento dinâmico**: Adaptam-se a fluxos contínuos de dados, processando sequências de forma incremental.
3. **Mecanismo de *gates***: Os *gates*[(a)] controlam o fluxo de informações, mantendo relevância e estabilidade nas previsões.
4. **Atualizações contínuas**: O modelo ajusta-se rapidamente a novas entradas de dados.

No entanto, a tarefa de um preditor de preços de ações em tempo real não é trivial. A complexidade, a natureza caótica e a dinâmica das séries financeiras fazem com que os pontos se descorrelacionem[(b)] rapidamente. Além disso, fatores externos inesperados, como crises políticas, desastres naturais e eventos econômicos globais, podem gerar pontos de mudança bruscos e imprevisíveis nos preços.

Diante desse cenário, o objetivo deste trabalho é implementar um modelo LSTM para a previsão de preços de ações em tempo real, avaliando sua acurácia e eficiência no tratamento de séries temporais de alta frequência, especificamente financeiras.

2. Materiais e Métodos

2.1. Integração com as APIs do Yahoo Finance

Previamente, utilizou-se a plataforma do **Yahoo Finance** para obter, no formato CSV, a série histórica completa do índice IBOVESPA com intervalos de vinte e quatro horas entre os

[(a)] Os *gates* são componentes internos das LSTMs que controlam o fluxo de informações, decidindo o que deve ser mantido ou esquecido em cada passo da sequência.

[(b)] Descorrelação refere-se à perda de relação ou dependência estatística entre dois ou mais pontos em uma série temporal, tornando a previsão mais difícil à medida que os padrões anteriores se tornam menos úteis para prever futuros.

pontos. Para melhorar a capacidade do modelo, foi necessário reduzir o intervalo entre os pontos e, consequentemente, aumentar a frequência da série.

O Yahoo Finance disponibiliza, também, *APIs*[(c)] gratuitas que são atualizadas continuamente. Para integração com essas APIs, foi utilizado o módulo Python *yfinance*, mantido oficialmente pelos desenvolvedores do Yahoo Finance, oferecendo uma experiência de desenvolvimento eficiente.

Apesar de ser gratuita e confiável, a API do Yahoo Finance possui algumas limitações quanto à disponibilização de dados intradiários, como:

1. **Intervalo de 1 minuto**: Séries históricas com intervalos de 1 minuto ficam disponíveis por apenas 30 dias.
2. **Intervalo de 60 minutos**: Séries históricas com intervalos de 60 minutos ficam disponíveis por 730 dias.

Devido à restrição de acesso à série completa em intervalos mais granulares, surgiu a necessidade de armazenar os pontos de dados de forma cumulativa. Para isso, foi desenvolvida uma interface de linha de comando (CLI), integrada ao módulo *yfinance*, capaz de simular um sistema em tempo real, obtendo e armazenando os dados faltantes do período de tempo disponível para a granularidade desejada. O código dessa CLI está disponível no seguinte link: https://github.com/Amorim33/lastim-cast/tree/main/get-historical-data.

2.2. Armazenamento de Dados com MongoDB

Para armazenar as séries históricas, optou-se pelo MongoDB, um banco de dados NoSQL orientado a documentos. A escolha justifica-se pela ausência de um esquema lógico fixo, permitindo avanço mais rápido em um projeto experimental. Além disso, o MongoDB oferece o tipo de coleção *TimeSeries*, que proporciona índices e restrições que melhoram significativamente a performance de leitura, a eficiência de armazenamento e o acesso ao disco para dados temporais. Mais detalhes estão disponíveis no seguinte link: https://www.mongodb.com/docs/manual/core/timeseries-collections/#benefits.

[(c)] API é a sigla para *Application Programming Interface* (Interface de Programação de Aplicações), que permite a comunicação entre diferentes sistemas.

Foi realizada a implantação de uma instância *serverless*[(d)] na AWS[(e)] utilizando o MongoDB Atlas[(f)]. Essa abordagem foi crucial, pois o modelo de precificação da instância é baseado no uso, sem limitar o tamanho do *cluster*[(g)], o que permitiu uma significativa redução de custos.

Apesar da ausência de esquema lógico fixo no MongoDB, conforme o projeto evolui, surge a necessidade de modelar e estruturar os dados. Para isso, foram criadas sete coleções.

Como mostrado na Figura 1, três coleções armazenam as séries históricas reais, enquanto outras três armazenam as séries históricas preditas. A coleção *Training*, por sua vez, é utilizada para armazenar os *checkpoints* dos treinamentos; ela possui a chave *LastTrainDatetime*, que armazena o momento do primeiro ponto da sequência mais recente presente no conjunto utilizado no último treinamento, bem como a chave *Interval*, que contém o intervalo da série, indicando qual dos três modelos foi treinado. Essa coleção é fundamental para a estratégia de retreinamento em tempo real.

2.3. Retreinamento em Tempo Real

Neste experimento, os modelos são primariamente treinados com 90% do conjunto de dados disponíveis. O objetivo é simular o comportamento de tempo real com os 10% restantes e avaliar a acurácia dos modelos. Para a construção dos modelos, foram utilizados os mesmos hiperparâmetros encontrados através da busca em grade realizada na primeira etapa do projeto.

O primeiro treinamento do modelo é realizado utilizando 90% dos dados históricos disponíveis para o intervalo desejado (1 minuto, 60 minutos ou 1 dia). Inicialmente, os dados

[(d)] O termo *serverless* refere-se a um modelo de computação em que o provedor de nuvem gerencia a infraestrutura, permitindo que o usuário se concentre apenas no código e na lógica de aplicação. Isso oferece escalabilidade automática e cobrança com base no uso efetivo dos recursos.

[(e)] Amazon Web Services (AWS) é uma plataforma de computação em nuvem fornecida pela Amazon que oferece uma ampla gama de serviços, incluindo infraestrutura como serviço (IaaS), plataforma como serviço (PaaS) e software como serviço (SaaS).

[(f)] MongoDB Atlas é uma plataforma de banco de dados como serviço (DBaaS) fornecida pela MongoDB, que oferece gerenciamento automatizado de bancos de dados MongoDB, incluindo escalabilidade automática e alta disponibilidade.

[(g)] Em um contexto de banco de dados, um *cluster* refere-se a um grupo de servidores ou instâncias que trabalham juntos para fornecer alta disponibilidade e escalabilidade ao banco de dados.

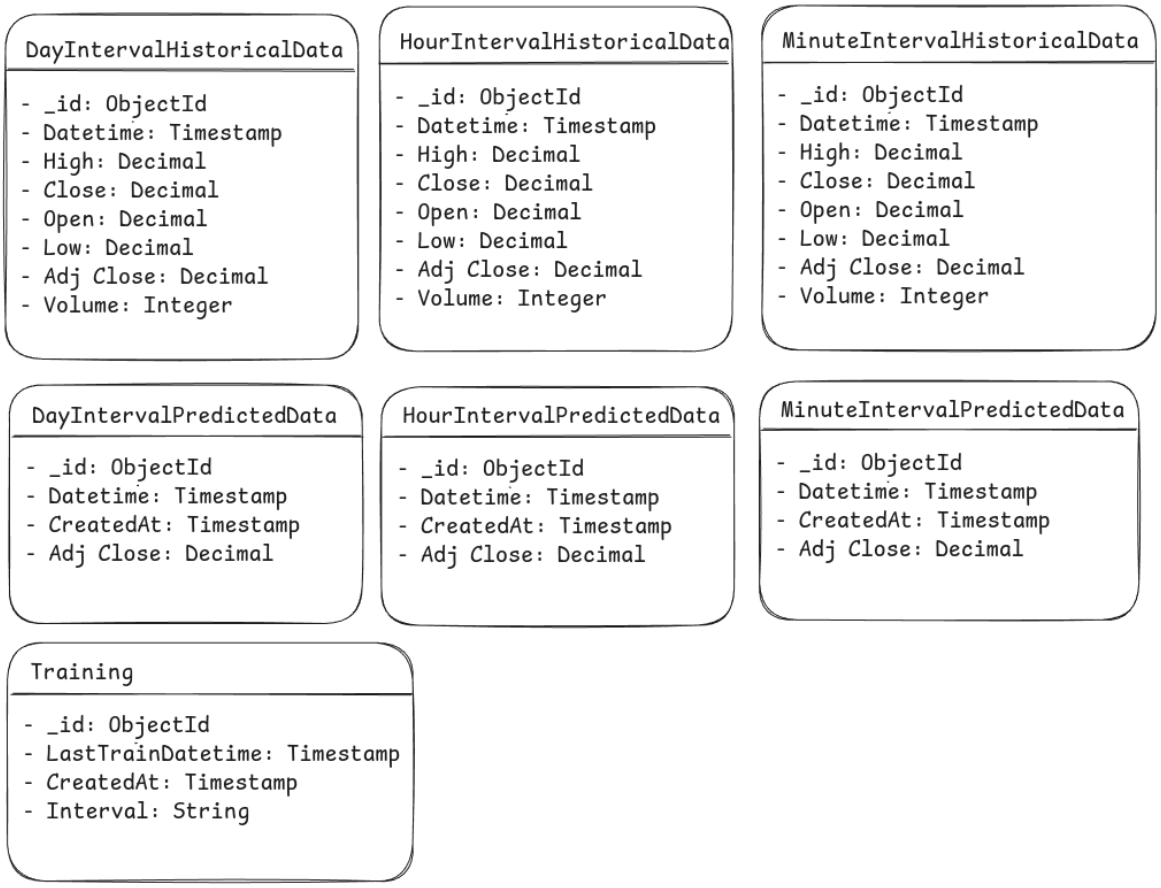

Figura 1: Modelagem dos dados.

são carregados e pré-processados, incluindo normalização e criação das sequências de entrada (*X_train*) e dos rótulos correspondentes (*y_train*). O modelo LSTM é então instanciado com os hiperparâmetros definidos, conforme mostrado na Figura 2. Em seguida, o modelo é treinado utilizando os dados de treinamento, ajustando os pesos da rede neural para minimizar a função de perda (*mean squared error*). Após o treinamento, o modelo é salvo em um arquivo com extensão `.keras`, identificado pelo intervalo de tempo correspondente (por exemplo, `lstm-bvsp-1m-model.keras`). Além disso, um registro é inserido na coleção *Training* do banco de dados, contendo informações sobre a data e hora do treinamento (*CreatedAt*), o intervalo da série (*Interval*) e o timestamp da última sequência utilizada (*LastTrainDatetime*).

Após o primeiro treinamento, é possível começar a realizar predições. Para facilitar a operação e reforçar a simulação do sistema em tempo real, foi desenvolvida uma interface de linha de comando, conforme ilustrado na Figura 4. Dois comandos principais foram criados: `make train`, que realiza o treinamento inicial, e `predict`, que gera predições enquanto houver pontos históricos reais no banco de dados que não foram incluídos no treinamento. A

```
seq_length = 10
model = tf.keras.Sequential([
    keras.layers.LSTM(units=256, input_shape=(seq_length, 1),
        return_sequences=True),
    keras.layers.LSTM(256),
    keras.layers.Dense(1)
])
model.compile(optimizer='rmsprop', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=150, batch_size=16, verbose=2)
```

Figura 2: Modelo em Python

```
last_train_datetime =
    data_points[-seq_length]['Datetime']
        if is_first_train
        else data_points[0]['Datetime']
db['Training'].insert_one({
    "CreatedAt": datetime.utcnow(),
    "Interval": interval,
    "LastTrainDatetime": last_train_datetime
})
```

Figura 3: Criação do Primerio Checkpoint de Treinamento

cada iteração, o próximo ponto da série é predito e, em seguida, a série histórica real é utilizada para retreinar o modelo. O loop se repete até que não haja mais dados históricos disponíveis. Dessa maneira, o modelo gera previsões para os novos dados disponíveis, permitindo que analisemos sua acurácia e finaliza a execução ao predizer o próximo ponto da série histórica.

```
aluisioamorim@unknown ~/C/A/l/predictor (main)> make help
Available commands:
make train - Perform the initial training with 90% of the available data
make predict - Predict all next datapoints and retrain the model with the new available data
make help - Show this help message
```

Figura 4: Comandos da CLI do preditor.

A ideia é que o modelo utilize todos os dados disponíveis até o momento para tentar predizer o próximo ponto, indicando se o preço de fechamento ajustado da ação irá aumentar ou diminuir.

```
while not has_finished_predicting(interval):
    retrain_model(interval, epochs=1, batch_size=1,
                                    is_first_train=False)
```

Figura 5: Loop de Retreinamento

O código apresentado na Figura 5 representa o loop que coordena o retreinamento incremental do modelo. Enquanto a função `has_finished_predicting(interval)` retornar `False`, indicando que ainda há predições a serem feitas, o modelo é retreinado utilizando a função `retrain_model()`. Nesta função, o modelo é atualizado com novos dados em pequenos incrementos (com `epochs=1` e `batch_size=1`), permitindo que ele se adapte rapidamente às mudanças nos dados sem a necessidade de um retreinamento completo. A definição das funções pode ser consultada no seguinte arquivo de código: https://github.com/Amorim33/lastim-cast/blob/main/predictor/lib/model.py

2.4. Interface Gráfica

Para facilitar a visualização dos resultados e proporcionar uma interface amigável ao usuário, foi desenvolvida uma aplicação web para exibir os dados reais e preditos. A interface gráfica foi construída utilizando o framework Remix, que permite a criação de aplicações web

modernas e eficientes em React. Para a renderização dos gráficos, a biblioteca Highcharts foi utilizada, conhecida por sua capacidade de criar gráficos interativos e de alta performance. A aplicação web consome os dados armazenados no MongoDB, atualizando os gráficos em tempo real à medida que novas previsões são realizadas. Essa solução permite que usuários acompanhem, de forma dinâmica, o desempenho do modelo e as tendências previstas pelo sistema.

A aplicação web foi implantada em um servidor acessível publicamente, com integração direta ao banco de dados. Essa integração permite que a aplicação acesse os dados de forma eficiente e segura, utilizando credenciais de acesso controladas. A interface gráfica está disponível no seguinte link: https://lastim-cast.vercel.app/.

Figura 6: Interface Gráfica.

Conforme ilustrado na Figura 6, o projeto foi denominado ”LaSTiM Cast”, uma combinação do acrônimo LSTM com a palavra *Forecast*. A interface também permite acessar os gráficos correspondentes aos três intervalos de tempo disponíveis. O código da aplicação pode ser consultado no seguinte link: https://github.com/Amorim33/lastim-cast/tree/main/lastim-cast-web

3. Resultados

Nesta seção, são apresentados os resultados obtidos a partir do modelo LSTM treinado para a previsão de preços no índice IBOVESPA. Os gráficos a seguir ilustram as previsões realizadas

para os diferentes intervalos de tempo estudados. A curva `azul` representa a série histórica real, enquanto a curva `roxa` corresponde à série histórica predita pelos modelos.

3.1. Previsões diárias

O Gráfico 7 apresenta as previsões do modelo com intervalo diário, utilizando o histórico do índice IBOVESPA desde 1993 até a data atual. Nota-se que o modelo conseguiu capturar as tendências gerais do índice ao longo do tempo.

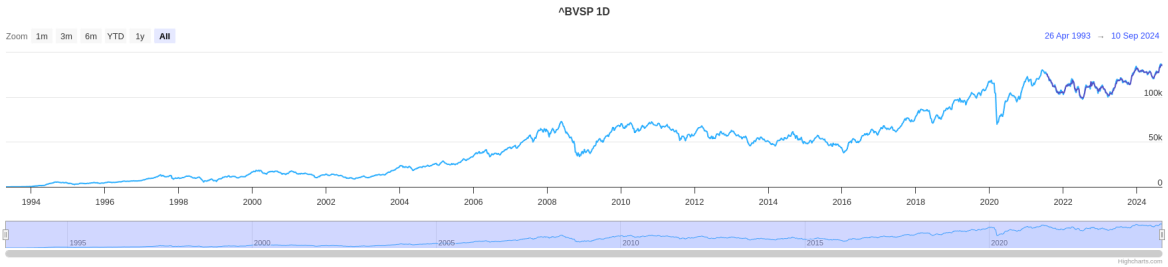

Figura 7: Histórico do índice IBOVESPA (1993 - 2024) com previsões diárias.

3.2. Previsões intradiárias

O Gráfico 8 apresenta as previsões intradiárias realizadas pelo modelo LSTM para um intervalo de 1 hora. Esse intervalo mais curto possibilita capturar flutuações de preço dentro de um único dia de negociação.

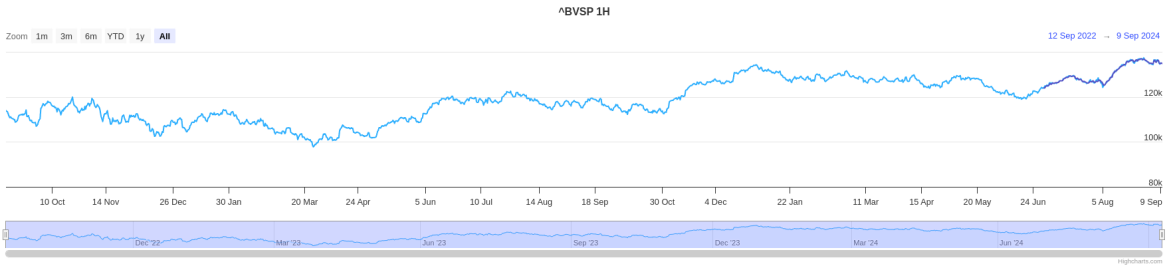

Figura 8: Previsões intradiárias, intervalo de 1 hora.

3.3. Previsões em Tempo Real

No Gráfico 9, é exibida a previsão realizada para um intervalo de 1 minuto, simulando o comportamento do modelo em tempo real. Este intervalo permite observar a precisão do modelo na captação de variações mais sutis e rápidas nos preços. Por sua vez, a

Figura 10 oferece uma visão ampliada das previsões, permitindo uma análise mais detalhada da comparação entre as séries reais e preditas pelo modelo. A visualização do comportamento nos minutos subsequentes reforça a capacidade do modelo de ajustar-se rapidamente às mudanças de curto prazo.

Figura 9: Previsão com intervalo de 1 minuto para o índice IBOVESPA.

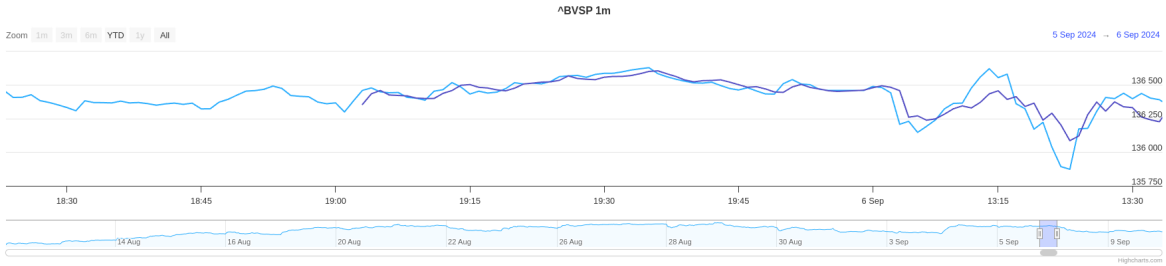

Figura 10: Visão Ampliada da Previsão com intervalo de 1 minuto para o índice IBOVESPA.

3.4. Previsão do Próximo Ponto

Por fim, a Figura 11 apresenta a previsão do próximo ponto baseado em todos os dados históricos disponíveis. É importante notar que o horário marcado no ponto predito não reflete o horário de negociação correto, pois corresponde a um período em que o mercado financeiro brasileiro já está fechado. Esse *timestamp*[(h)] deve ser interpretado como o primeiro horário útil do dia de negociação seguinte.

4. Discussão e Conclusões

Ao analisar os resultados obtidos, observou-se que, apesar de o modelo LSTM se ajustar adequadamente aos pontos da série histórica, ele não foi capaz de prever com boa acurácia os pontos de mudança abrupta. Ao examinar os gráficos gerados, verifica-se que o modelo parece

[(h)] O termo *timestamp* refere-se ao registro exato de tempo em que um evento ou transação ocorreu.

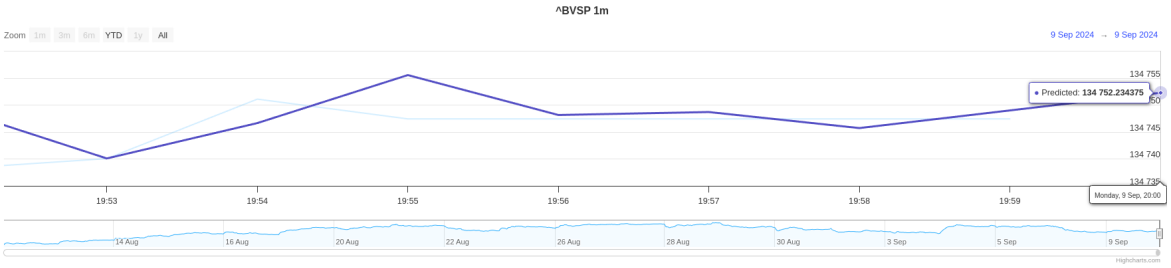

Figura 11: Previsão do próximo ponto do índice IBOVESPA.

apenas replicar as mudanças que ocorreram nos dados passados. Isso pode ser notado pelo fato de que, se deslocarmos a curva predita alguns momentos para a esquerda, ela quase se sobrepõe à curva real em todos os três intervalos de tempo (diário, intradiário e de 1 minuto). Este comportamento sugere que o conjunto de hiperparâmetros escolhido, apesar de eficiente na captura de tendências gerais, não foi capaz de ajustar o modelo para prever as viradas bruscas de tendência, as chamadas ”quebras estruturais”na série temporal.

Além disso, a estratégia de retreinamento em tempo real, apesar de promissora em séries temporais estacionárias, não se mostrou adequada para séries não estacionárias, como as financeiras. Uma série temporal estacionária é caracterizada por ter média, variância e autocorrelação constantes ao longo do tempo, o que significa que seu comportamento passado pode ser usado com mais confiança para prever o futuro. No entanto, séries financeiras são, por natureza, não estacionárias: os dados são influenciados por uma ampla gama de fatores externos (como eventos econômicos, políticos e sociais) que resultam em mudanças contínuas no ”motor”que gera os pontos. Com isso, o modelo LSTM, mesmo com uma estratégia de retreinamento contínuo de uma época conforme novos dados são recebidos, não consegue alterar significativamente seus pesos para acompanhar essas mudanças constantes. Esse fenômeno faz com que o modelo rapidamente perca a capacidade de predizer o comportamento da série com acurácia razoável, já que ele se ajusta apenas a padrões passados que já não estão em vigor.

Como próximos passos, é essencial considerar uma abordagem mais robusta de retreinamento. Uma possível solução seria implementar uma estratégia de retreinamento completo da rede após um certo período de tempo, como a cada três horas, por exemplo, para mitigar o problema da perda de acurácia. Este retreinamento mais abrangente garantiria que o modelo se recondicionasse aos novos padrões de dados que surgem no mercado financeiro.

Adicionalmente, sugere-se a inclusão de indicadores de mercado padrão como variáveis

de entrada no modelo, além dos dados históricos. Indicadores como médias móveis, índice de força relativa (RSI), volatilidade e outros podem fornecer mais informações ao modelo, ajudando-o a capturar não apenas os movimentos passados dos preços, mas também padrões mais complexos que podem indicar mudanças futuras, como visto em [2].

Caso essas melhorias resultem em uma performance mais satisfatória do modelo, a próxima etapa será a implantação do sistema em produção utilizando uma infraestrutura em nuvem. A utilização de soluções em *cloud* proporcionaria a escalabilidade necessária para processar grandes volumes de dados em tempo real, garantindo que o modelo possa ser utilizado de maneira eficiente no mercado financeiro.

Referências

[1] Aluisio Amorim C. J. Ricardo G. Mendonça. Relatório 1 - redes neurais do tipo lstm para detecção de pontos de mudança em séries temporais de alta frequência. Iniciação Científica, 2024.

[2] David M. Q. Nelson, Adriano C. M. Pereira, and Renato A. de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1419–1426, 2017.