Probabilidade de uma empresa júnior / instância bater sua meta a partir da base de dados da FEJESP

1st Luiza Araujo de Oliveira Caram Saliba
Science and technology institute (UNIFESP)
São José dos Campos, Brazil
luiza.saliba@unifesp.br

2nd Harrison Caetano Candido
Science and technology institute (UNIFESP)
São José dos Campos, Brazil
h.candido20@unifesp.br

Resumo*—Este estudo visa analisar a probabilidade de empresas juniores atingirem suas metas utilizando dados fornecidos pelas principais federações de empresas juniores do Brasil, que promovem o empreendedorismo e a capacitação profissional, possuindo uma rica base de dados que pode revelar fatores determinantes para o sucesso dessas empresas. Utilizando técnicas de mineração de dados e análise estatística, este estudo emprega um modelo de rede neural artificial para a tarefa de regressão, a fim de prever a probabilidade de sucesso das empresas juniores. A metodologia inclui etapas de conhecimento de domínio, pré-processamento dos dados, extração de padrões, pós-processamento e utilização do conhecimento. Espera-se que os insights obtidos ajudem gestores a tomar decisões estratégicas, fortalecendo o movimento de empresas juniores no estado de São Paulo.

Palavras chave — redes neurais artificiais, empresa júnior, faturamento, inovação, aprendizado de máquina

I. Introdução e Motivação

A análise da probabilidade de uma empresa júnior atingir suas metas é essencial para entender os fatores que contribuem para o sucesso dessas organizações. As federações FEJESP, FEJEMG, FEJEPE, FEJECE, Concentro e RN júnior (Federações das Empresas Juniores de São Paulo, Minas Gerais, Pernambuco, Ceará, Distrito federal e Rio grande do Norte, respectivamente) possuem uma rica base de dados, com mais de 690 empresas juniores, que pode ser explorada para obter insights valiosos sobre o desempenho dessas empresas. Essas federações desempenham um papel crucial no desenvolvimento e suporte das empresas juniores, promovendo o empreendedorismo, a capacitação profissional e a integração entre estudantes e o mercado, sendo responsável por faturar mais de vinte e um milhões de reais no ano passado.

Este estudo tem como objetivo analisar os dados fornecidos pela base para calcular a probabilidade de uma empresa júnior alcançar suas metas de faturamento e, com isso, auxiliar no processo aquelas que têm probabilidades mais baixas de atingir os objetivos.

Através da utilização de técnicas de mineração de dados e ferramentas de análise estatística, será possível identificar padrões, tendências e variáveis que influenciam o sucesso das empresas juniores. Com isso, espera-se fornecer informações que possam auxiliar na tomada de decisões estratégicas, contribuindo para o fortalecimento do movimento de empresas juniores e promovendo a eficiência e eficácia na gestão dessas organizações e no sucesso das federações.

II. Conceitos Fundamentais

A. Mineração de Dados

O avanço das tecnologias de aquisição e armazenamento de dados, como a internet, levou a criação da área de ciência de dados. Definimos a mineração de dados como uma área que estuda a descoberta de padrões e conhecimento para inteligência a partir de dados estruturados ou não [1].

No processo de estudo e análise dos dados nós seguimos uma trajetória que inicia em conhecimento de domínio (coleta dos dados), Pré-processamento (limpeza, transformação, formatação, filtragem etc), Extração de padrões (Treino e teste de modelos para extração de padrões) e Pós-Processamento (Análise dos padrões descobertos nos dados) [1].

B. Aprendizado de Máquina

O aprendizado de máquina é uma linha de pesquisa da área de inteligência artificial que trata de ensinar modelos com dados de exemplos e assim melhorar a acurácia em uma aplicação alvo [2].

Segundo a noção de aprendizado indutivo, dizemos que o aprendizado é expresso por uma função objetivo f que mapeia um conjunto X (amostra de distribuição desconhecida) a um conjunto Y, sendo X um conjunto de dados que usamos para treinar e testar o modelo e Y o valor a ser previsto [2].

Nosso objetivo aqui é encontrar uma hipótese H que melhor se ajuste a função f dentre as infinitas hipóteses que temos no chamado espaço de hipóteses, que são as infinitas possibilidades de performance por configuração do nosso modelo utilizado [2].

Sempre que estamos trabalhando com Modelos de aprendizado de máquina devemos nos atentar a escolha de hipóteses que sejam consistentes (hipóteses que se ajustam a todos os dados da amostra), menos complexas (dentre as possibilidades de hipóteses consistentes, escolher a menos

complexa, seguindo a Lâmina de Ockham) e generalistas (se ajustam bem a um conjunto de treino diferente)[2].

C. *Aprendizado Supervisionado*

O aprendizado de máquina supervisionado consiste na ideia de conhecer os valores de Y (saída) e treinar e testar o modelo em X para reconhecer erros e acertos e apontar ao modelo qual resultado está mais próximo do desejado. Aqui nós temos dois tipos de saída: Y discreta (tarefa de classificação) e Y contínua (tarefa de regressão) [2].

D. *Redes Neurais Artificiais Recorrentes*

Este tipo de modelo de aprendizado supervisionado pode lidar com a tarefa de regressão, onde uma função recebe uma sequência de vetores de atributos e retorna um único valor contínuo como saída [2].

Como queremos fornecer a probabilidade de uma empresa júnior atingir seus objetivos, com uma facilidade de interpretação ao longo do tempo, usamos este modelo como base do projeto. Nas RNNs, os nós são chamados de Units (neurônios), que estão ligados por arestas chamadas Links. Cada neurônio possui uma função de ativação que permite encontrar uma hipótese que se ajusta aos dados de entrada da rede, recebendo não apenas o valor atual, mas também informações das iterações anteriores . Os dados de entrada são introduzidos pela Input Layer, onde cada atributo é multiplicado pelo peso associado a ele e ao neurônio correspondente [2].

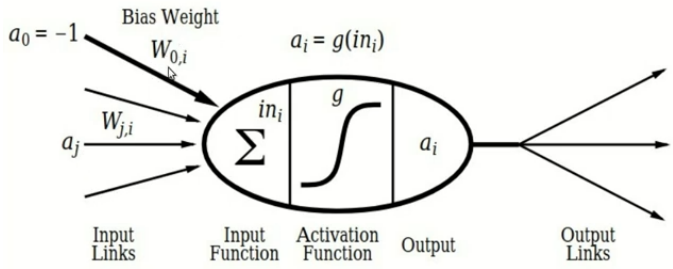

Fig 1. Estrutura de um neurônio [2].

As redes neurais recorrentes possuem, além das camadas de entrada e saída (Input e Output Layers), uma ou mais camadas ocultas (Hidden Layers), onde cada neurônio possui uma conexão com sua própria saída anterior. Esta estrutura permite que a rede aprenda dependências temporais e sequenciais nos dados. As Hidden Layers em RNNs são compostas por grafos cíclicos direcionados, onde a saída em tempo tt depende tanto do atributo de entrada em tt quanto das saídas anteriores .

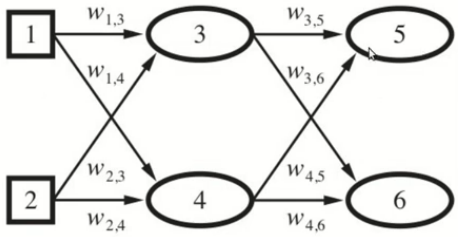

Fig 2. Multilayer Perceptron para tarefa de classificação [2].

Cada atributo de saída é calculado como a derivação da soma do produto do de um atributo de entrada com o peso associado a ele [2].

E. *Ajuste de pesos*

Um conceito essencial em redes neurais recorrentes é o ajuste de pesos utilizando o algoritmo de Backpropagation Through Time (BPTT) [2]. Neste processo, os dados são submetidos à rede através da Input Layer, e os parâmetros dos neurônios são calculados pelas funções de ativação, que propagam os resultados ao longo do tempo e das camadas [2]. Na Output Layer, o valor previsto é comparado com o valor esperado, e a partir daí o erro é calculado e utilizado pela função de perda, que propaga de volta na rede para correção dos pesos, não apenas em uma direção, mas ao longo das várias etapas temporais que a rede processa [2]. O ajuste de pesos é fundamental para que a rede tenha uma hipótese mais ajustada ao comportamento dos dados e seja menos propensa ao overfitting. As correções são feitas através da regra de atualização de pesos (Weight Update Rule) ao longo do tempo, utilizando métodos de otimização derivados do Gradient Descent [2]. A variação temporal da Weight Update Rule leva em consideração as interações ao longo das várias etapas de tempo.

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$$

Fig 3. Regra de atualização de pesos da camada de saída [2].

Sendo a taxa de erro de cada neurônio na última camada o produto do erro (diferença entre os valores reais e previstos), como na figura abaixo [2].

$$\Delta_i = Err_i \times g'(in_i)$$

Fig 4. Taxa de erro na camada de saída [2].

Enquanto a segunda parte da Weight Update Rule é feita entre a camada de entrada e a última camada oculta e utiliza a seguinte variação da Gradient Descent [2].

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j \ .$$

Fig 5. Regra de atualização de pesos da camada oculta e de entrada [2].

Sendo a taxa de erro de cada neurônio nas camadas oculta e de entrada o produto da derivada da função de ativação e o somatório do produto do peso de cada atributo com a taxa de erro de cada Unit, como na figura abaixo.

$$\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i \ .$$

Fig 6. Taxa de erro na camada oculta e de entrada [2].

Um outro conceito importante no ajuste de pesos é o de Epochs, que são nada mais do que as rodadas de treinamento da rede, então a cada Epoch com backpropagation, nós temos o treinamento feedforward, backpropagation e feedforward enquanto a perda não for mínima [2].

F. Overfitting

Para evitar o sobreajuste da hipótese, o que prejudica a capacidade de performar em conjuntos de dados desconhecidos, nós podemos utilizar algumas técnicas como:

- Active regularization: Quando nós penalizamos o modelo durante o treinamento, baseado na magnitude das ativações de cada Unit [2];
- Weight Constraint (ou clipping): Nós fazemos o ajuste de cada peso baseado num intervalo predefinido de magnitude de funções de ativação [2];
- Poda (Dropout): Removemos Units aleatoriamente do modelo de forma que a complexidade da rede diminua [2]. Apenas durante o treinamento, a cada iteração de mini batch nós removemos temporariamente uma Unit com uma probabilidade p associada e rodamos o treinamento [2]. A complexidade diminui justamente por que a remoção de neurônios elimina a computação de pesos e atributos na rede durante o feedforward e durante a backpropagation, o que torna nosso modelo menos ajustado aos dados de treinamento;
- Validation Set: A fim de obter uma hipótese generalista, nós podemos utilizar uma das técnicas de protocolo de validação chamado validation set. Basicamente nós monitoramos o desempenho da rede durante o treinamento através do training set e validation set, em que ajustamos os hiperparâmetros do modelo para performar melhor [2];
- Early Stopping: Nesse método o modelo é treinado até atingir um limiar de degradação, ou seja, até que a generalização diminua e o overfitting seja detectado, sendo uma boa alternativa para diminuir o sobreajuste e melhorar a performance do modelo em conjuntos de dados não vistos [2].

III. Trabalhos Relacionados

Para desenvolver este trabalho nos baseamos em alguns artigos já publicados e encontrados através de buscas no Google Scholar, utilizando as strings de busca “Empresa júnior” e “Machine Learning”.

A. Artigos

- O paper “Planejamento Estratégico da Rede: Resultados do movimento empresa júnior a partir de estímulo para crescimento”, escrito por Rutzen e Paula Benedetti utiliza duas bases de dados da FEJESP para medir lacunas como metas a serem atingidas e lacunas como metas já atingidas. Os resultados mostram os estados com maior e menor percentual de crescimento, bem como os grupos de empresas juniores.

IV. Objetivo

O principal objetivo deste projeto é analisar a probabilidade de uma empresa júnior ou instância alcançar suas metas, utilizando dados disponíveis na base das federações selecionadas. Por meio da aplicação de técnicas avançadas de análise de dados e modelagem estatística, o projeto visa identificar padrões, tendências e fatores que influenciam o desempenho das empresas juniores e fornecer insights para gestores e tomadores de decisão melhorarem a performance de suas empresas.

V. Metodologia Experimental

Como foi mencionado anteriormente, o modelo estatístico escolhido foram as redes neurais artificiais para prever a probabilidade de uma empresa júnior ou instância alcançar suas metas. A tecnologia utilizada será o interpretador Python de qualquer versão e a infraestrutura será o Google Colab.

A. Conhecimento de domínio

Aqui nós realizamos uma investigação detalhada dos dados fornecidos pela FEJESP dos anos de 2022, 2023 e 2024 para compreender a estrutura e as características dos dados, identificando possíveis desafios e oportunidades. Nossas análises nos permitiram observar uma base de dados com as seguintes features para cada ano: ID, Empresa Júnior, CNPJ, Faturamentos mensais, Federação (dentre FEJESP, FEJEPE, FEJECE, FEJEMG, Concentro e RN Júnior), Cluster, Alto Crescimento, Meta de Faturamento, Faturamento Alcançado, E-mail, Site, Facebook, Estado, Cidade, Universidade, Cursos e Membros.

B. Pré-Processamento

Esta fase colabora com a seleção dos atributos mais relevantes para serem incluídos no modelo (feature selection), tratamento de valores ausentes em atributos, codificação de variáveis e limpeza de valores sujos (não legíveis pelo modelo). Aqui também é necessário entender a relação entre as variáveis independentes (atributos) e a dependente (saída), para que possamos saber se a hipótese é linear ou não linear.

Sendo assim, considerando que nosso objetivo é prever o faturamento de uma empresa júnior baseado nos dados do passado, apenas as features de faturamento mensal foram selecionadas para servir de entrada para o modelo. Inicialmente a base de dados seria composta apenas dos registros da FEJESP, porém, a fim de diminuir a incidência de overfitting, as outras federações foram escolhidas para complementar o conjunto de dados do projeto, visto que elas possuem maturidade parecida com a da FEJESP.

Considerando que nós estamos utilizando redes neurais recorrentes, que quando especializadas como redes GRU e LSTM são capazes de trabalhar com a complexidade de períodos temporais, nós utilizamos a separação dos dados em tuplas, em que blocos de 3 meses de faturamento consecutivo servem de entrada para as redes na intenção de prever o quarto mês consecutivo, também encapsulado em um bloco e dentro dessa tupla (Bloco X, Bloco Y), realizando shift à direita desse bloco, o que, para os anos de 2022, 2023 e 2024 totaliza 26 períodos de faturamento mensal, de janeiro de 2022 à maio de 2024.

---------- BLOCK 25 ----------
---------- X SET ----------

	2_2024	3_2024	4_2024
86	125.0	125.0	125.0
304	13000.0	13000.0	21260.0
434	985.0	985.0	985.0
312	680.0	2900.0	2900.0
72	13796.0	13796.0	15342.2
..	...	...	...
428	35400.0	37300.0	58036.0
176	15000.0	27300.0	27300.0
594	2925.0	2925.0	2925.0
621	0.0	0.0	0.0
327	342.0	342.0	342.0

[129 rows x 3 columns]
---------- Y SET ----------

	5_2024
86	125.0
304	33020.0
434	985.0
312	2900.0
72	18742.2
..	...
428	59886.0
176	40220.0
594	2925.0
621	0.0
327	342.0

[129 rows x 1 columns]

Fig 7. Tabela de exemplo de separação dos dados em bloco. Fonte: Autoria própria.

Além disso, o conjunto de dados foi separado em 80% para treinamento, 20% destes para o conjunto de validação e 20% para o conjunto de teste, totalizando 409, 197 e 129 dados respectivamente e, a partir desta divisão, foi feita a separação em blocos (períodos de 4 meses consecutivos).

C. Extração de Padrões

Para a extração de padrões, nós definimos como configuração padrão das redes um total de 50 units, taxa de dropout de 50%, regularização l2 de 0% e taxa de aprendizado de 0.01%. Tanto o modelo GRU quanto o LSTM teve como hiperparâmetros 100 neurônios, dropout de 0.5%, taxa de aprendizado de 0.001% e L2 de 0.01%.

Tendo em vista que trabalhamos com gradient descent para encontrar pesos, que são sensíveis a escala dos valores das features, nós realizamos a normalização padrão dos valores ao redimensionar as características para que tenham média 0 e variância 1, nos ajudando a evitar que uma feature domine o aprendizado, o que nos ajuda a convergir a uma solução ótima mais rapidamente.

Ao observar nossos conjuntos de dados de treino, validação e teste, podemos considerar que um batch de tamanho 4 nos permite atualizar os parâmetros no treinamento de maneira que o impacto de outliers, como os faturamentos mensais de empresas como Poli Usp e FEA Jr, não acabem por enviesar o treino, o que aumenta a Perda. Uma observação é que para batches maiores do que 8 o modelo começa a ter dificuldades para se ajustar aos dados de entrada, além de um número menor que 4 tornar o treinamento mais lento e de convergência instável.

Além disso, o treinamento foi dividido em duas abordagens, uma sem e outra com Early Stopping. A primeira rodou com 10 Epochs para cada um dos 26 blocos de dados, totalizando 260 Epochs. Já a segunda com 100 Epochs, totalizando possíveis 26000 Epochs. E tendo em vista os desempenhos dos modelos analisados pelas curvas de aprendizado, podemos dizer que uma patience de 10 Epochs para GRU e LSTM nos dá uma partida do mínimo aceitável do modelo, sem que o treinamento cesse muito cedo ou acabe caindo em overfitting.

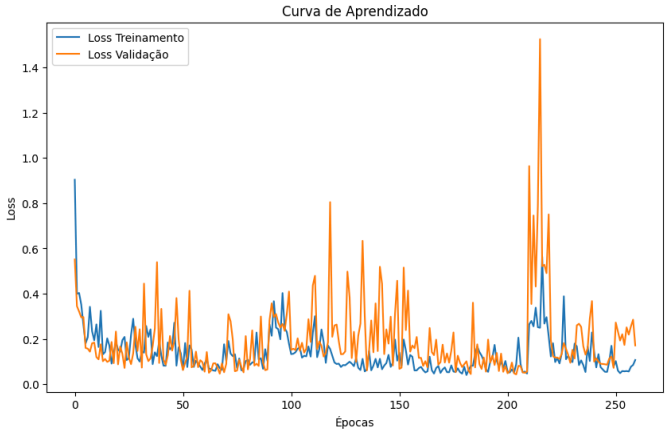

Fig 8. Curva de aprendizado do modelo GRU. Fonte: Autoria própria.

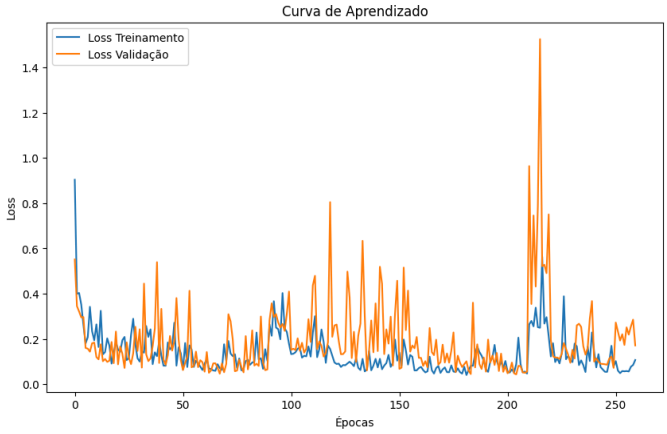

Fig 9. Curva de aprendizado do modelo LSTM. Fonte: Autoria própria.

D. Pós-Processamento

Nessa etapa vamos avaliar a performance do modelo desenvolvido utilizando métricas como MSE (mean square error), MAE (Mean Absolute Error), coeficiente de determinação R2 e teste T. Para as métricas de Empirical Loss e R2, foi feita a média das pontuações de todos os 26 blocos de treino, validação e teste, o que suaviza as variações individuais dos blocos e fornece uma estimativa mais confiável de como o modelo performa em novos conjuntos. Além disso, foi possível visualizar graficamente a diferença de faturamento real e previsto pelos modelos em gráficos de linha, dispersão e confusão, três óticas diferentes que nos dão um norte de erro, tendência e capacidade de generalização, bem como a visualização da diferença da perda L2 na última Epoch de um bloco com a sua primeira, nos permitindo ter uma outra visão do quão sobre ajustado está o modelo durante o treinamento.

F. Principais Bibliotecas

Pandas: Utilizada para manipulação de dataframes, facilitando a análise e manipulação de dados tabulares. É especialmente útil para operações como leitura, escrita, filtragem e agregação de dados.

Numpy: Biblioteca fundamental para computação científica em Python, usada principalmente para manipulação de matrizes e arrays multidimensionais. Numpy oferece funções matemáticas de alto desempenho e operações vetorizadas.

Matplotlib: Uma biblioteca de visualização de dados que permite criar gráficos estáticos, animados e interativos em Python. É amplamente utilizada para criar gráficos básicos como linhas, barras e histogramas.

Seaborn: Construída sobre o Matplotlib, Seaborn oferece uma interface de alto nível para criar visualizações estatísticas mais atraentes e informativas, como gráficos de distribuição, gráficos de correlação e gráficos categóricos.

Scikit-Learn: Uma biblioteca robusta de aprendizado de máquina que oferece ferramentas para pré-processamento de dados, implementação de modelos, e cálculo de métricas de avaliação. No contexto do seu código, é usada para técnicas de validação cruzada, regressão e cálculo de erros.

Keras (via TensorFlow):

- Sequential: Uma interface simples para construir modelos de redes neurais camada por camada.
- Dense: Uma camada totalmente conectada para redes neurais artificiais.
- Dropout: Uma técnica de regularização que ajuda a prevenir o overfitting desligando aleatoriamente neurônios durante o treinamento.
- Adam: Um otimizador eficiente que combina as vantagens dos métodos Adagrad e RMSProp.
- GRU: Gated Recurrent Unit, uma camada de rede neural recorrente utilizada em tarefas de séries temporais.
- LSTM: Long Short-Term Memory, outra camada de rede neural recorrente que é eficaz para capturar dependências de longo prazo em dados de séries temporais.
- Input: Uma camada usada para definir a entrada de um modelo.
- Regularizers: Usados para aplicar regularização em pesos de camadas para evitar overfitting.
- Early Stopping: Um callback que interrompe o treinamento se a métrica monitorada não melhorar após um número definido de épocas.

Google Colab:

- Drive: Biblioteca usada para montar o Google Drive no ambiente Colab, permitindo acessar arquivos armazenados no Google Drive.
- gspread e oauth2client: Bibliotecas usadas para acessar e manipular planilhas do Google Sheets a partir do Python, utilizando autenticação OAuth 2.0 para permitir o acesso seguro aos dados.

Standard Scaler: Parte do Scikit-Learn, é usada para padronizar os dados (ou seja, redimensionar as características para que tenham média 0 e variância 1).

VI. Resultados/ Discussões

Sem Early Stopping

Foi discutido que o desempenho de LSTM e GRU poderiam melhorar diante da mudança dos hiperparâmetros da rede e do treinamento, sendo estes um dos próximos possíveis passos para melhorar o desempenho do modelo, além de testar em novas abordagens a possibilidade de utilizar K-Fold Cross Validation para aumentar a generalização.

O modelo GRU mostrou ter bom desempenho quando configurado com 100 neurônios, dropout de 0.5%, taxa de aprendizado de 0.001 e L2 de 0.01, com os valores de avaliação abaixo. Se observarmos o desempenho durante a validação e teste, podemos sugerir que o desempenho melhor no conjunto de validação é devido a uma dificuldade maior de aprendizado daqueles dados. A linha “Blocks que houveram algum aprendizado” nos dá a possibilidade de medir a diferença da perda da primeira Epoch para a última daquele bloco que, quando negativa, nos sugere que o modelo teve dificuldades de aprender aquele período de faturamento.

```
Final Train Mean Squared Error: 0.04180448579195031
Final Train Mean Absolute Error: 0.07144311555349113
Final Train R² Score: 0.9581955142080496
Final Validation Mean Squared Error: 0.14422049509440749
Final Validation Mean Absolute Error: 0.10546382207306007
Final Validation R² Score: 0.5974137930637962
Blocks que houveram algum aprendizado:  18 / 26
Final Test Mean Squared Error: 0.04998464687381778
Final Test Mean Absolute Error: 0.08001264862374549
Final Test R² Score: 0.8135923808639505
```

Fig 10. Tabela da média de MSE, MAE e R2 Score de cada bloco de dados na rede GRU. Fonte: Autoria própria.

Abaixo conseguimos observar a diferença da função de perda L2 da primeira Epoch com a última de cada bloco de dados da rede, o que nos permitiu realizar ajustes no treinamento.

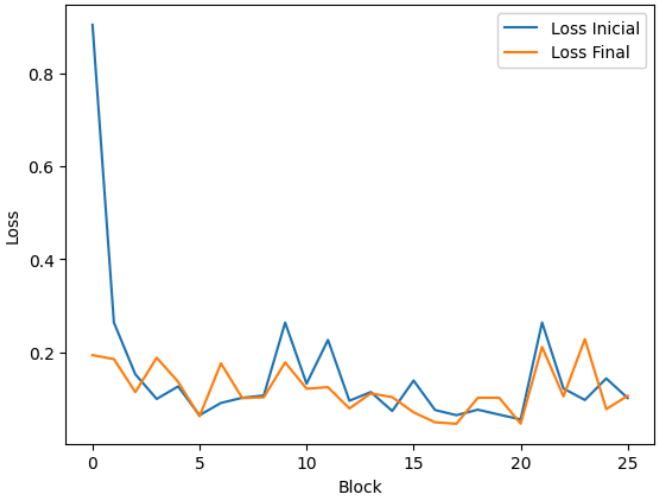

Fig 11. Diferença da função de perda L2 da primeira Epoch para a última de cada bloco de dados na rede GRU. Fonte: Autoria própria.

Na figura 12, podemos observar a diferença entre os valores previstos e reais do conjunto de teste em cada fim de Epoch de um bloco do treinamento, o que evidencia que a presença de outliers pode estar prejudicando a previsão de valores mais altos de faturamento.

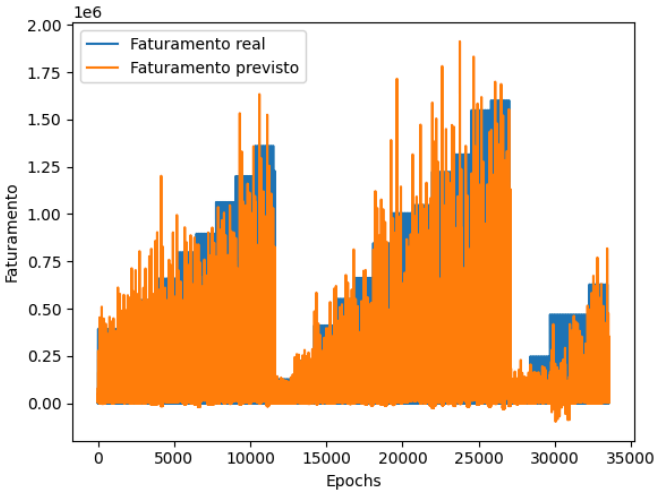

Fig 12. Diferença do faturamento real e previsto em cada fim de Epoch na rede GRU. Fonte: Autoria própria.

O gráfico abaixo mostra que o modelo tem uma tendência geral de acompanhar o comportamento dos dados reais, mas há variações que indicam erros de previsão. A dispersão ao redor da linha ideal (y=x) indica a precisão do modelo em diferentes faixas de valores de faturamento. Se a maioria dos pontos estivesse muito próxima da linha vermelha, isso indicaria um modelo com alta precisão. No entanto, a dispersão sugere que há espaço para melhorar a precisão das previsões.

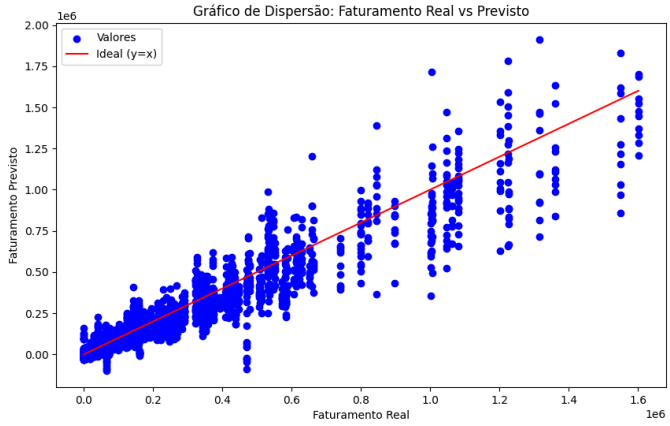

Fig 13. Faturamento Real vs Previsto no gráfico de dispersão na GRU. Fonte: Autoria própria.

A matriz de confusão adaptada na figura 15 revela a distribuição dos erros de previsão do modelo. Se a maioria dos valores estivesse concentrado ao longo da diagonal, isso indicaria que o modelo estava fazendo previsões bastante precisas. No entanto, a dispersão dos valores indica onde o modelo está subestimando ou superestimando o faturamento, fornecendo insights sobre áreas onde o modelo pode precisar de ajustes.

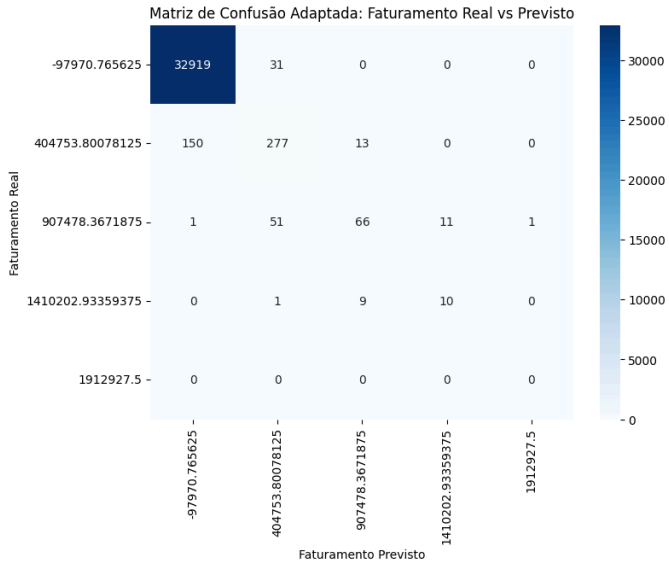

Fig 15. Faturamento Real vs Previsto na Matriz de Confusão na GRU. Fonte: Autoria própria.

O modelo LSTM mostrou um comportamento parecido com a GRU nos conjuntos de validação e teste. Como os dados utilizados pelos dois modelos são os mesmos e com divisão semelhante, isso reforça nossa ideia de que o conjunto de validação é mais difícil do que o de teste. Além disso, é observável que o modelo teve uma performance pior do que o modelo GRU, o que pode sugerir que o LSTM está com dificuldade de se ajustar aos dados de entrada, tendo em vista

que possui maior MSE, MAE e menor R2 em todos os conjuntos.

```
Final Train Mean Squared Error: 0.07957380771057881
Final Train Mean Absolute Error: 0.09215794315489825
Final Train R² Score: 0.9204261922894212
Final Validation Mean Squared Error: 0.2232083509841494
Final Validation Mean Absolute Error: 0.13764174803104876
Final Validation R² Score: 0.45730648290172027
Blocks que houveram algum aprendizado:  14 / 26
Final Test Mean Squared Error: 0.06242398155968715
Final Test Mean Absolute Error: 0.09623679321049186
Final Test R² Score: 0.7768944837698769
```

Fig 16. Tabela da média de MSE, MAE e R2 Score de cada bloco de dados na rede LSTM. Fonte: Autoria própria.

Assim como na GRU, abaixo conseguimos observar a diferença da função de perda L2 da primeira Epoch com a última de cada bloco de dados da rede LSTM, o que nos permitiu realizar ajustes no treinamento deste modelo.

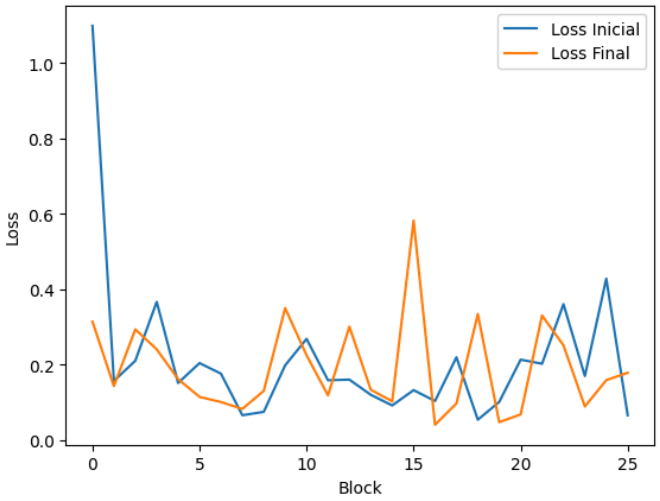

Fig 17. Diferença da função de perda L2 da primeira Epoch para a última de cada bloco de dados na rede LSTM. Fonte: Autoria própria.

Na figura 18, podemos observar um cenário parecido com o ocorrido em GRU, em que a diferença entre os valores previstos e reais do conjunto de teste em cada fim de Epoch de um bloco do treinamento evidencia que a presença de outliers pode estar prejudicando a previsão de valores mais altos de faturamento.

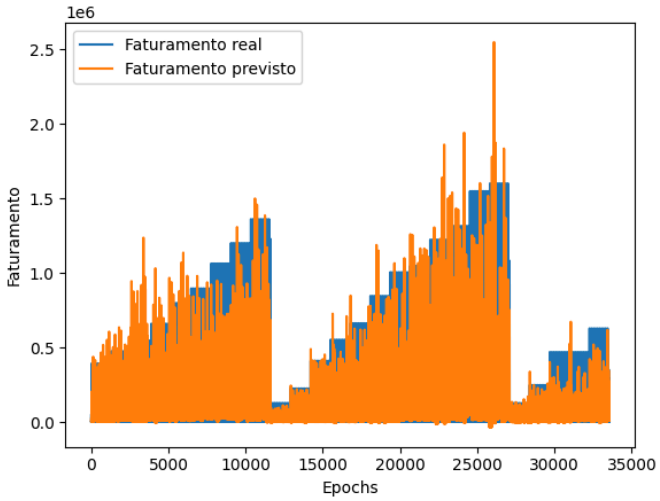

Fig 18. Diferença do faturamento real e previsto em cada fim de Epoch na rede LSTM. Fonte: Autoria própria.

O gráfico na figura 19 mostra um cenário parecido com o modelo GRU, em que existe uma tendência geral de acompanhar o comportamento dos dados reais, mas há variações que indicam erros de previsão. Novamente, a dispersão sugere que há espaço para melhorar a precisão das previsões.

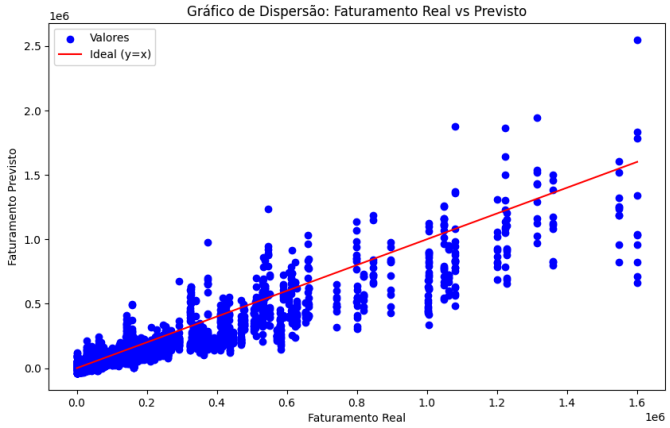

Fig 19. Faturamento Real vs Previsto no gráfico de dispersão da LSTM. Fonte: Autoria própria.

Assim como na matriz anterior, a matriz de confusão adaptada da figura 20 revela onde o modelo está subestimando ou superestimando o faturamento. No entanto, essa nova matriz sugere que o modelo tem dificuldades especialmente em prever os valores mais altos, já que as células associadas a altos valores de faturamento real e previsto têm valores muito baixos ou nulos. Isso pode indicar a necessidade de ajustes no modelo, especialmente para melhorar a previsão em faixas de faturamento mais elevadas.

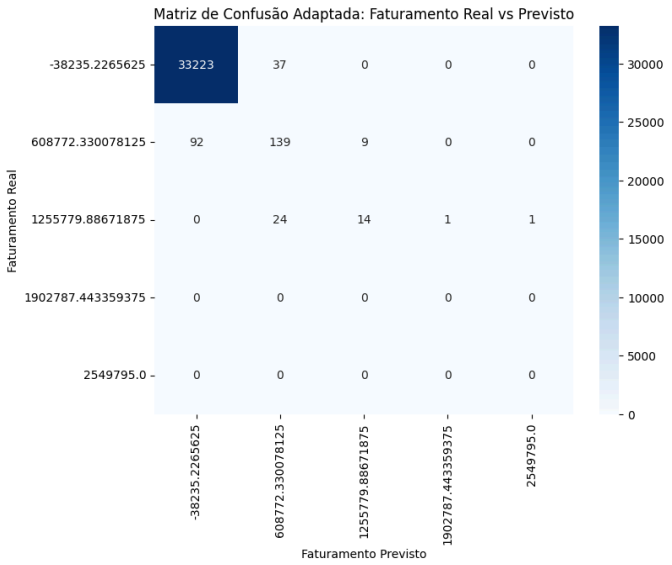

Fig 20. Faturamento Real vs Previsto na Matriz de Confusão da LSTM Fonte: Autoria própria.

Além disso, o Teste T evidenciou que existe diferença estatisticamente significativa entre o desempenho dos modelos GRU e LSTM no contexto em que foram comparados. O valor da estatística T (51.19) e o P-valor de 0.0 indicam que as médias dos dois grupos de coeficiente de determinação R2 são suficientemente diferentes para rejeitar a hipótese nula, que assume que não há diferença entre os desempenhos dos modelos.

```
Teste T para comparar GRU e LSTM:
Estatística T: 51.198967460740704
Valor-p: 0.0
```

Fig 21. Tabela de Teste T e p-valor. Fonte: Autoria própria.

Com Early Stopping

O uso de early stopping melhorou a generalização do modelo GRU, resultando em um R^2 de 0.6708 na validação e 0.8269 no teste, comparado a 0.5974 e 0.8136 sem early stopping. O modelo com early stopping também apresentou menor Mean Squared Error na validação (0.1238 vs. 0.1442) e mais blocos com aprendizado (23/26 vs. 18/26), indicando que essa técnica ajudou a evitar o overfitting e aprimorou a eficiência do treinamento. Já o modelo LSTM com early stopping mostrou um desempenho superior na validação, com um MSE e MAE menores e um R^2 Score mais alto, indicando melhor generalização e menor overfitting. No entanto, no treinamento, o modelo sem early stopping teve um desempenho ligeiramente melhor, com menor MSE e MAE e um R^2 Score mais alto. No teste, ambos os modelos apresentaram resultados semelhantes, mas o modelo sem early stopping teve ligeiramente melhores métricas de erro. Portanto, o early stopping é recomendado para equilibrar desempenho e generalização em futuras implementações.

```
Final Train Mean Squared Error: 0.05187115890042334
Final Train Mean Absolute Error: 0.06836512358381272
Final Train R² Score: 0.9481288410995765
Final Validation Mean Squared Error: 0.1237953814644859
Final Validation Mean Absolute Error: 0.09573236837846293
Final Validation R² Score: 0.6708198580887355
Blocks que houveram algum aprendizado:  23 / 26
Final Test Mean Squared Error: 0.047121892402689616
Final Test Mean Absolute Error: 0.07371607395925128
Final Test R² Score: 0.8269626035207641
```

Fig 22. Tabela da média de MSE, MAE e R2 Score de cada bloco de dados na rede GRU com Early Stopping. Fonte: Autoria própria.

```
Final Train Mean Squared Error: 0.07957380771057881
Final Train Mean Absolute Error: 0.09215794315489825
Final Train R² Score: 0.9204261922894212
Final Validation Mean Squared Error: 0.2232083509841494
Final Validation Mean Absolute Error: 0.13764174803104876
Final Validation R² Score: 0.45730648290172027
Blocks que houveram algum aprendizado:  14 / 26
Final Test Mean Squared Error: 0.06242398155968715
Final Test Mean Absolute Error: 0.09623679321049186
Final Test R² Score: 0.7768944837698769
```

Fig 23. Tabela da média de MSE, MAE e R2 Score de cada bloco de dados na rede GRU com Early Stopping. Fonte: Autoria própria.

Novamente, assim como na GRU sem Early Stopping, abaixo conseguimos observar a diferença da função de perda L2 da primeira Epoch com a última de cada bloco de dados de entrada, o que nos permitiu realizar ajustes no treinamento deste modelo ao analisar a capacidade de aprendizado por bloco.

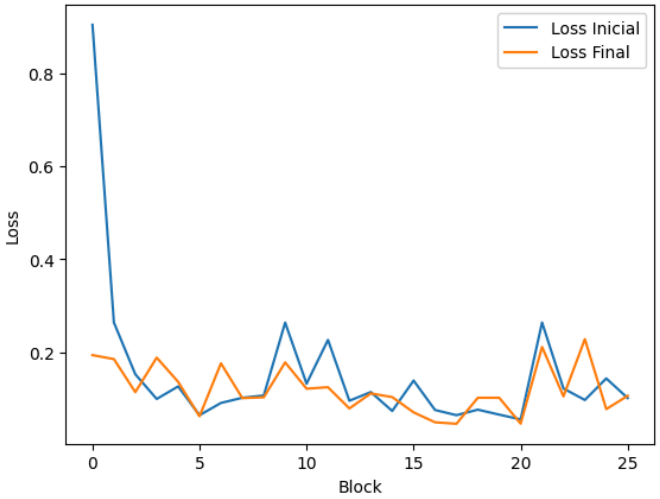

Fig 24. Diferença da função de perda L2 da primeira Epoch para a última de cada bloco de dados na rede GRU com Early Stopping. Fonte: Autoria própria.

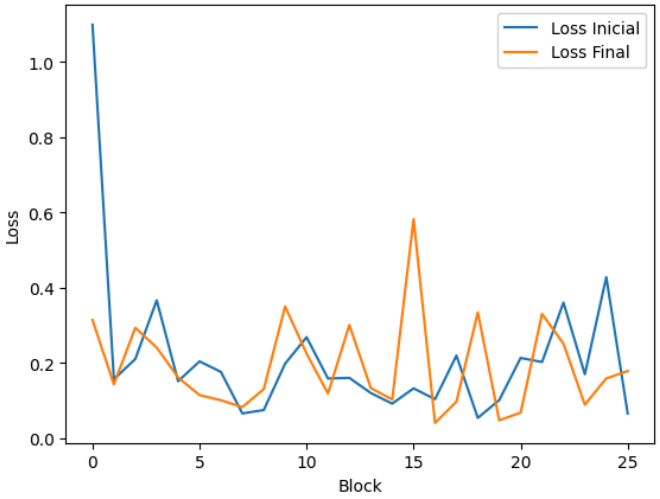

Fig 25. Diferença da função de perda L2 da primeira Epoch para a última de cada bloco de dados na rede LSTM com Early Stopping. Fonte: Autoria própria.

Para a rede GRU e LSTM com Early Stopping de patience 10, podemos observar na figura 26 e 27 um cenário bem parecido com o ocorrido em GRU e LSTM sem Early Stopping, em que a diferença entre os valores previstos e reais do conjunto de teste em cada fim de Epoch de um bloco do treinamento evidencia que a presença de outliers pode estar prejudicando a previsão de valores mais altos de faturamento.

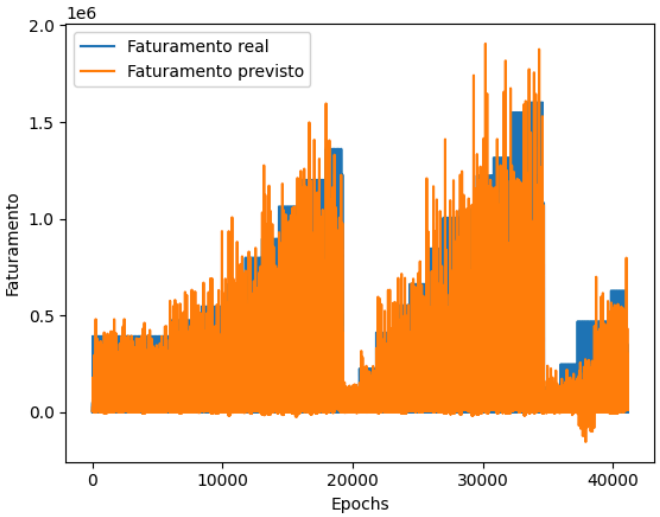

Fig 26. Diferença do faturamento real e previsto em cada fim de Epoch na rede GRU com Early Stopping. Fonte: Autoria própria.

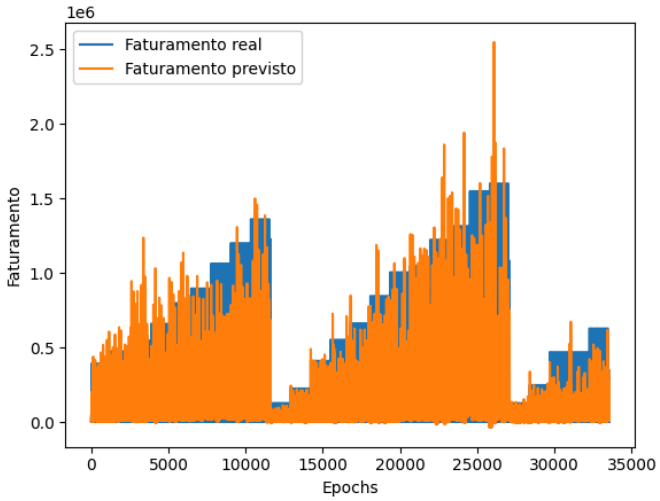

Fig 27. Diferença do faturamento real e previsto em cada fim de Epoch na rede LSTM com Early Stopping. Fonte: Autoria própria.

Os dois gráficos da matriz de confusão abaixo mostram um cenário parecido com o dos dois modelos GRU e LSTM sem Early Stopping, em que existe uma tendência geral de acompanhar o comportamento dos dados reais, mas há variações que indicam erros de previsão.

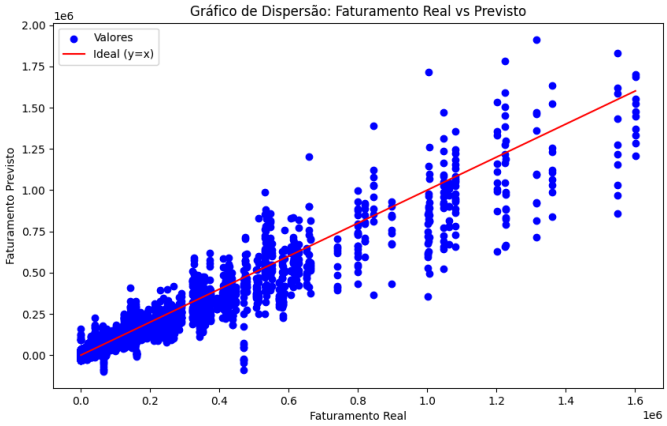

Fig 28. Faturamento Real vs Previsto no gráfico de dispersão da GRU com Early Stopping. Fonte: Autoria própria.

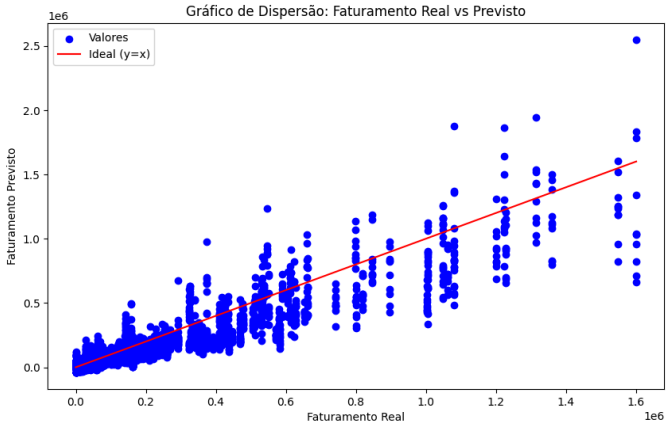

Fig 29. Faturamento Real vs Previsto no gráfico de dispersão da GRU com Early Stopping. Fonte: Autoria própria.

Assim como nas duas matrizes anteriores, as matrizes de confusão adaptadas da figura 30 e 31 revelam onde o modelo está subestimando ou superestimando o faturamento, tendo as mesmas sugestões estatísticas que as últimas duas matrizes.

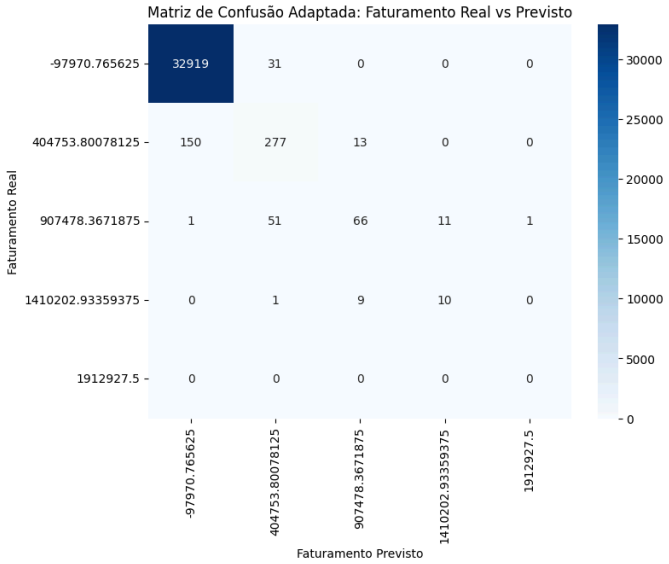

Fig 30. Faturamento Real vs Previsto na Matriz de Confusão da GRU com Early Stopping. Fonte: Autoria própria.

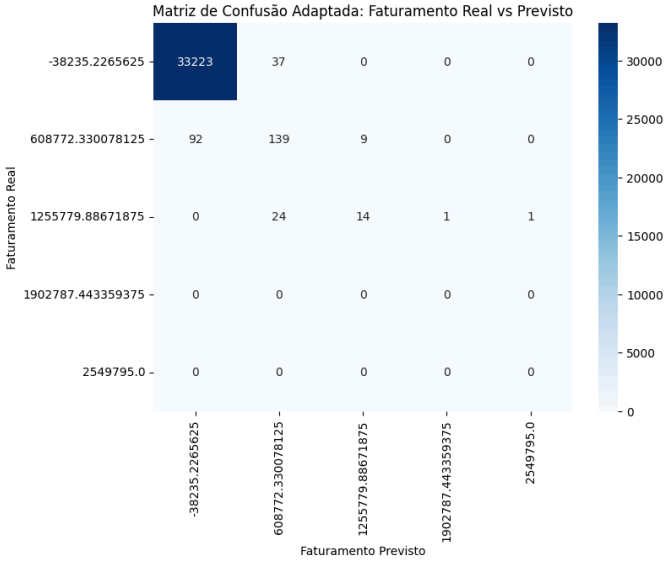

Fig 31. Faturamento Real vs Previsto na Matriz de Confusão da LSTM com Early Stopping. Fonte: Autoria própria.

VII. Conclusões/ Trabalhos Futuros

Neste estudo, avaliamos a eficácia dos modelos GRU e LSTM para prever a probabilidade de uma empresa atingir sua meta de faturamento. Os resultados demonstraram que o modelo GRU superou o LSTM em termos de MSE, MAE e R^2 Score, tanto com quanto sem early stopping. A superioridade do GRU pode ser atribuída à sua estrutura mais simples e menor número de parâmetros, o que promove uma melhor adaptação aos dados e reduz o risco de overfitting. Além disso, a GRU se mostrou mais eficaz na captura de dependências temporais relevantes para a previsão de metas de faturamento, e sua menor demanda computacional contribuiu para um desempenho mais eficiente. Esses fatores combinados destacam o GRU como a escolha preferencial para prever a probabilidade de uma empresa atingir sua meta de faturamento, oferecendo resultados mais precisos e eficientes.

A FEJESP como federação de coordenação e apoio às empresas juniores do estado de São Paulo será capaz de prover estímulos às diferentes empresas juniores através dos seus núcleos, bem como as próprias empresas juniores serão capazes de analisar seu próprio desempenho naquele período e a se preparar antecipadamente para atingir os seus resultados, tudo isso orientado a dados. Além disso, o projeto é capaz de ser estendido a empresas do mercado sênior se continuado, sendo uma oportunidade de aplicação cada vez mais empregada nos dias atuais.

References

[1] S. O. Rezende, “Sistemas inteligentes: Fundamentos e Aplicações”, 2003.

[2] S. Russel, P. Norvig, “Artificial Intelligence: A Modern Approach,” 3rd ed, pp. 529–551, 2009.