EP2 - MAC0425

Nome: Renan Ryu Kajihara, NUSP: 14605762

30 de junho de 2025

Resumo

O presente relatório descreve as atividades realizadas no Exercício Programa 2 da disciplina Inteligência Artificial (MAC0425), que consistiu na criação de uma rede neural com o objetivo de prever se, dado um conjunto de exames de um paciente, ele possui a doença COVID-19, com base na presença ou ausência de anticorpos do tipo IgG. A criação da rede neural envolveu três etapas principais, que serão descritas neste relatório: pré-processamento dos dados, construção e treinamento da rede neural e análise dos resultados.

Conteúdo

1 Introdução

A pandemia de COVID-19 trouxe desafios sem precedentes para a saúde pública em todo o mundo, ressaltando a importância crucial da ciência e da tecnologia. Em meio a tantas incertezas, uma lição permanece clara: a detecção rápida e precisa da doença é uma das ferramentas mais eficazes para salvar vidas, conter a disseminação do vírus e permitir que as comunidades se reorganizem com mais segurança.

Nesse contexto, após o estudo do funcionamento de perceptrons e redes neurais, o Exercício Programa 2 da disciplina Inteligência Artificial (MAC0425) teve como objetivo aplicar esses conceitos na criação de uma rede neural capaz de prever, a partir de um conjunto de exames, se um paciente está infectado com a COVID-19.

O presente relatório está organizado em três seções principais: metodologia, resultados, discussão e bibliografia.

2 Metodologia

2.1 Pré-Processamento dos Dados

Os dados são fundamentais para o treinamento de uma rede neural de qualidade. Nesse sentido, contar com informações bem estruturadas e organizadas é essencial para garantir o bom desempenho do modelo.

Para o treinamento e validação do modelo desenvolvido, foram utilizados dados de exames realizados no Hospital das Clínicas da Faculdade de Medicina da USP (HC). Esses dados estavam, inicialmente, organizados em duas tabelas: HC_EXAMES_1.csv, que contém os exames realizados pelos pacientes, e HC_PACIENTES_1, que reúne as informações dos pacientes. O pré-processamento foi realizado a partir dessas tabelas, resultando em uma tabela final que consolida os resultados dos exames, tendo o ID do atendimento como chave principal.

A primeira etapa do pré-processamento consistiu em ordenar a tabela de exames pelo ID do atendimento e pela data, de modo a garantir que, ao remover duplicatas, fosse mantido apenas o resultado mais recente de cada exame dentro de um mesmo atendimento.

Em seguida, o conjunto de dados foi filtrado para excluir pacientes que não realizaram o exame COVID-19 - PESQUISA DE ANTICORPOS IgG. Após essa filtragem, foi realizado o pivotamento do dataset, transformando o ID_ATENDIMENTO em índice principal; os diferentes DE_EXAME, em colunas; e os DE_RESULTADO, em valores.

É importante observar que nem todos os atendimentos incluem todos os exames possíveis. Por isso, os dados foram tratados de forma que, quando um atendimento não contivesse determinado exame, o resultado fosse preenchido com zero (para exames numéricos) ou 0,5 (para exames binários).

Em seguida, os valores não numéricos foram convertidos em valores numéricos. No caso do exame COVID-19 - PESQUISA DE ANTICORPOS IgG, o resultado Reagente foi mapeado para 1 e Não reagente, para 0. Além disso, resultados como Negativo ou Indetectável foram transformados em 0. Exames cujos resultados consistiam em informações técnicas que não podiam ser convertidas em números foram substituídos por zero.

Por fim, foi realizado o merge da tabela de exames com a tabela de pacientes, adicionando o ano de nascimento e o sexo de cada paciente. O sexo masculino foi codificado como 1, e o feminino, como 0. Para registros em que o ano de nascimento era desconhecido, adotou-se 1980 como valor padrão.

A tabela final resultante apresentou 361 colunas e 554 linhas.

2.2 Arquitetura da rede neural

A arquitetura da rede neural utilizada é composta por: uma camada de entrada que recebe um vetor com dimensão igual ao número de atributos dos dados de treinamento, totalizando 359 features após o pré-processamento — excluindo o ID do paciente e o exame COVID-19 - PESQUISA DE ANTICORPOS IgG, que é justamente a variável que a rede deve prever; doze camadas ocultas, cada uma com 256 neurônios, que aplicam uma transformação linear seguida de uma função de ativação

ReLU (Rectified Linear Unit); e uma camada de saída com 2 neurônios, correspondendo às duas classes possíveis: presença ou ausência de anticorpos IgG indicativos de infecção por COVID-19.

A rede neural utiliza a função de perda CrossEntropyLoss, adequada para tarefas de classificação, para calcular o erro entre as saídas previstas e os valores reais. O treinamento é realizado com o otimizador Stochastic Gradient Descent (SGD), com taxa de aprendizado de 0,01, ajustando iterativamente os pesos da rede para minimizar a função de perda.

O treinamento é conduzido por 1.000 épocas, utilizando a combinação da função de perda e do otimizador definidos, até que a rede aprenda a classificar corretamente, com base nos dados de entrada, se um paciente apresenta ou não anticorpos IgG.

2.3 Descrição dos experimentos

O treinamento foi realizado utilizando o processo de k-fold. Nesse sentido, os dados foram divididos em 5 folds, em que, a cada iteração, 80% dos dados foram usados para treinamento e 20% para validação. Essa divisão foi feita utilizando a função KFold da biblioteca scikit-learn, garantindo que cada subconjunto fosse utilizado uma vez como validação, enquanto os demais serviam para treinar o modelo.

Para cada fold, foram calculados os valores de True Positives, True Negatives, False Positives e False Negatives, obtidos a partir do conjunto de validação de cada fold.

3 Resultados

O número de True Positives, True Negatives, False Positives e False Negatives obtidos a partir do conjunto de validação de cada fold pode ser visto no gráfico a seguir:

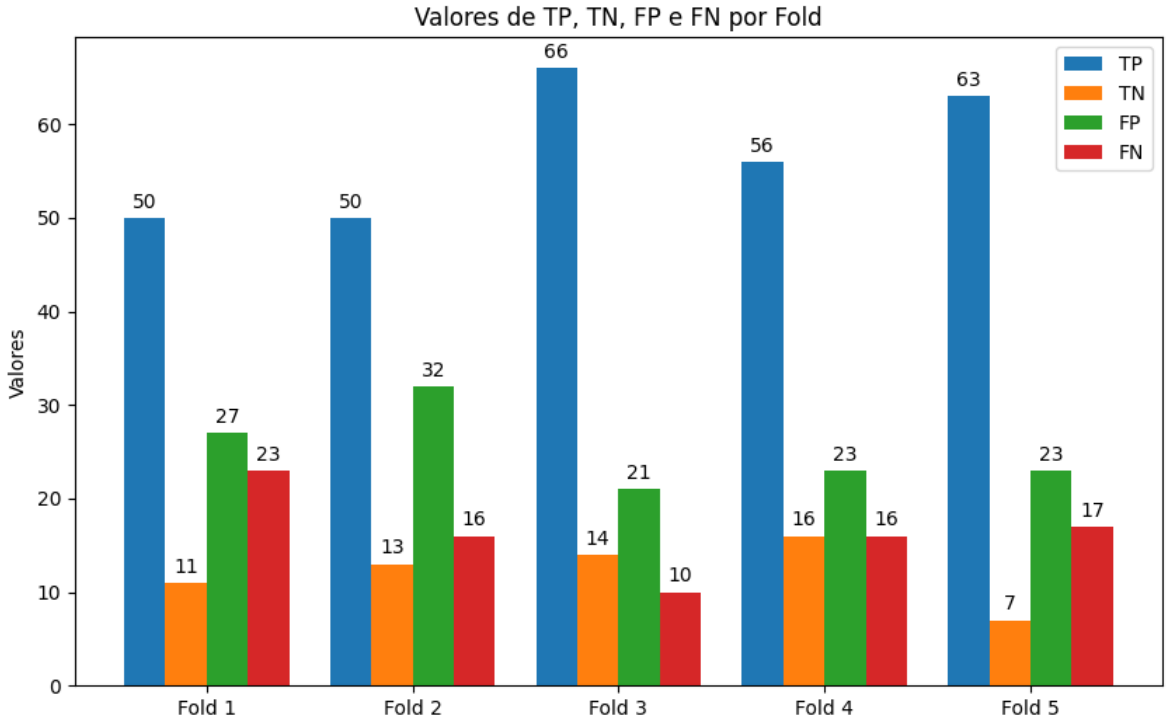

Figura 1: Número de TP, TN, FP e FN por fold.

A acurácia de cada fold pode ser vista no gráfico a seguir:

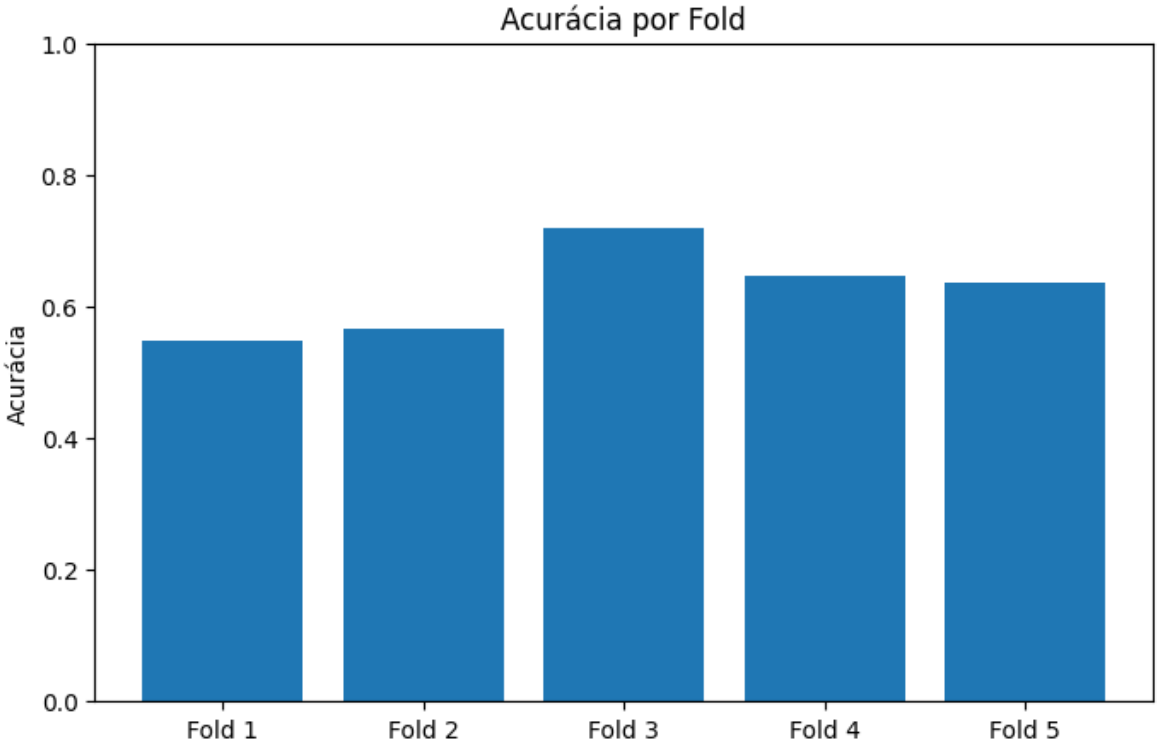

Figura 2: Acurácia por fold.

A cobertura de cada fold pode ser vista no gráfico a seguir:

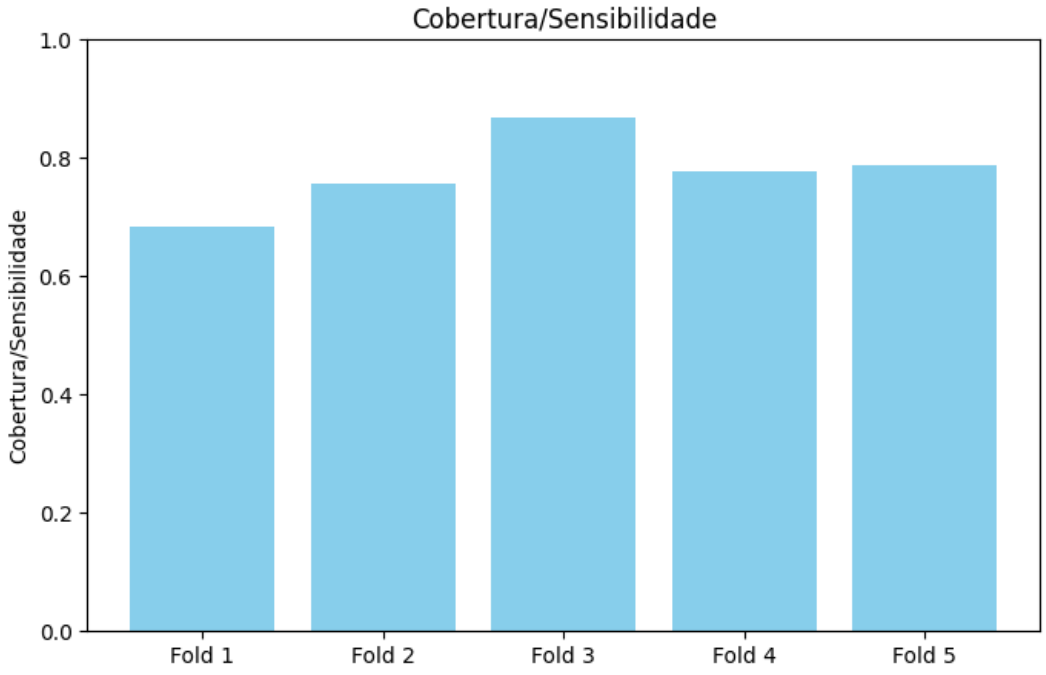

Figura 3: Cobertura por fold.

A precisão de cada fold pode ser vista no gráfico a seguir:

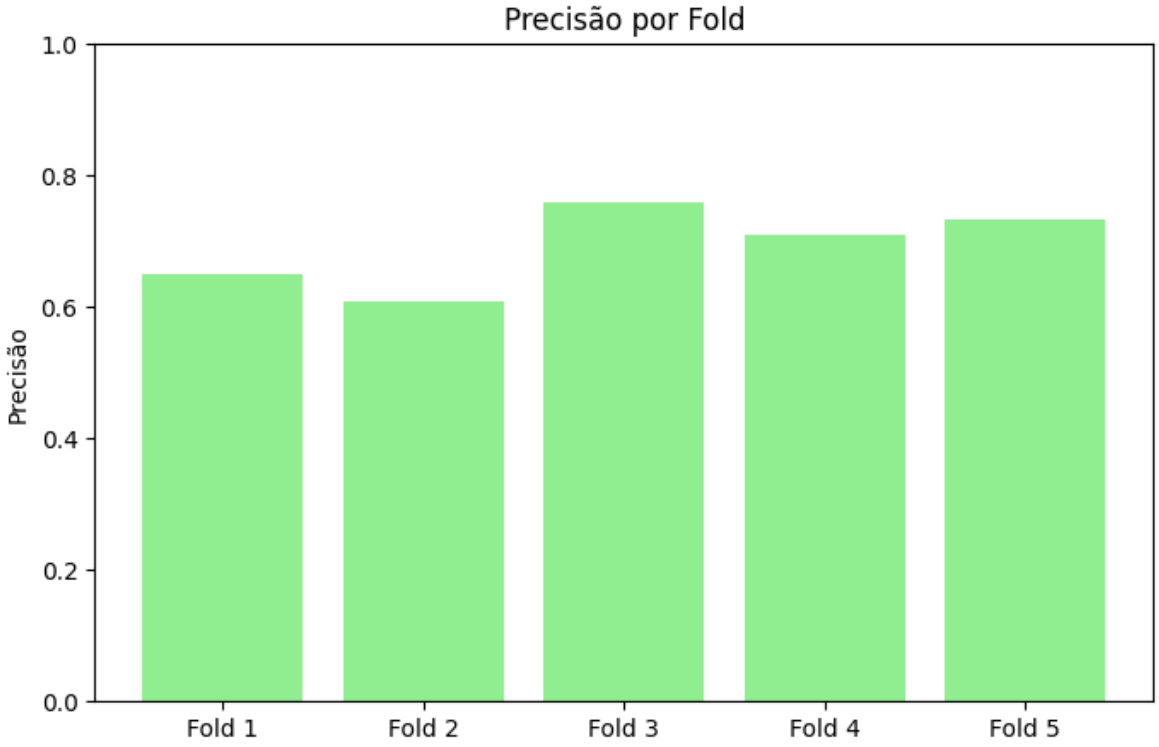

Figura 4: Precisão por fold.

A medida-F de cada fold pode ser vista no gráfico a seguir:

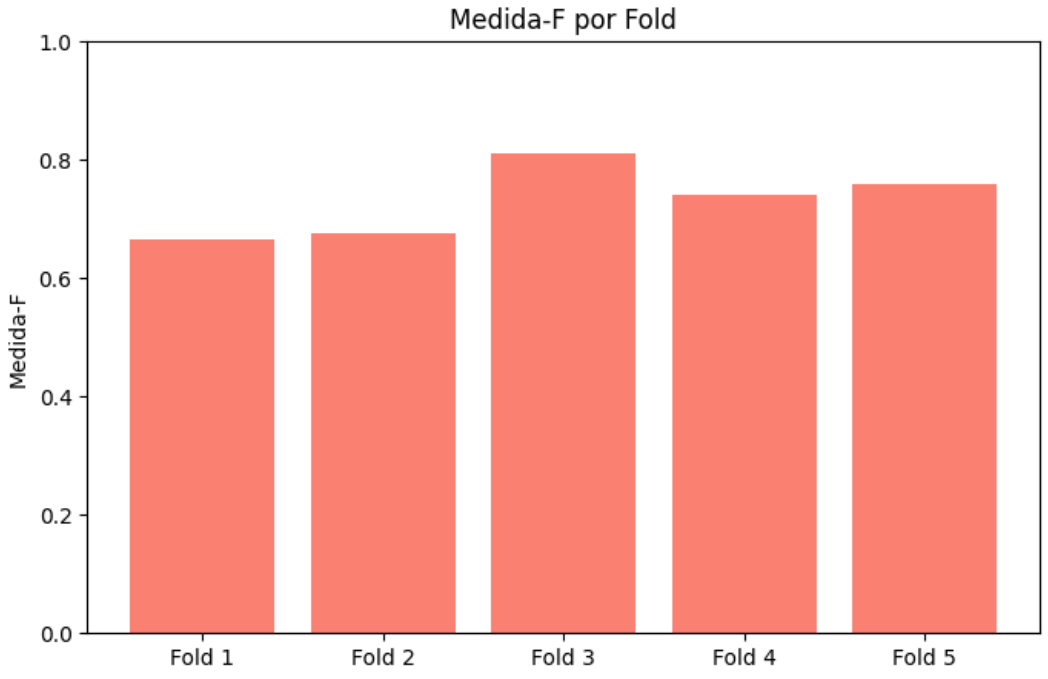

Figura 5: Medida-F por fold.

Foram calculadas as métricas de desempenho do modelo considerando os cinco folds da validação cruzada. A acurácia apresentou uma média de 62,46%, com desvio padrão de 6,14%, variando entre 54,95% (pior resultado) e 72,07% (melhor resultado).

A métrica de precisão obteve média de 69,18%, com desvio padrão de 5,47%, variando de 60,98% a 75,86%.

A cobertura (recall) apresentou desempenho superior, com média de 77,52% e desvio padrão de 5,88%, oscilando entre 68,49% e 86,84%.

Por fim, a medida-F (F1-score) teve média de 73,06%, com desvio padrão de 5,35%, apresentando melhor valor de 80,98% e pior valor de 66,67%.

4 Discussão

Após a realização dos experimentos, observou-se que, apesar da quantidade limitada de dados disponíveis (apenas 554 amostras), o modelo apresentou desempenho relativamente satisfatório nos dados de validação de cada fold. Naturalmente, em razão do tamanho reduzido do conjunto de dados e da falta da realização de exames em grande parte deles, ocorreu overfitting, o que se refletiu em um ótimo desempenho nos dados de treinamento, mas em resultados menos consistentes na validação.

Assim, conclui-se que a arquitetura neural proposta mostra-se promissora e potencialmente útil como preditora da fase de infecção, uma vez que, mesmo com poucos exemplos, apresentou resultados satisfatórios. Nesse contexto, com um volume maior de dados e informações mais completas, seria possível desenvolver um modelo baseado nessa mesma arquitetura capaz de prever a fase de infecção com maior precisão e robustez, ampliando sua utilidade em cenários reais de apoio ao diagnóstico.

5 Bibliografia

Referências

[1] Documentação do scikit-learn. Disponível em: https://scikit-learn.org/stable/supervised_learning.html.

[2] Documentação do PyTorch. Disponível em: https://docs.pytorch.org/docs/stable/index.html.