

Aprimoramento de um Modelo de IA para Operações Matemáticas Básicas

Ismael Wesley Neves de Brito - 2018101710

Objetivo do trabalho

- 1 - Aprimorar um modelo de aprendizado de matemática básica para melhorar sua eficiência e gerar previsões mais precisas.
- 2 - Comparar os resultados da versão original com a versão otimizada e apresentar uma análise gráfica para evidenciar a melhoria no desempenho do modelo.
- 3 - Melhorar o sistema especialista utilizado em conjunto com o modelo para fornecer um feedback mais detalhado dos resultados.

Arquivos do projeto

Todo o código do projeto foi desenvolvido em Python utilizando o google collab, o código fonte pode ser encontrado em:

https://colab.research.google.com/drive/1C-MUJz70zCyJ6im_q2AgPdGgBu217B7X?usp=sharing

Link para o modelo já treinado:

https://drive.usercontent.google.com/u/0/uc?id=1N1A_4jMUyYvC5igAUQyHf6rgs8J98Jbc&export=download

pode ser carregado através do código:

```
model = load_model(model_path)
```

Melhorias no modelo matemático

—

Base de dados utilizada

Base pequena com dados fixos e extremamente limitada

```
X = np.array([  
    [2, 3, 0], # 2 + 3  
    [3, 2, 0], # 3 + 2  
    [5, 1, 0], # 2 + 3  
    [5, 1, 1], # 5 - 1  
    [4, 2, 2], # 4 * 2  
    [6, 3, 3], # 6 / 3  
])
```

Gerar todas as combinações possíveis de num1 e num2 no intervalo, incluindo valores negativos

```
gerar_dados(num1_range=(-15, 15), num2_range=(-15, 15),  
step=1)
```

Utilizada uma função que gera todas as operações possíveis com números inteiros, usando A = [-15 até 15] e B = [-15 até 15] para cada uma das 4 operações utilizadas (+ - * /)



Exemplos:

$$15 + 15 = 30$$

$$10 * 10 = 100$$

$$-10 - 10 = -20$$

Estrutura

Original:

O modelo é uma rede neural com uma camada oculta de 10 neurônios e uma camada de saída, com um total de 155 parâmetros

Atualizado para:

Uma rede neural mais complexa, com três camadas ocultas de 128 neurônios, com um total de 33,665 parâmetros

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 10)	40
dense_5 (Dense)	(None, 1)	11

Total params: 155 (624.00 B)
Trainable params: 51 (204.00 B)
Non-trainable params: 0 (0.00 B)
Optimizer params: 104 (420.00 B)



Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	512
dense_1 (Dense)	(None, 128)	16,512
dense_2 (Dense)	(None, 128)	16,512
dense_3 (Dense)	(None, 1)	129

Total params: 33,665 (131.50 KB)
Trainable params: 33,665 (131.50 KB)
Non-trainable params: 0 (0.00 B)

Treino do modelo

Treino padrão com 100 épocas.

```
model_ori.fit(X_train2, y_train2,  
              epochs=100,  
              batch_size=2,  
              verbose=1)
```

Avalie o modelo

```
loss =  
model_ori.evaluate(X_test2,  
y_test2, verbose=0)
```

```
print(f"Erro médio quadrático no  
teste: {loss}")
```



Modelo treinado por X épocas, N vezes para se aperfeiçoar, a cada época do treino o modelo é avaliado e salva-se o melhor peso do treinamento.

```
for epoch in range(epochs_max):  
    print(f"\nIniciando Epoch {epoch + 1}/{epochs_max}")  
  
    history = model.fit(X_train, y_train,  
                        epochs=1,  
                        batch_size=2,  
                        verbose=1,  
                        validation_split=0.2  
                        )  
  
    train_loss = history.history['loss'][0]  
    val_loss = history.history['val_loss'][0]  
  
    y_val_pred = model.predict(X_train[int(0.9 * len(X_train))])  
  
    mse = mean_squared_error(y_train[int(0.9 * len(X_train)):], y_val_pred)  
  
    epoch_errors.append(mse)  
  
    if val_loss < best_val_loss:  
        best_val_loss = val_loss  
        best_model = model.get_weights()
```

Época por época

Em vez do modelo ser treinado por exemplo, 100 épocas e então finalizado, o modelo é treinado por cada época individualmente e após esse treino, o modelo é avaliado, para que caso seu desempenho seja melhor que o melhor desempenho atual, então ele possa ser armazenado como novo melhor.

Isso faz com que após todo o treinamento do modelo, o modelo final seja o melhor modelo encontrado durante as épocas, podendo ser treinado novamente após isso em busca de um modelo ainda mais eficiente.

```
Epoch 17:
  Perda de Treinamento: 105.6972
  Perda de Validação: 21.9670
  Erro Quadrático Médio (MSE): 21.7413

Iniciando Epoch 18/100
1190/1190 ————— 5s 4ms/step - loss: 26.6426 - mae: 2.6967 - val_loss: 22.7737 - val_mae: 2.5764
10/10 ————— 0s 5ms/step
Epoch 18:
  Perda de Treinamento: 36.7239
  Perda de Validação: 22.7737
  Erro Quadrático Médio (MSE): 24.7172

Iniciando Epoch 19/100
1190/1190 ————— 4s 3ms/step - loss: 22.6154 - mae: 2.5296 - val_loss: 11.5237 - val_mae: 2.0308
10/10 ————— 0s 4ms/step
Epoch 19:
  Perda de Treinamento: 30.5028
  Perda de Validação: 11.5237
  Erro Quadrático Médio (MSE): 12.9910
  ** Melhores Pesos Salvos! **

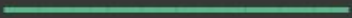
Iniciando Epoch 20/100
1190/1190 ————— 4s 3ms/step - loss: 24.8628 - mae: 2.6877 - val_loss: 11.5817 - val_mae: 1.9153
10/10 ————— 0s 6ms/step
Epoch 20:
  Perda de Treinamento: 35.3405
  Perda de Validação: 11.5817
  Erro Quadrático Médio (MSE): 12.3578
```

```
# Restaura os melhores pesos no modelo
model.set_weights(best_model)
print(f"Melhor perda na validação: {best_val_loss}")
```

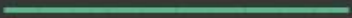
```
Melhor perda na validação: 0.4500686228275299
```


Sistema especialista

Feedback melhorado para o usuário

1/1  0s 72ms/step

Resposta incorreta: 0.31331413984298706. A resposta correta é 5. Tente novamente!

1/1  0s 48ms/step

Resposta incorreta: 0.7584635019302368. A resposta correta é 4. Tente novamente!



$-3.00 * -18.00 = 48.2584$ ✗ (Exato: 54). Erro de 10.63%! Valor previsto menor que o real.

$-6.00 - 4.00 = -10.035 \approx$ (Exato: -10). Resposta aceitável! Erro: 0.35% (máx: 7%).

$11.00 * -1.00 = -11.4698 \approx$ (Exato: -11). Resposta aceitável! Erro: 4.27% (máx: 7%).

$-8.00 * -1.00 = 8.6887$ ✗ (Exato: 8). Erro de 8.61%! Valor previsto maior que o real.

$9.00 / 11.00 = 0.8623 \approx$ (Exato: 0.8182). Resposta aceitável! Erro: 5.39% (máx: 7%).

Taxa percentual de erro

Como o modelo provavelmente não acertará exatamente o valor real do cálculo, foi adicionado uma taxa percentual de aceitação, especificamente **7%** de desvio para ser considerada uma resposta aceitável.

O motivo desta taxa ser percentual e não absoluta é porque o modelo foi treinado apenas com números relativamente baixos, então quando utilizado com números mais altos, sua precisão se torna não tão boa, principalmente quando se trata de multiplicação/divisão visto que os valores podem explodir muito.

Dessa forma, para números maiores, uma taxa percentual em vez de absoluta torna os erros mais aceitáveis visto que o modelo não foi treinado para resultados tão grandes.

Análise dos resultados do modelo

—

Exemplos de resultados

$-20.00 - -11.00 = -8.211$ X (Exato: -9). Erro de 8.77%! Valor previsto maior que o real.

$-6.00 + -20.00 = -25.5 \approx$ (Exato: -26). Resposta aceitável! Erro: 1.92% (máx: 7%).

$0.00 + 10.00 = 9.8955 \approx$ (Exato: 10). Resposta aceitável! Erro: 1.04% (máx: 7%).

$-7.00 * -20.00 = 130.0048$ X (Exato: 140). Erro de 7.14%! Valor previsto menor que o real.

$6.00 / -4.00 = -1.8403$ X (Exato: -1.5). Erro de 22.69%! Valor previsto menor que o real.

$-14.00 * -5.00 = 69.8251 \approx$ (Exato: 70). Resposta aceitável! Erro: 0.25% (máx: 7%).

$-14.00 + 18.00 = 3.7892 \approx$ (Exato: 4). Resposta aceitável! Erro: 5.27% (máx: 7%).

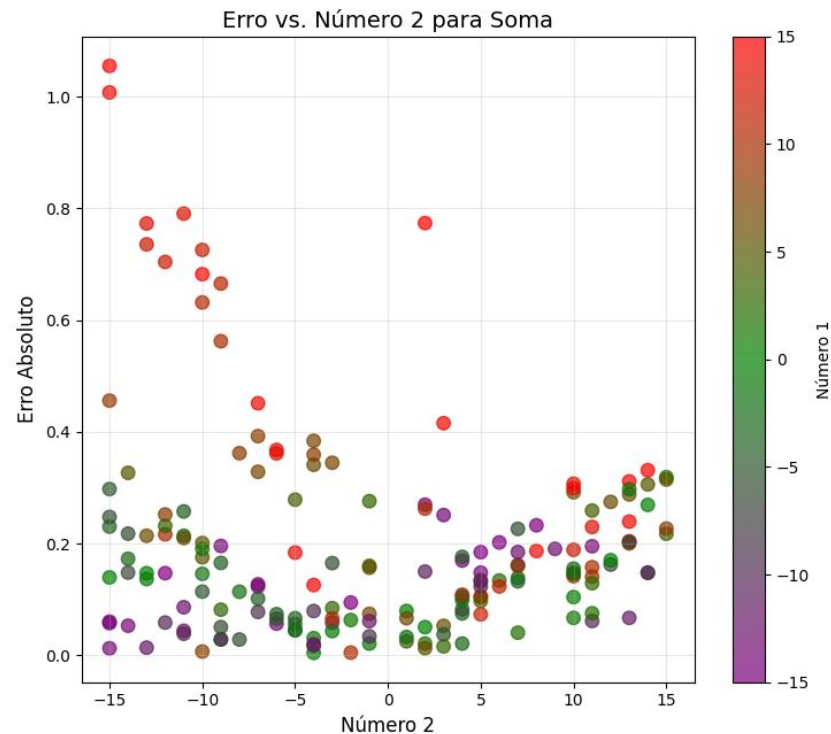
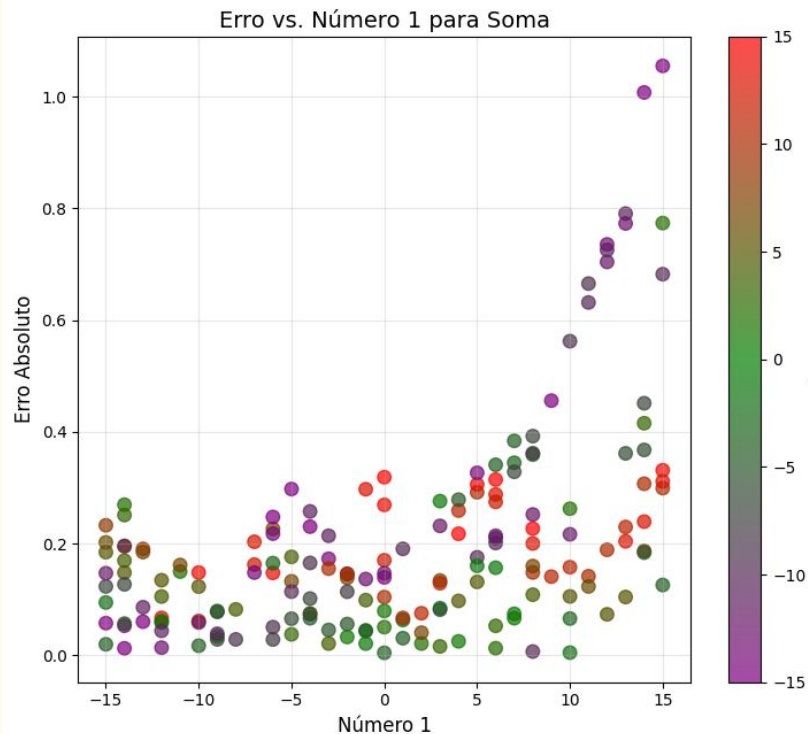
$-2.00 + -16.00 = -17.8623 \approx$ (Exato: -18). Resposta aceitável! Erro: 0.77% (máx: 7%).

$3.00 * 7.00 = 20.5519 \approx$ (Exato: 21). Resposta aceitável! Erro: 2.13% (máx: 7%).

$4.00 / -19.00 = -2.3853$ X (Exato: -0.2105). Erro de 1033.03%! Valor previsto menor que o real.

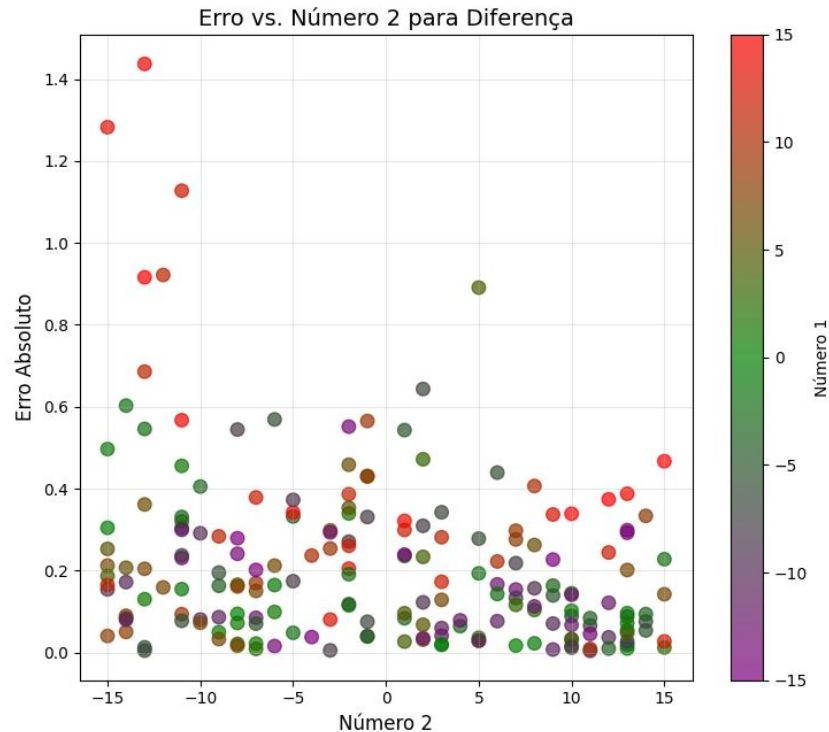
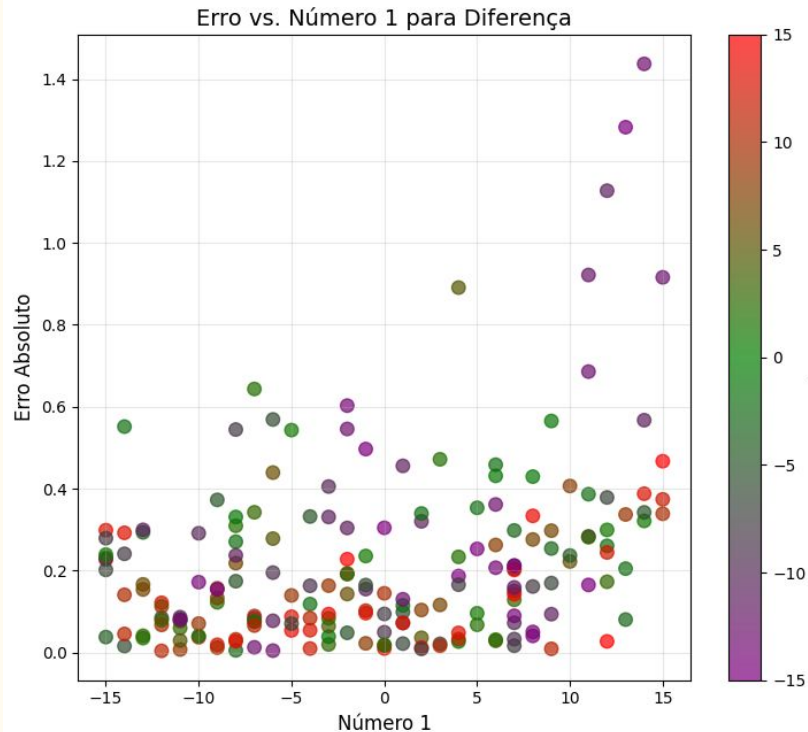
Análise de erros - Adição

Análise de Erros por Valores de Entrada para Adição



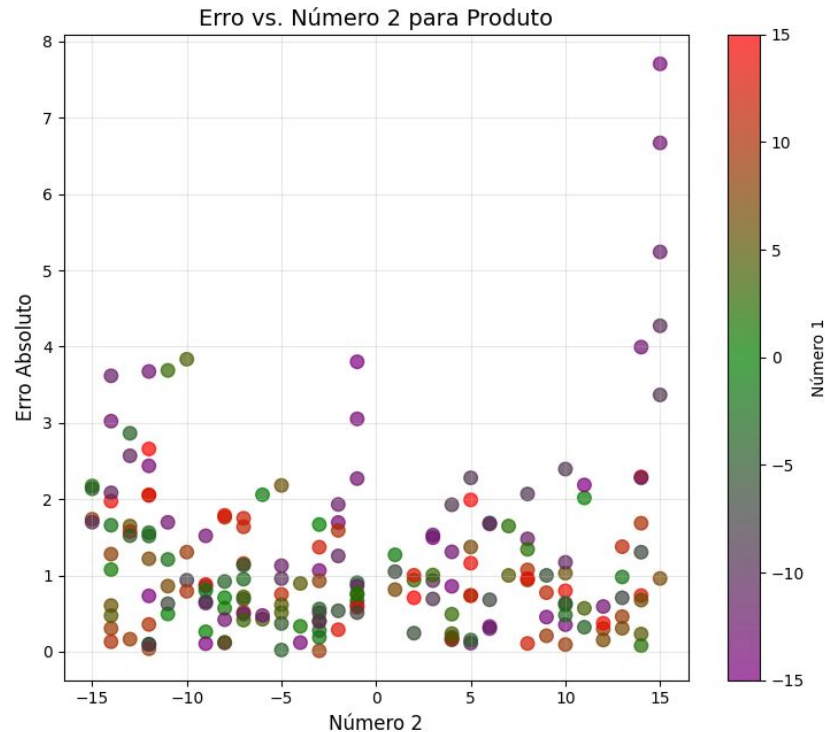
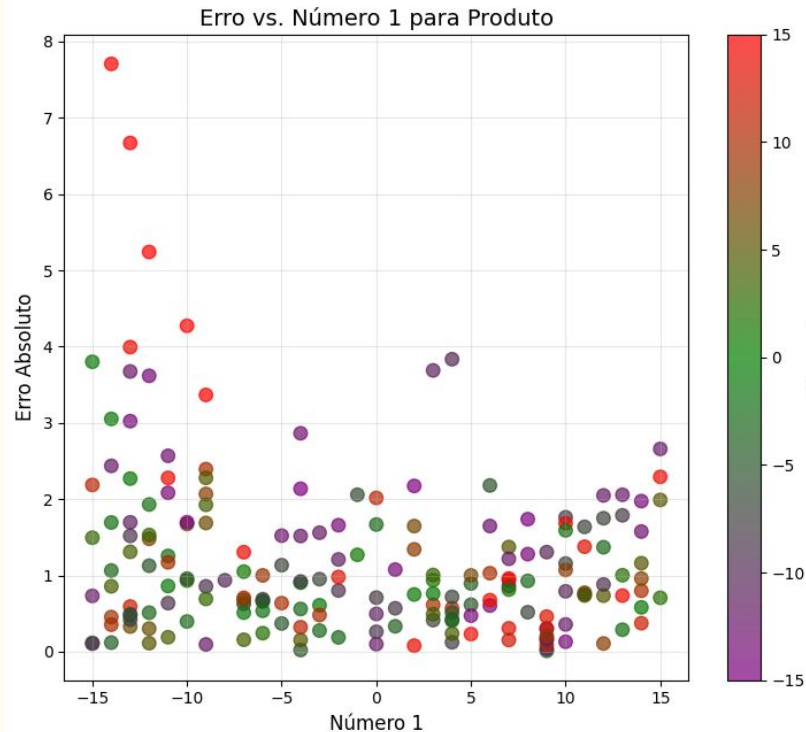
Análise de erros - Subtração

Análise de Erros por Valores de Entrada para Subtração



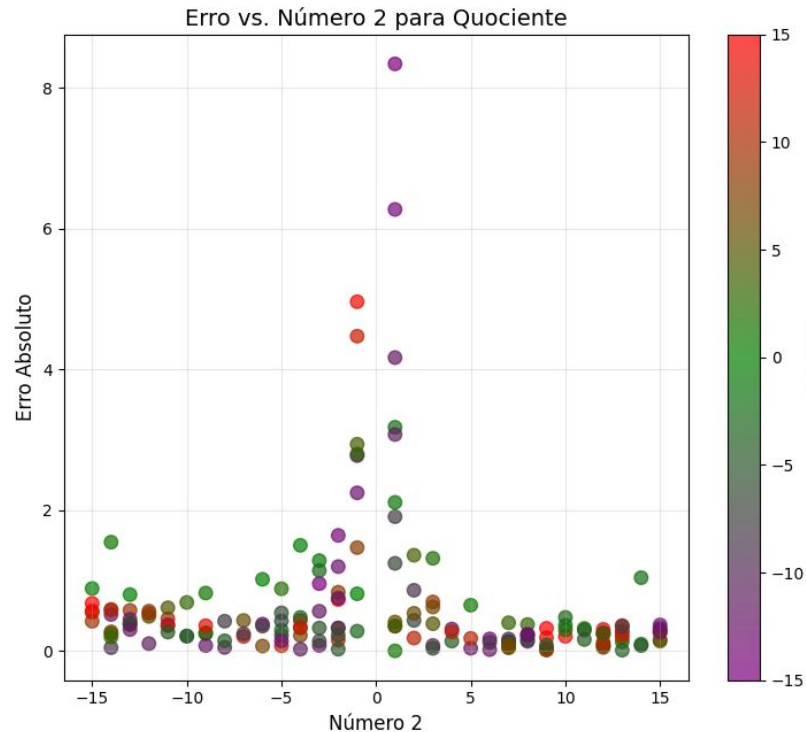
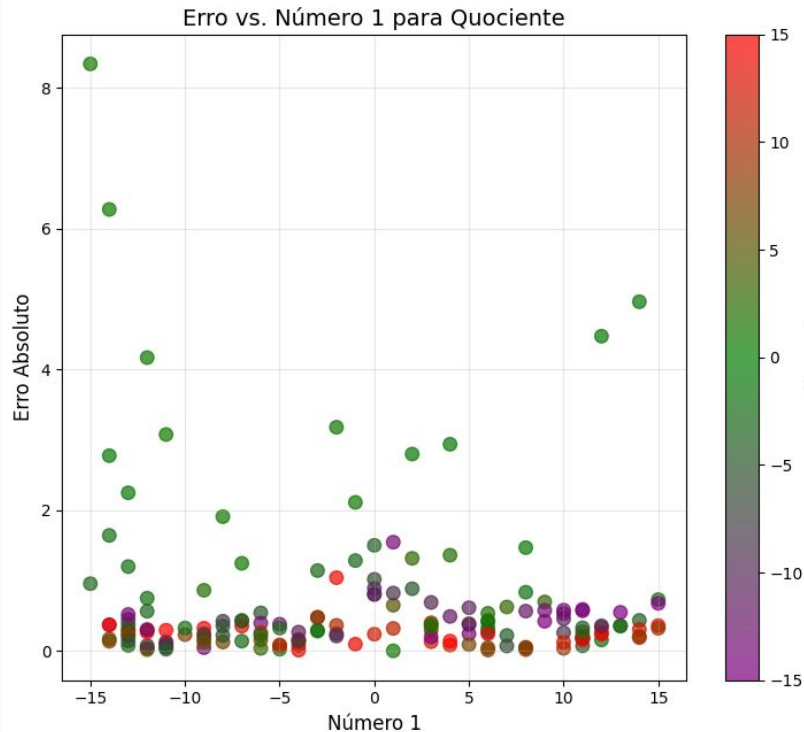
Análise de erros - Multiplicação

Análise de Erros por Valores de Entrada para Multiplicação

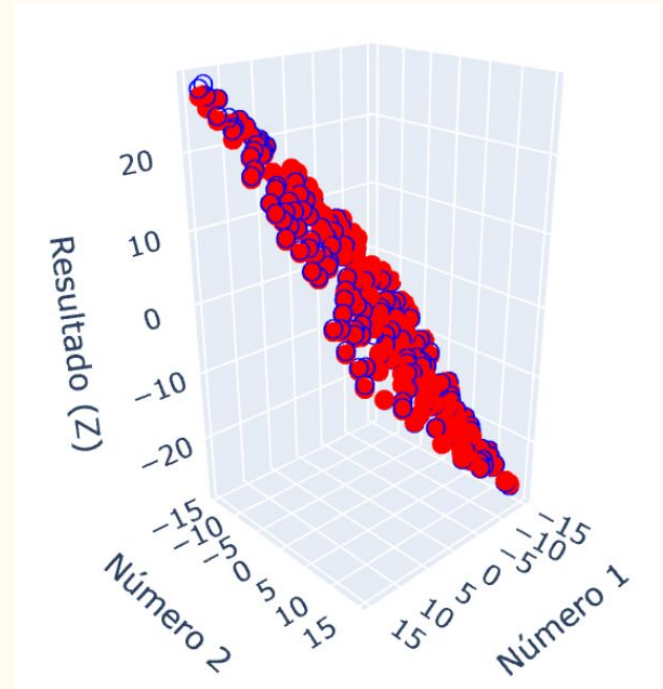
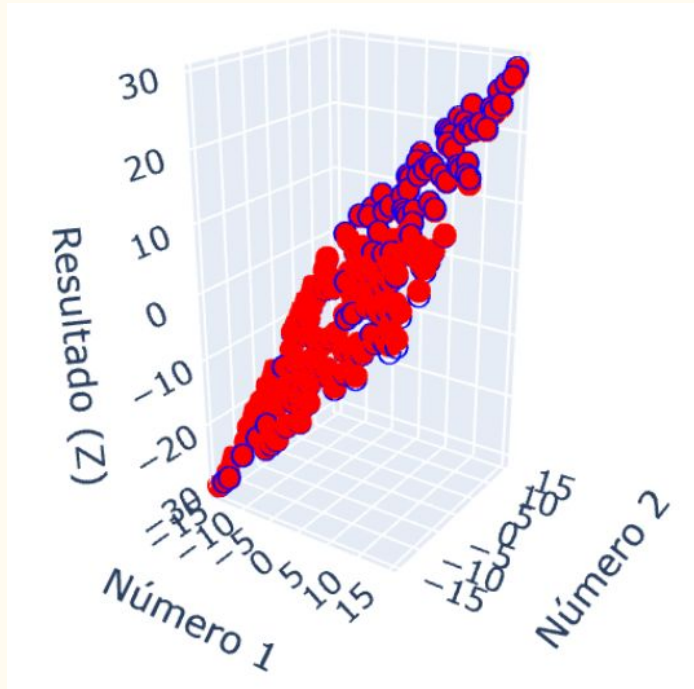


Análise de erros - Divisão

Análise de Erros por Valores de Entrada para Divisão

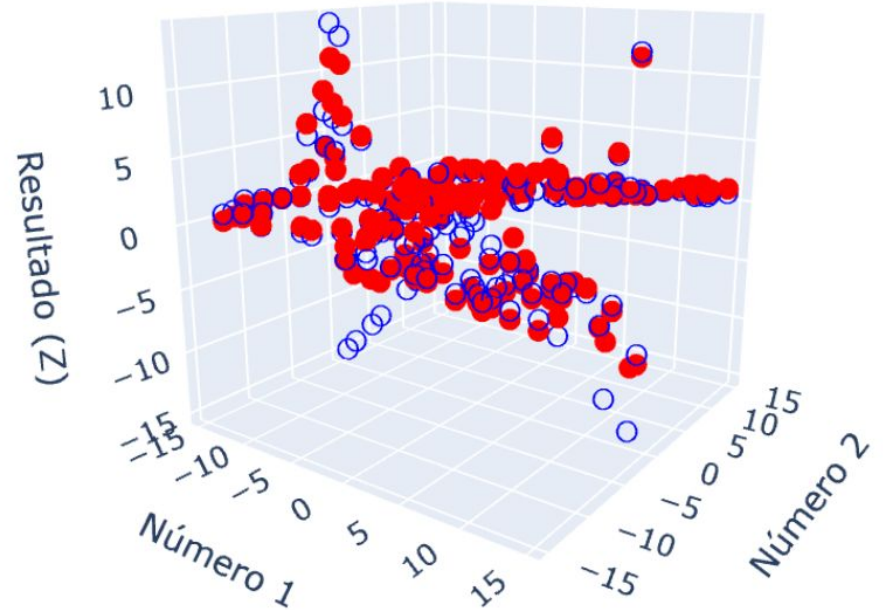
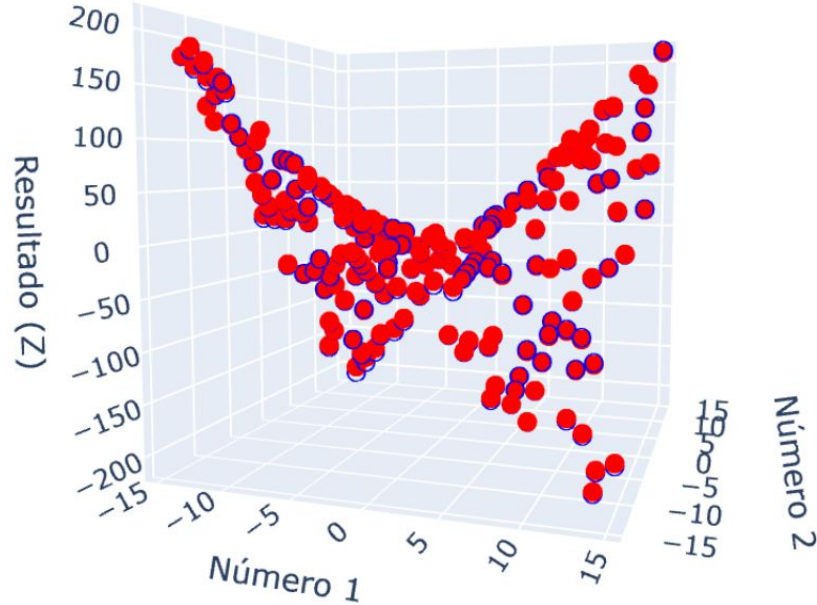


Real x Previsto - Soma e Subtração



*Esses gráficos 3D são interativos, então são melhores visualizados direto no repositório.

Real x Previsto - Multiplicação e Divisão



*Esses gráficos 3D são interativos, então são melhores visualizados direto no repositório.

Comparando os modelos Original x Novo

—

Como a comparação foi realizada

Para realizar a comparação entre os 2 modelos, foi gerado uma base para previsão contendo 300 operações aleatórias, com números entre -50 até 50 com todas as 4 operações (Cálculos com números que os modelos foram treinados e com números que não foram) .

Após os dois modelos preverem toda a base, foi gerada a comparação entre os seus resultados, de forma absoluta e também percentual.

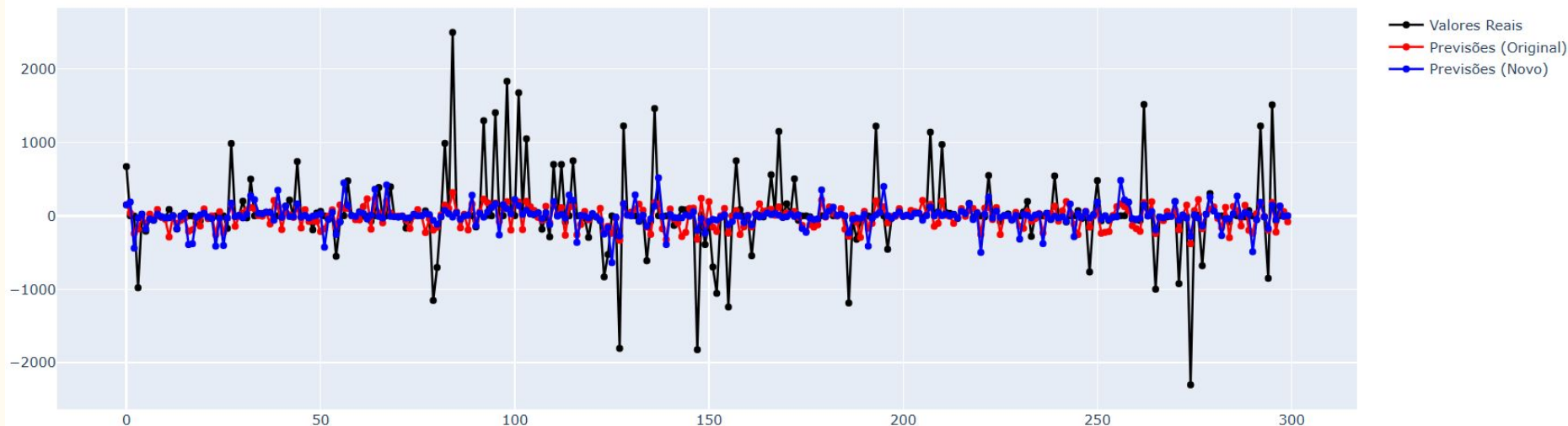
Também foi gerado gráficos comparativos para isso.

Previsões comparativas

	Operação	Real	Modelo Original	Erro Modelo Original	Erro Percentual Original (%)	Modelo Novo	Erro Modelo Novo	Erro Percentual Novo (%)
0	-21 * -32	672.00	153.63	-518.37	-77.14	147.84	-524.16	-78.00
1	12 / 40	0.30	39.44	39.14	13046.35	188.20	187.90	62632.71
2	45 / -22	-2.05	-240.68	-238.63	11666.42	-440.42	-438.38	21431.85
3	-20 * 49	-980.00	-181.37	798.63	-81.49	-28.80	951.20	-97.06
4	36 + -16	20.00	-18.74	-38.74	-193.69	24.72	4.72	23.62
5	-11 * 19	-209.00	-97.42	111.58	-53.39	-177.66	31.34	-14.99
6	-18 + -28	-46.00	23.39	69.39	-150.86	-42.46	3.54	-7.70
7	12 * -5	-60.00	-36.95	23.05	-38.41	-58.63	1.37	-2.29
8	-26 / -7	3.71	86.44	82.72	2227.15	24.97	21.25	572.21
9	-15 - -3	-12.00	-21.05	-9.05	75.39	-12.25	-0.25	2.05
10	-27 + 1	-26.00	-46.61	-20.61	79.28	-24.95	1.05	-4.06
11	44 - -43	87.00	-287.47	-374.47	-430.43	-26.46	-113.46	-130.41
12	7 / 4	1.75	0.36	-1.39	-79.71	1.49	-0.26	-15.01
13	-12 * 15	-180.00	-105.87	74.13	-41.18	-174.76	5.24	-2.91
14	-5 / 26	-0.19	-60.47	-60.28	31345.75	-9.68	-9.49	4934.41
15	26 - -11	37.00	-33.18	-70.18	-189.66	34.59	-2.41	-6.50
16	-41 / 34	-1.21	-209.93	-208.72	17308.43	-391.80	-390.60	32390.94

Gráfico das previsões totais

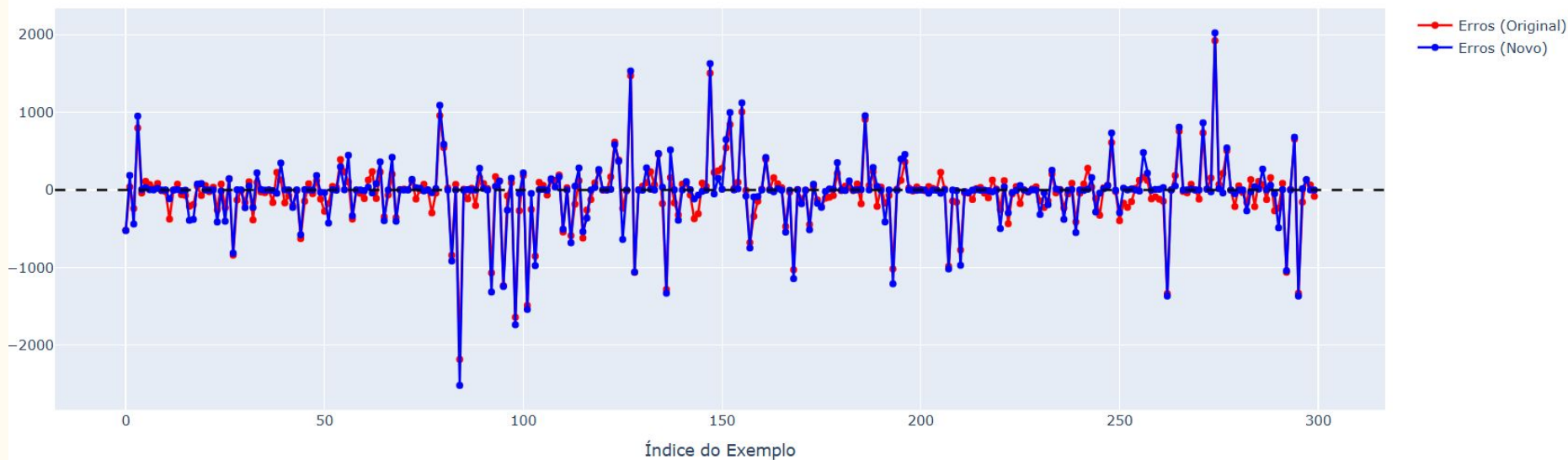
Comparação das Previsões (Todos os Operadores)



*Esse gráfico é interativos, então é melhor visualizado direto no repositório.

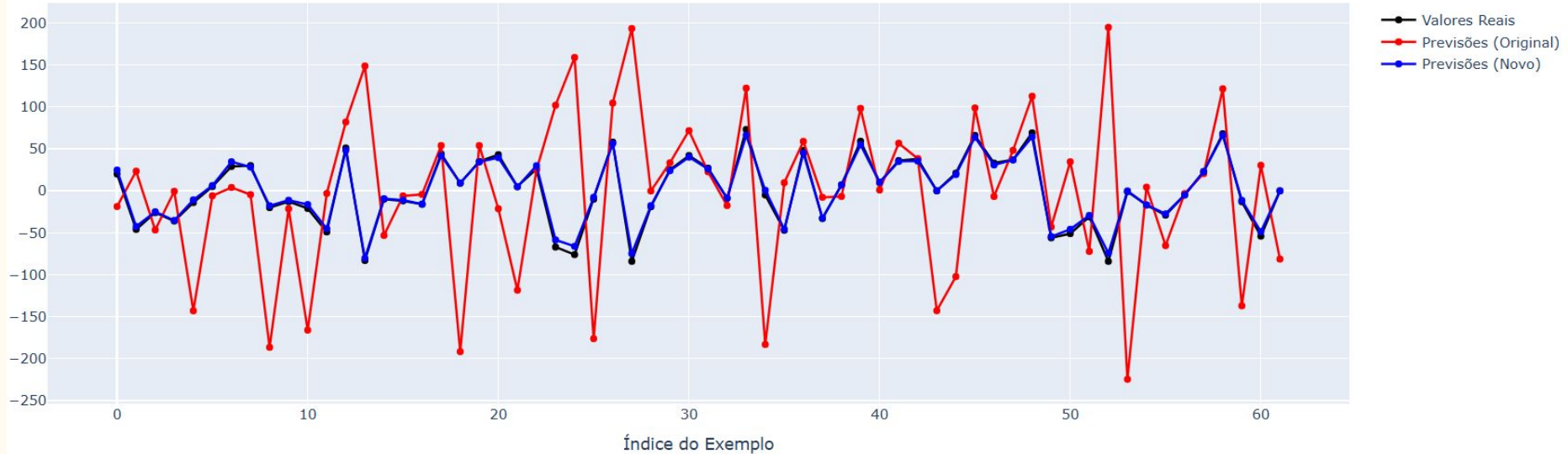
Gráfico dos erros totais

Comparação dos Erros de Previsão



*Esse gráfico é interativos, então é melhor visualizado direto no repositório.

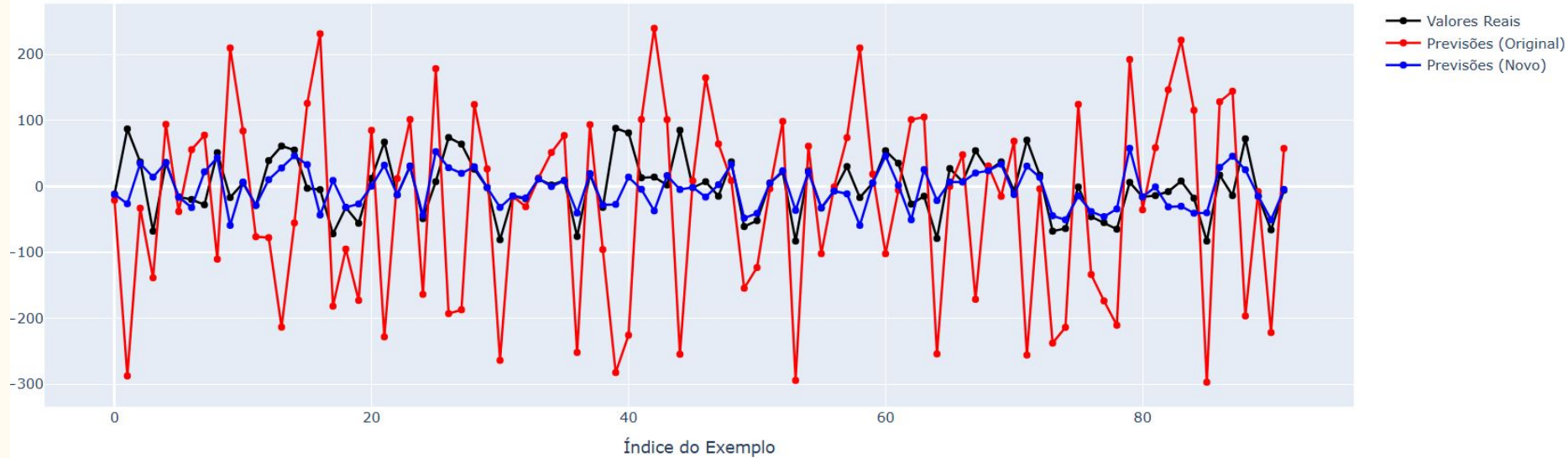
Gráfico das previsões de soma



Essa foi a operação em que o modelo melhor se adaptou, resultando em resultados muito precisos.

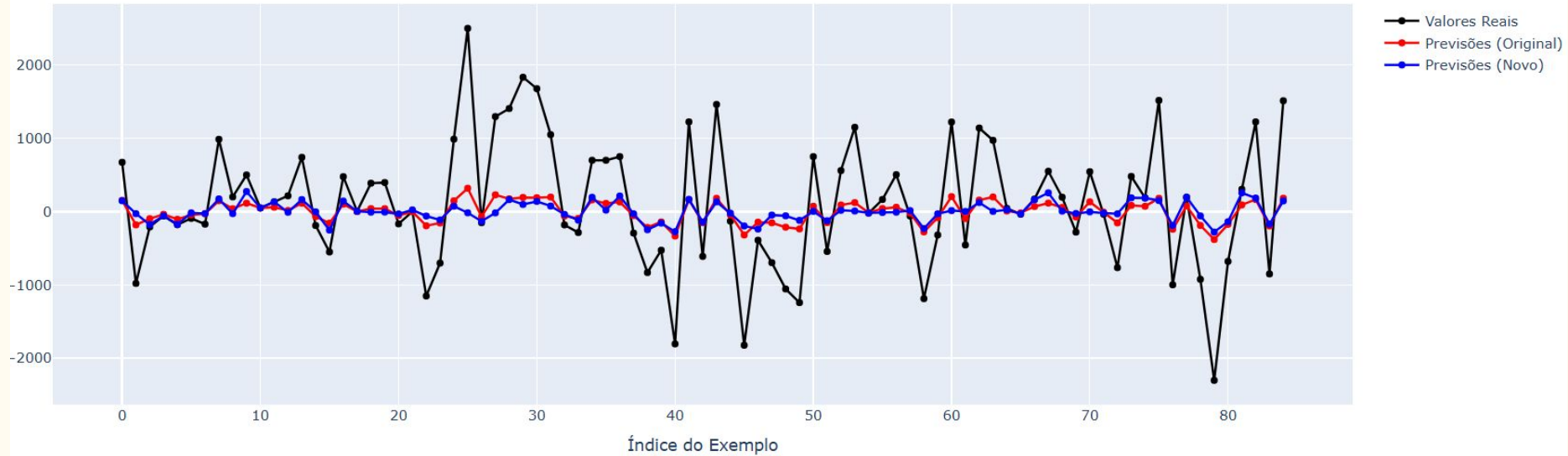
*Esse gráfico é interativos, então é melhor visualizado direto no repositório.

Gráfico das previsões de subtração



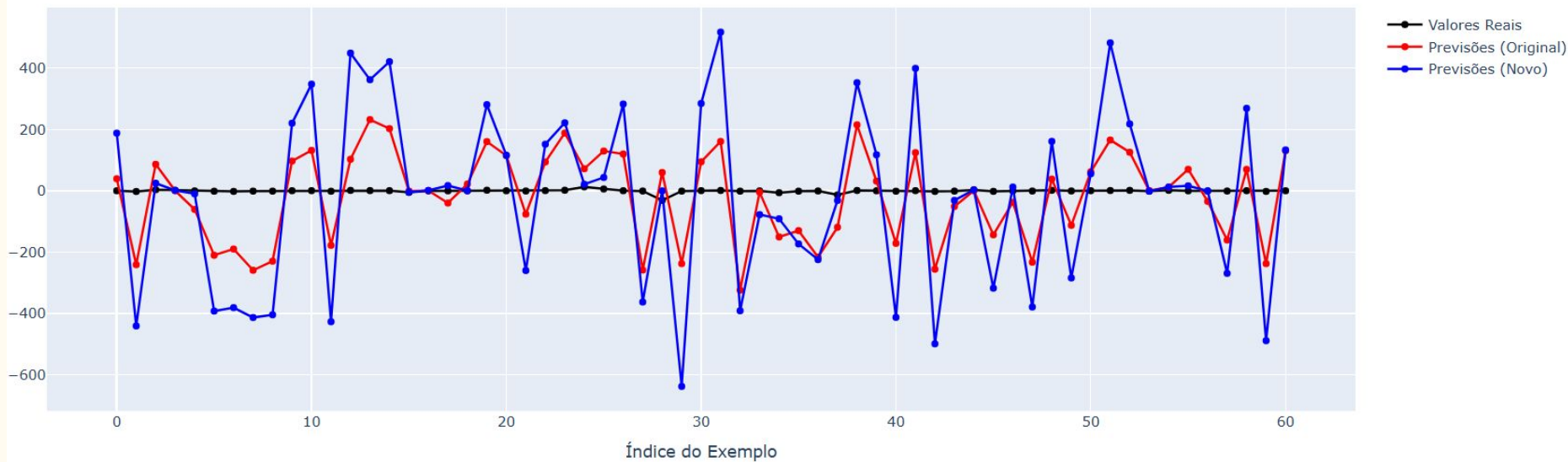
*Esse gráfico é interativos, então é melhor visualizado direto no repositório.

Gráfico das previsões de multiplicação



*Esse gráfico é interativos, então é melhor visualizado direto no repositório.

Gráfico das previsões de divisão



Pode-se perceber que essa é a pior das 4 operações, nenhum dos 2 modelos conseguiu se adaptar bem a ela, sendo necessário um treinamento melhor no modelo ou alterações em sua arquitetura.

*Esse gráfico é interativos, então é melhor visualizado direto no repositório.

MAE de ambos os modelos

O MAE (Mean Absolute Error), ou Erro Absoluto Médio, é uma métrica usada para avaliar a precisão de um modelo de previsão ou regressão. Ele calcula a média das diferenças absolutas entre os valores previstos e os valores reais.

Erro Absoluto Médio (MAE) nos exemplos (Original): **232.4641**

Erro Absoluto Médio (MAE) nos exemplos (Novo): **217.3965**

Erro Absoluto Total (TAE) nos exemplos (Original): **69739.2244**

Erro Absoluto Total (TAE) nos exemplos (Novo): **65218.9351**

Pode-se perceber que o modelo **Novo** obteve uma pontuação maior que o **Original**, durante o teste com 80 operações aleatórias

Conclusões

Observações finais

Dessa forma, foi possível identificar diversas estratégias para aprimorar a eficiência de um modelo, como o aumento da complexidade de sua estrutura, incorporando novas camadas e mais neurônios, além de melhorar a base de treinamento, garantindo que o modelo seja exposto a um volume maior de dados.

Também é fundamental monitorar o treinamento de diferentes maneiras e analisar as métricas de desempenho para avaliar se o modelo está de fato evoluindo de acordo com as expectativas.

É necessário também, avaliar se a nova complexidade do modelo pode realmente trazer uma melhora significativa para os resultados, ou se ela não é tão necessária como parece.

Obrigado!

