A Comparative Analysis of Deep Learning and Classical Regression Algorithms for Predicting Bicycle Rentals

Johnattan D. F. Viana
CIn - Centro de Informática
UFPE
Recife, Brasil
jdfv@cin.ufpe.br

Krsna M. A. Rodrigues
CIn - Centro de Informática
UFPE
Recife, Brasil
kmar@cin.ufpe.br

***Resumo*—Predicting rental demand accurately is crucial for optimizing resource allocation and strategic bicycle redistribution in bike-sharing systems. In this sense, this study investigates whether deep learning models can more accurately predict bicycle rentals compared to traditional regression algorithms. Using data from Capital BikeShare and meteorological information, we compared various deep learning models with classical regressors. We applied Artificial Neural Network (ANN), Long Short-Term Memory (LSTM), Transformer, TabNet, and Convolutional Neural Network (CNN). We introduce two new error metrics and analyze CNN model's sensitivity to various hyperparameters. The goal is to provide a thorough comparison of deep learning and classical regression methods for optimizing resource management in bicycle-sharing systems. Our results show CNN model performs similarly to the RFR model, and even models designed for tabular data, like TabNet, did not surpass classical regression models for this dataset. Source code is provided[1].**

***Index Terms*—Deep Learning, Prediction, Forecasting**

I. Introdução

Os Sistemas de Bicicletas Compartilhadas (SBC) promovem um transporte sustentável compartilhado por demanda. Estes sistemas consistem em diversas estações de aluguel distribuídas por cidades, cada uma com um número limitado de bicicletas disponíveis. Nestas estações, os usuários podem alugar uma bicicleta por um período específico, retirando-a de uma estação e devolvendo-a na estação de destino. A conveniência desses sistemas atrai novos ciclistas, pois elimina a necessidade de manutenção e preocupações com estacionamento, além de contribuir para a redução do uso de carros em viagens curtas [1]. Além dos benefícios práticos dos SBC, os dados gerados por esses sistemas os tornam atrativos para pesquisas, uma vez que são registradas variáveis como horários e localizações do início e fim do trajeto.

O rebalancemanto de bicicletas (*Bike Sharing Rebalancing Problem* - BRP) é um problema que os gestores desses sistemas precisam enfrentar [2]. Dado que a demanda de bicicletas é incerta e se altera frequentemente, as ações de rebalanceamento ainda são desafiadoras nos SBC [3].

[1]Código fonte disponível no GitHub e Google Drive.

O BRP requer estratégias eficientes de redistribuição, que dependem da modelagem e previsão da demanda de aluguéis. Esse tipo de estimativa é fundamental para os provedores desse serviço, uma vez que com essas informações eles conseguem organizar os recursos (bicicletas) para compartilhamento, identificando qual a demanda. Essa administração estratégica otimiza a alocação dos recursos do SBC, aumentando a eficiência, visibilidade e transparência [4].

O estado da arte na predição de demanda nos SBC utiliza registros históricos aliados às variáveis ambientais (por exemplo, temperatura) para prever a demanda futura [5]. A previsão de demanda tem ganhado cada vez mais atenção com o rápido desenvolvimento e implantação dos SBC e estudos com esse propósito foram aplicados em diversos SBC na Califórnia [6], Seoul [7], Washington [8], Nova York [9], Chicago [5], Londres [3] e São Francisco [10].

Na previsão de demanda de aluguéis, especificamente nos SBC, o objetivo não é meramente obter valores específicos de predição, mas otimizar o sistema para agendar o rebalanceamento de bicicletas compartilhadas [11].

[8] analisou um conjunto de dados de sistemas de compartilhamento de bicicletas enriquecido com informações meteorológicas e sazonais. O estudo identificou padrões de atividade dos ciclistas relacionados a data e clima, além de determinar os parâmetros que influenciam o fluxo de aluguel de bicicletas. Utilizando o algoritmo regressor Random Forest (RFR), foram criados modelos de regressão com Coeficiente de Determinação (R^2) de quase 95%.

Este projeto é um trabalho futuro apresentado por [8], que utilizou modelos clássicos de Aprendizagem de Máquina (ML) para predição de demanda. Diferentemente, o objetivo deste estudo é avaliar o desempenho de diversos modelos de aprendizagem profunda, incluindo Rede Neural Artificial Clássica (RNA) [12], Long Short-Term Memory (LSTM) [13], Transformer [14], TabNet [15], e Rede Neural Convolucional (CNN) [16]. O intuito é responder à seguinte pergunta de pesquisa:

A aprendizagem profunda supera o desempenho alcançado por regressores clássicos na estimação da quantidade de

aluguéis de bicicletas?

Para tal, os objetivos específicos deste trabalho são:

- Construir modelos de aprendizagem profunda e compará-los com regressores clássicos;
- Propor novas métricas de erro para avaliar os modelos;
- Investigar a sensibilidade da CNN, explorando o impacto dos hiperparâmetros.

II. Metodologia

A. Base de Dados

Serão utilizados os dados do Capital BikeShare (CBS)[2], o maior sistema de compartilhamento de bicicletas nos Estados Unidos, que disponibiliza os registros de aluguéis de maneira contínua e periódica[3]. A base de dados utilizada nesta pesquisa é o fornecido por [8], referente aos aluguéis do ano de 2019, e já pré-processada. A base não possui valores faltosos.

Os registros são agrupados por hora, com 8737 registros e 13 atributos. 35,4% dos aluguéis foram realizados por usuários casuais e 64,6% por usuários cadastrados. As informações de clima foram retiradas do site Freemeteo[4]. A variável *qtd* é a variável classe, representando a totalidade de aluguéis que aconteceram em cada hora. Para inserção das variáveis climáticas nos registros agrupados, considerou-se a média das medidas climáticas naquela hora.

Tabela I: Atributos da base de dados.

Nome	Descrição	Tipo
hour	Hora do dia	Numérico
day	Dia do mês	Numérico
month	Mês	Numérico
workday	Dia útil (1 se verdadeiro)	Binário
weekday	Dia da semana	Numérico
season	Estação do ano	Categórico
temperature	Temperatura	Numérico
r_temperature	Sensação térmica	Numérico
wind	Velocidade do Vento	Numérico
humidity	Umidade do ar	Numérico
dew_point	Ponto de condensação da água	Numérico
pressure	Pressão atmosférica	Numérico
qtd	Quantidade de aluguéis realizados	Numérico

As variáveis *season* (categórica) e *workday* (binária) foram convertidas para numéricas usando o `LabelEnconder` [17]. Após isso, todas as variáveis foram normalizadas usando o `StandardScaler`. Para ambos os procedimentos, utilizou-se a biblioteca Scikit-learn [17].

Realizou-se a divisão dos dados de maneira aleatória em treinamento (80%) e teste (20%).

B. Métricas de avaliação

Para avaliar a qualidade dos modelos gerados será usado o R^2. Ele corresponde a uma medida de ajuste de um modelo estatístico linear generalizado em relação aos valores observados, variando de 0 a 1 [18]. Essa métrica representa o quanto o modelo pode explicar os valores observados.

[2]https://capitalbikeshare.com/system-data

[3]https://s3.amazonaws.com/capitalbikeshare-data/index.html

[4]freemeteo.com

Quanto maior esta métrica, mais explicativo será o modelo, descrevendo melhor as amostras. Esse coeficiente é definido pela Equação 1, onde $\hat{y}_i$ é uma estimativa, y_i é a observação real, $\bar{y}$ é a média das estimativas e n é o número de pontos do experimento [18].

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{1}$$

Também serão usadas as métricas de Erro Médio Absoluto (MAE) e Raiz do Erro Quadrático Médio (RMSE), que medem, respectivamente, a magnitude média dos erros nas previsões, sem considerar a direção (positiva ou negativa) dos erros; e a magnitude média dos erros, penalizando erros maiores de forma mais significativa. As métricas são usadas para avaliar a performance dos modelos como segue:

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^{n} |y_i - \hat{y}_i| \tag{2}$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{3}$$

onde y_i representam o valor real do i-ésimo dado de teste, $\hat{y}_i$ representam o valor predito, e n representam o número total de exemplos no conjunto de teste.

Tanto o MAE quanto o RMSE expressam o erro médio do modelo preditivo. Eles podem variar de 0 a ∞ e são indiferentes às direções de erro. Portanto, por representarem taxas de erro, valores menores correspondem a modelos melhores [19].

Se considerarmos que, para um sistema de aluguel, é preferível que o modelo preveja um número maior de aluguéis do que realmente ocorrem (ou seja, é melhor errar para cima), então usar uma métrica que penalize mais os erros negativos (aluguel previsto inferior ao real) pode ser apropriado. Então, sugerimos também as métricas MAE* (Equação 4) e RMSE* (Equação 5) ponderadas:

$$\text{MAE}^* = \text{MAE} \cdot p \tag{4}$$

$$\text{RMSE}^* = \text{RMSE} \cdot p \tag{5}$$

Aplicou-se um fator de penalidade p que dá mais peso aos erros de previsão que estão abaixo do valor real. Nos experimentos, considerou-se $\alpha = 1,5$.

$$\begin{cases} p = 1,0 & \text{se } \hat{y}_i \geq y_i \\ p = \alpha & \text{se } \hat{y}_i < y_i \end{cases}$$

C. Regressores Clássicos

Os algoritmos clássicos de regressão foram implementados e executados, de acordo como descrito em [8]. Na Figura 1 é mostrada a métrica R^2 de cada algoritmo de regressão. Foram utilizadas as configurações padrões dos hiperparâmetros [17].

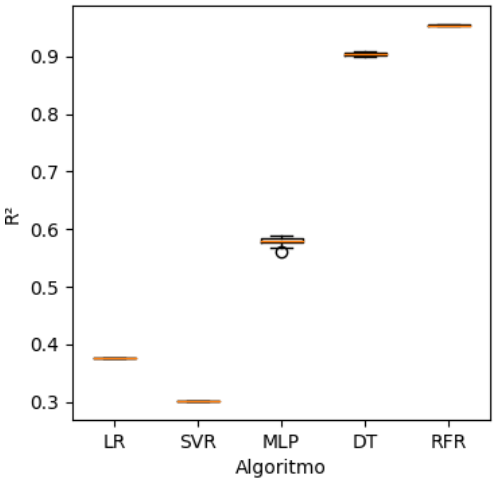


Figura 1: R² dos regressores clássicos

D. Aprendizagem Profunda

Será avaliado o desempenho dos seguintes modelos de aprendizagem profunda: Rede Neural Artificial Clássica (RNA); *Long Short-Term memory* (LSTM); Transformer; TabNet [15]; e Rede Neural Convolucional (CNN). O Tensorflow foi usado para implementação dos modelos, que foram treinados por 50 épocas, com *bach* de tamanho 8.

A RNA foi estruturada como uma arquitetura feedforward densamente conectada, foi construída a estrutura de camadas 16-16-8-4-1. Todos os neurônios da camada de entrada e intermediárias usam a função de ativação ReLU.

O LSTM foi desenvolvido com duas camadas: 1 camada LSTM (128 neurônios, passo temporal = 1), eficaz para capturar dependências de longa distância em dados sequenciais, e uma camada Densa totalmente conectada composta de 1 neurônio, sendo essa a camada de saída.

O Transformer foi construído com 1 bloco embedding para extração de features com a seguinte configuração: 1 camada densa (64 neurônios), 1 Normalization e 1 de Reshape para compatibilizar com a camada Multi-Head Attention. Após esse bloco de embedding há 24 blocos transformer, cada um com as seguintes camadas sequenciais: 1 Multi-Head Attentiom (24 heads), 1 Dropout (0.2), 1 Normalization (épsilon=1e-6), 1 camada Add para operação de adição, 1 camada Densa (64 neurônios), 1 Dropout (0.2), 1 camada Densa (64 neurônios), 1 Normalization (épsilon=1e-6), 1 camada Add para operação de adição. Por fim, há uma camada Flatten e a camada Densa de saída (1 neurônio, activation='linear').

Para o TabNet foi utilizado o modelo nativo do pyTorch, o TabNetRegressor. A configuração do modelo foi composta de máscaras do tipo 'sparsemax' (16 camadas densas nas máscaras de decisão e 16 camadas densas nas máscaras de atenção), 2 camadas densas específicas para cada decisão, 2 camadas densas compartilhadas entre todas as decisões, Decoder (1 camada densa compartilhada e 1 camada densa independente) e otimizador Adam (lr=2e-2). O modelo TabNetRegressor possui para o tamanho de lote o valor padrão igual a 1024, sendo este utilizado para a sua modelagem neste trabalho, visto que valores inferiores geram gradativa instabilidade do modelo, notadamente quando seu valor é reduzido à casa das dezenas.

A CNN foi construída com:

- 2x Convolucional 1D com 32 filtros
- 2x Convolucional 1D com 64 filtros
- 2x Convolucional 1D com 128 filtros
- MaxPolling
- 2x Convolucional 1D com 256 filtros
- 1x Convolucional 1D com 528 filtros
- MaxPolling
- Achatamento
- Full-Connected com 528 unidades
- Full-Connected com 256 unidades
- Full-Connected com 64 unidades
- 2 x Full-Connected com 32 unidades
- 2x Full-Connected com 16 unidades
- Full-Connected com 1 saída

Todas as camadas de convolução usaram a função de ativação ReLU, e todas as camadas densamente conectadas usaram a função de ativação Swish [20].

III. Resultados e Discussões

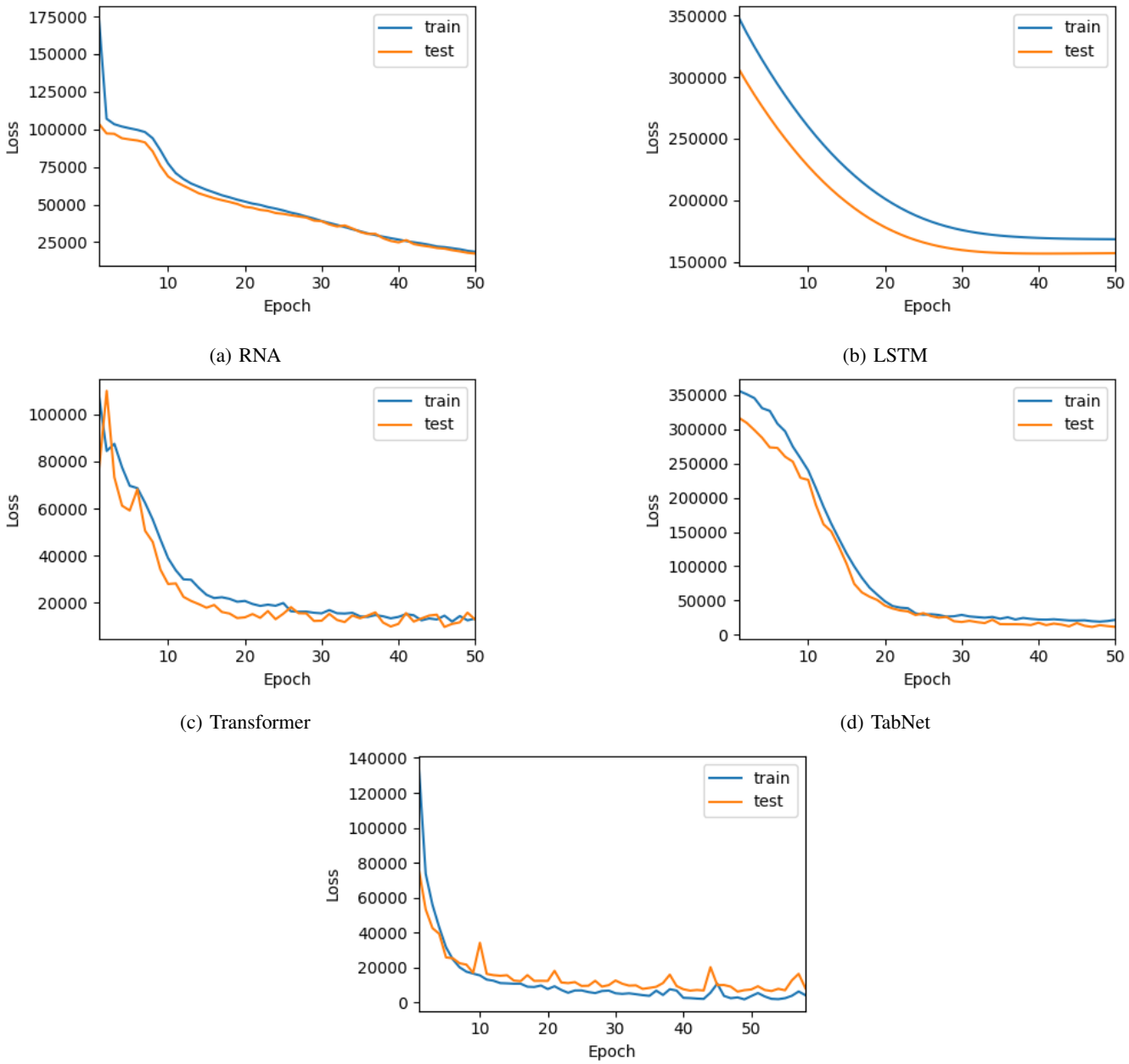
Nesta seção, primeiramente será apresentado o baseline alcançado com os regressores clássicos (Seção III-A). Depois o desempenho das abordagens que utilizam aprendizagem profunda é apresentado (Seção III-B). Na Seção III-C, a arquitetura CNN é explorada, sendo aprensentada uma análise de sensibilidade e otimização de hiperparâmetros.

A. Regressores Clássicos

Para obtenção de um baseline de comparação com os modelos de aprendizagem profunda, os algoritmos clássicos de regressão foram executados. Na Figura 1 é mostrada a métrica R² de cada algoritmo de regressão. Os algoritmos baseados no ganho de informação, como RFR e DT, apresentaram melhores resultados, principalmente o RFR, que foi capaz de explicar aproximadamente 95% das demandas de aluguéis por hora por meio das variáveis independentes utilizadas nos respectivos modelos.

No RFR, o valor de MAE (Tabela II) indica que, em média, as previsões do modelo têm um desvio absoluto médio de cerca de 50 unidades em relação aos valores reais. Embora um R² alto demonstre um bom ajuste do modelo, o MAE sugere que ainda há uma margem de erro substancial nas previsões. Portanto, embora o modelo seja explicativo, suas previsões individuais podem variar consideravelmente em relação aos valores reais.

O RMSE alto indica que o modelo está mais distante dos dados reais, o que significa que está fazendo previsões menos precisas. Além disso, o MAE e RMSE ponderados indicam que esses modelos erram muito para baixo.



(a) RNA

(b) LSTM

(c) Transformer

(d) TabNet

(e) CNN

Figura 2: Curva do MSE por 50 épocas.

B. Abordagens com Aprendizagem Profunda

Os modelos de aprendizagem profunda foram aplicados para o problema de regressão apresentado (Figura 2).

Os modelos clássicos como RFR e DT são conhecidos por sua robustez e capacidade de lidar com variáveis de entrada heterogêneas e complexas. Esses modelos não requerem grandes quantidades de dados para treinamento e são menos suscetíveis a problemas de overfitting, especialmente quando bem parametrizados. No caso do RFR, a combinação de várias árvores de decisão reduz o risco de overfitting e melhora a generalização, resultando em uma performance superior. Por outro lado, os modelos de aprendizagem profunda, como LSTM, são mais complexos e exigem uma quantidade significativa de dados para treinamento eficaz.

O Transformer, apesar de sua necessidade de mais recursos computacionais e tempo para treinamento, apresentou um bom desempenho, o que pode ser atribuído à sua capacidade de capturar dependências de longo prazo nos dados, algo que é um ponto fraco em modelos mais simples.

Em resumo, os resultados destacam que modelos clássicos, como o RFR, são eficazes para o conjunto de dados específico,

Tabela II: Desempenho entre regressores clássicos (acima) e modelos de aprendizagem profunda (abaixo).

Modelo	R^2↑	MAE↓	MAE*↓	RMSE↓	RMSE*↓
LR	0,377	239,990	368,677	312,758	367,867
SVR	0,301	229,206	298,566	331,115	348,654
MLP	0,578	189,980	290,645	257,317	300,078
DT	0,909	69,572	105,769	119,594	148,788
RFR	**0,953**	**50,963**	**79,253**	**85,610**	**108,626**
RNA	0,696	141,359	196,037	218,392	244,914
LSTM	0,496	189,601	269,376	281,291	310,229
Transformer	0,918	73,205	113,079	113,427	142,038
TabNet	0,926	74,570	121,228	107.801	137.541
CNN	**0,950**	**50,280**	**71,697**	**79,063**	**93,059**

provavelmente devido à sua robustez e menor necessidade de ajuste detalhado. Por outro lado, os modelos de aprendizagem profunda podem oferecer vantagens em termos de capacidade de captura de padrões complexos, mas exigem mais dados, ajuste preciso e recursos computacionais, o que pode não ter sido totalmente atendido no contexto dos resultados apresentados.

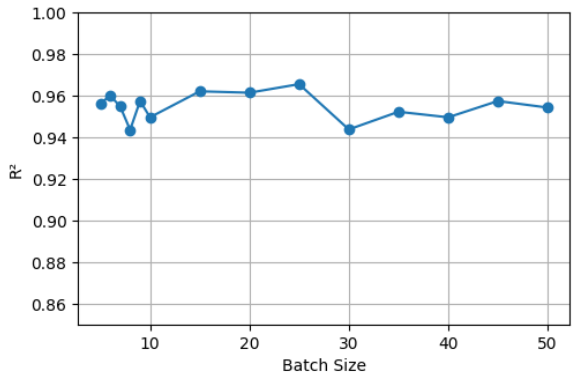
A curva do MSE durante o treinamento é mostrado ao longo de 50 épocas na Figura 2. Cada gráfico compara o desempenho no conjunto de treino (linha azul) e no conjunto de teste (linha laranja).

No RNA (Figura 2a), o MSE diminui constantemente tanto para os dados de treino quanto para os de teste. Inicialmente, o MSE é elevado, mas há uma redução gradual ao longo das épocas. A boa correspondência entre as curvas de treino e teste sugere que o modelo RNA não está superajustado, indicando um bom equilíbrio entre precisão e generalização. Na CNN (Figura 2e) a perfomance é similar à RNA, embora com um pouco mais de variabilidade no processo de treinamento.

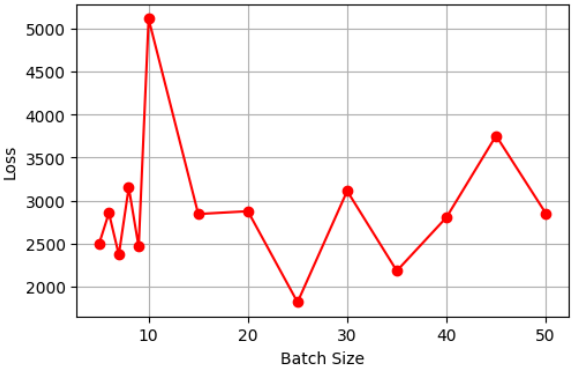
O desempenho do LSTM (Figura 2b) é retratado com uma diminuição mais acentuada do MSE nas primeiras épocas, seguido de uma estagnação ao decorrer do treinamento, demonstrando que o LSTM atinge um bom desempenho de forma rápida e estabiliza mais cedo que os outros modelos. A proximidade das curvas sugere que o LSTM tem uma boa capacidade de generalização.

O Transformer (Figura 2c) apresenta curvas de treino e teste similares ao LSTM, com pequenas flutuações no MSE ao longo das épocas. Portanto, assim como o LSTM, o transformer apresentou bom desempenho de forma rápida e estabilização mais cedo que outros modelos, bem como uma boa capacidade de generalização.

O TabNet (Figura 2d) revela curvas de aprendizado similares às do LSTM e Transformer, sobretudo por apresentar estabilização pouco depois da época 20. As baixas flutuações do MSE ao longo das épocas sugerem que o modelo TabNet possui boa estabilidade. Assim como o RNA, o TabNet apresenta boa correspondência entre as curvas de treino e teste, notadamente após a estabilização, indicando que não há overfitting e o modelo TabNet possui boa habilidade de generalização.



(a) R^2 ↑



(b) Loss ↓

Figura 3: Impacto do tamanho do lote no treinamento.

C. *Análise de Sensibilidade da CNN*

Esta subseção visa aprofundar a compreensão sobre a sensibilidade da CNN frente a diferentes configurações de hiperparâmetros, como tamanho do batch e taxa de aprendizagem.

1) Tamanho do batch: O tamanho do batch é fundamental no treinamento do modelo, influenciando tanto o tempo necessário quanto o desempenho alcançado. Lotes maiores aceleram o processo ao otimizar o uso dos recursos computacionais, porém podem gerar instabilidade. Por outro lado, lotes menores resultam em estimativas de gradiente menos precisas devido à variação nos gradientes calculados, o que pode prolongar o tempo total de treinamento e retardar a convergência.

Portanto, encontrar um equilíbrio entre eficiência computacional e estabilidade é essencial ao determinar o tamanho ideal do batch. Nesse sentido, para avaliar o impacto do tamanho do lote na CNN, o tamanho do lote de treinamento foi variado de 5 a 50, com o modelo sendo treinado por 50 épocas utilizando o otimizador 'adam' e a função de perda MSE. Após cada ciclo de treinamento, a performance do modelo foi avaliada por meio do R^2 (Figura 3a) e o último MSE foi registrado (Figura 3b).

O tamanho do lote 25 foi selecionado para os experimentos devido ao alto R^2 (0,96), indicando bom ajuste aos dados, e a função de perda baixa em relação aos demais tamanhos testados. Além disso, esse tamanho acelerou o treinamento

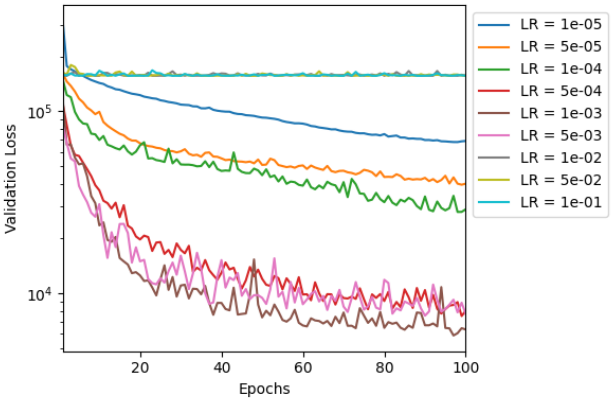


Figura 4: Impacto da taxa de treinamento na validação na CNN.

sem instabilidades significativas ou problemas de memória.

2) Taxa de aprendizagem: A Figura 4 mostra a perda na validação para diferentes taxas de aprendizado (LR) durante o treinamento do modelo, com taxas variando de 1e-05 a 1e-01. Observa-se que a convergência do modelo é influenciada pela taxa de aprendizado escolhida.

Taxas muito altas (1e-02, 5e-02 e 1e-01) fazem com que o modelo ajuste os parâmetros de forma exagerada a cada iteração, causando oscilações excessivas e impedindo a convergência. Por outro lado, taxas muito baixas resultam em ajustes tão pequenos que a perda de validação diminui lentamente (1e-05 e 5e-05), exigindo mais tempo de computação. Taxas intermediárias tendem a encontrar um equilíbrio entre a velocidade de convergência e a estabilidade do aprendizado. Para este modelo, a melhor taxa de aprendizagem foi 0,001.

D. RFR x CNN

Uma vez que uma arquitetura de CNN com bom desempenho foi identificada, esta seção compara a performance dos modelos RFR e CNN em métricas de erro, tempo de execução e uso de memória pelo modelo. Para isso, cada um dos modelos foi rodado 30 vezes e os resultados discutidos a partir daqui são as médias desses valores.

Na Figura 5, os erros são apresentados em gráficos de caixa, considerando 30 execuções para cada modelo. Observa-se que, em geral, o RFR tende a exibir erros menores e com menor variabilidade. A variância nos resultados da CNN pode ser atribuída à inicialização aleatória dos pesos do modelo durante o treinamento.

Por o RFR consistir em um conjunto de árvores de decisão cujas previsões são combinadas para produzir uma previsão final, a variabilidade dos resultados é bem pequena. Em contrapartida, a CNN é sensível à inicialização aleatória dos pesos, resultando em diferentes mínimos locais no espaço de erro e, consequentemente, maior variabilidade nos resultados.

Além disso, o RFR possui poucos hiperparâmetros e sua robustez aumenta com o número de árvores devido à média das previsões. Por outro lado, a CNN possui muitos hiperparâmetros que podem afetar significativamente seu desempenho III-C.

Considerando o tempo de execução e a memória utilizada, o RFR utiliza mais memória devido ao armazenamento de um grande número de árvores de decisão que precisam ser mantidas na memória durante a execução. Mesmo assim, o treinamento é menos demorado, uma vez que a construção das árvores não envolve cálculos complexos. Por outro lado, a CNN utiliza menos memória, já que só precisa armazenar os parâmetros da rede (pesos e viés) e os dados intermediários gerados por cada camada. No entanto, o treinamento de uma CNN é extremamente intensivo em termos de processamento. Isso se deve às inúmeras operações de convolução, pooling e, principalmente, à retropropagação, que ajusta iterativamente os pesos da rede para minimizar a função de perda. Cada iteração de treinamento envolve cálculos de derivadas e atualizações de pesos, exigindo um grande número de operações computacionais. Na Tabela III é mostrado um comparativo do tempo médio de execução (em segundos) e do consumo de memória (em MB) para ambos os modelos.

Tabela III: Comparação entre o tempo de execução dos modelos (em segundos) e o consumo de memória (em MB), após 30 execuções.

Modelo	**Tempo (segundos)**	**Memória (MB)**
CNN	243.726	133.386
RFR	2.553	189.739

Os modelos de aprendizagem profunda são altamente eficazes em tarefas que lidam com grandes volumes de dados não estruturados, como imagens, áudio e texto, devido à capacidade de capturar relações complexas entre os dados. No entanto, quando aplicados a dados tabulares tradicionais, como os analisados neste estudo, mesmo modelos avançados como TabNet, especificamente desenvolvidos para melhorar o desempenho em conjuntos de dados estruturados com características bem definidas, não demonstraram vantagens significativas em comparação com métodos como RFR.

IV. Conclusões

Modelos baseados em árvores podem ser vantajosos quando aplicados para a previsão em dados tabulares devido à sua capacidade de lidar bem com conjuntos de dados menores e de oferecer interpretabilidade. Por outro lado, modelos de aprendizagem profunda, como CNN, enfrentam desafios significativos em cenários de dados tabulares com quantidades relativamente baixas de registros. Esses modelos dependem de grandes volumes de dados para aprender eficazmente padrões complexos, ajustar uma grande quantidade de parâmetros e mitigar o risco de sobreajuste. Portanto, sua aplicação em conjuntos de dados menores pode resultar em desempenho inferior em comparação com modelos baseados em árvores, que são mais robustos em contextos de dados limitados.

Além do mais, a CNN é muito sensível aos hiperparâmetros devido à sua complexidade e à natureza interconectada de suas camadas. Hiperparâmetros relacionados ao processo de

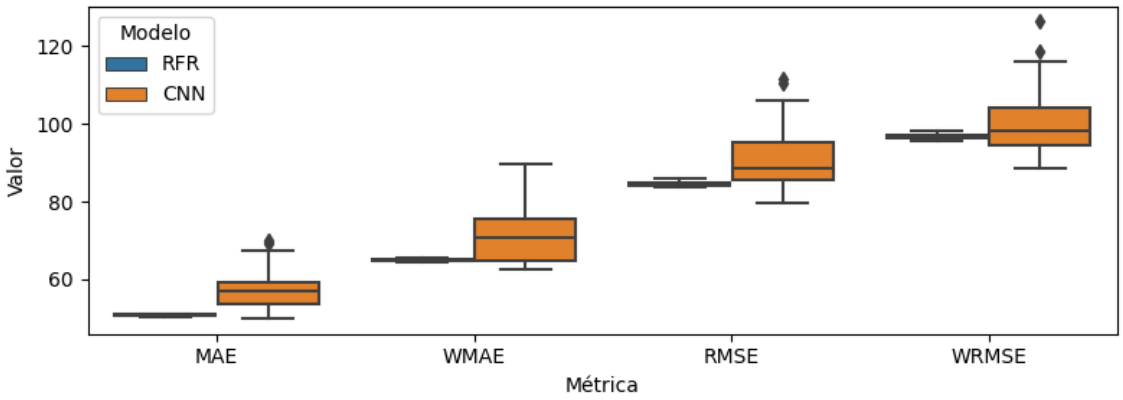


Figura 5: Métricas de erros do RFR e CNN.

treinamento, como a taxa de aprendizado, o número de épocas de treinamento, o tamanho do lote (batch size) e as técnicas de regularização (como dropout), têm um impacto crucial na convergência do modelo e na capacidade de generalização para novos dados. Pequenas variações nesses hiperparâmetros podem resultar em grandes mudanças no desempenho da CNN.

Como observado na investigação da questão de pesquisa (*"A aprendizagem profunda supera o desempenho alcançado por regressores clássicos na estimação da quantidade de aluguéis de bicicletas?"*), a CNN demonstrou desempenho comparável ao do RFR, mas com uma exigência computacional consideravelmente maior. Isso sublinha a dificuldade intrínseca das abordagens de aprendizagem profunda em lidar com dados tabulares.

Referências

[1] E. Fishman, S. Washington, and N. Haworth, "Bike share: A synthesis of the literature," *Transport Reviews*, vol. 33, no. 2, pp. 148–165, 2013. [Online]. Available: https://doi.org/10.1080/01441647.2013.775612

[2] X. Wang, H. Sun, S. Zhang, Y. Lv, and T. Li, "Bike sharing rebalancing problem with variable demand," *Physica A: Statistical Mechanics and its Applications*, vol. 591, p. 126766, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437121009559

[3] T. Xu, G. Han, X. Qi, J. Du, C. Lin, and L. Shu, "A hybrid machine learning model for demand prediction of edge-computing-based bike-sharing system using internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7345–7356, 2020.

[4] J. Liu, L. Sun, W. Chen, and H. Xiong, "Rebalancing bike sharing systems: A multi-source data smart optimization," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1005–1014. [Online]. Available: https://doi.org/10.1145/2939672.2939776

[5] Y. Yang, A. Heppenstall, A. Turner, and A. Comber, "Using graph structural information about flows to enhance short-term demand prediction in bike-sharing systems," *Computers, Environment and Urban Systems*, vol. 83, p. 101521, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0198971520302544

[6] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[7] V. E. Sathishkumar, P. Jangwoo, and C. Yongyun, "Using data mining techniques for bike sharing demand prediction in metropolitan city," *Computer Communications*, vol. 153, pp. 353–366, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419318997

[8] J. D. F. Viana, O. Braga, L. Silva, and F. M. Neto, "Analyzing patterns of a bicycle sharing system for generating rental flow predictive models," in *Anais do III Workshop de Computação Urbana*. Porto Alegre, RS, Brasil: SBC, 2019, pp. 57–70. [Online]. Available: https://sol.sbc.org.br/index.php/courb/article/view/7468

[9] Z. Yang, J. Hu, Y. Shu, P. Cheng, J. Chen, and T. Moscibroda, "Mobility modeling and prediction in bike-sharing systems," ser. MobiSys '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 165–178. [Online]. Available: https://doi.org/10.1145/2906388.2906408

[10] H. I. Ashqar, M. Elhenawy, and H. A. Rakha, "Modeling bike counts in a bike-sharing system considering the effect of weather conditions," *Case Studies on Transport Policy*, vol. 7, no. 2, pp. 261 – 268, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2213624X16301018

[11] Y. Zhang, H. Wen, F. Qiu, Z. Wang, and H. Abbas, "ibike: Intelligent public bicycle services assisted by data analytics," *Future Generation Computer Systems*, vol. 95, pp. 187 – 197, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18322787

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[15] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, pp. 6679–6687, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16826

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] B. Ait-Amir, P. Pougnet, and A. E. Hami, "6 - meta-model development," in *Embedded Mechatronic Systems 2*, A. E. Hami and P. Pougnet, Eds. Elsevier, 2015, pp. 151 – 179. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781785480140500062

[19] A. Kassambara, *Machine Learning Essentials: Practical Guide in R*. STHDA, 2018.

[20] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.