Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Transferência de Aprendizado Para Classificar Pacientes Que Tomaram a Terceira Dose da Vacina da COVID-19 no Estado do Rio de Janeiro

Fernanda Gonçalves da Silva

JUIZ DE FORA
JULHO, 2023

Transferência de Aprendizado Para Classificar Pacientes Que Tomaram a Terceira Dose da Vacina da COVID-19 no Estado do Rio de Janeiro

Fernanda Gonçalves da Silva

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Luciana Conceição Dias Campos
Coorientador: Karla Tereza Figueiredo Leite

JUIZ DE FORA
JULHO, 2023

TRANSFERÊNCIA DE APRENDIZADO PARA CLASSIFICAR PACIENTES QUE TOMARAM A TERCEIRA DOSE DA VACINA DA COVID-19 NO ESTADO DO RIO DE JANEIRO

Fernanda Gonçalves da Silva

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Luciana Conceição Dias Campos
Doutora em Engenharia Elétrica

Karla Tereza Figueiredo Leite
Doutora em Engenharia Elétrica

Luciana Brugiolo Gonçalves
Doutora em Ciência da Computação

Bárbara de Melo Quintela
Doutora em Modelagem Computacional

JUIZ DE FORA
04 DE JULHO, 2023

Aos meus amigos e irmãos.

Aos parentes, pelo apoio.

Resumo

A vacinação em massa contra a COVID-19 é essencial para combater a propagação do vírus e reduzir o número de mortes. Com a introdução de doses adicionais da vacina, é importante investigar sua eficácia e impacto na saúde dos pacientes. Este estudo utiliza um modelo de classificação para avaliar a possibilidade de evolução dos casos clínicos de pacientes que tomaram a terceira dose da vacina da COVID-19 no estado do Rio de Janeiro. O objetivo é realizar a classificação dos pacientes que receberam a terceira dose, usando transferência por aprendizado de um modelo treinado com uma base de dados balanceada. Resultados preliminares mostram que os modelos treinados com bases balanceadas superam os treinamentos com bases desbalanceadas.

Palavras-chave: Aprendizado de Máquina, Transferência de Aprendizado, Algoritmos de Classificação;

Abstract

Mass vaccination against COVID-19 is essential to combat the spread of the virus and reduce the number of deaths. With the introduction of additional doses of the vaccine, it is important to investigate their efficacy and impact on patients' health. This study uses a classification model to evaluate the possibility of clinical cases progressing in patients who received the third dose of the COVID-19 vaccine in the state of Rio de Janeiro. The objective is to classify patients who received the third dose using transfer learning from a model trained with a balanced database. Preliminary results show that models trained with balanced databases outperform those trained with imbalanced ones.

Keywords: Machine Learning, Transfer Learning, Classification Algorithms;

Agradecimentos

A todos os meus parentes e amigos, pelo encorajamento e apoio.

As professoras Luciana e Karla, por suas orientações, amizade e acima de tudo, pela paciência sem a qual este trabalho não se realizaria.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o meu enriquecimento pessoal e profissional.

"Não há fatos eternos, como não há verdades absolutas."

Friedrich Nietzsche

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AM	Aprendizado de Máquina
DCC	Departamento de Ciência da Computução
e-SUS	Sistema Único de Saúde Eletrônico
FN	Falsos Negativos
FP	Falsos Positivos
KNN	K-Nearest Neighbors
LGPD	Lei Geral de Proteção de Dados
MLP	Multi-Layer Perceptron
ROC	Receiver Operating Characteristic
TA	Transferência de Aprendizado
TCC	Trabalho de Conclusão de Curso
TN	Verdadeiros Negativos
TP	Verdadeiros Positivos
UERJ	Universidade do Estado do Rio de Janeiro
UFJF	Universidade Federal de Juiz de Fora

1 Introdução

Este capítulo aborda o problema da pandemia de COVID-19, destacando sua origem, sintomas e impacto no estado do Rio de Janeiro. Explica também a motivação, justificativa e objetivo do estudo.

A pandemia de COVID-19 teve origem em Dezembro de 2019 na cidade de Wuhan, na China, quando foi identificado um surto de pneumonia causado por um beta-coronavírus desconhecido. A Organização Mundial da Saúde (OMS) designou esse novo coronavírus como 2019-nCoV em janeiro de 2020, e em fevereiro do mesmo ano foi oficialmente nomeado como SARS-CoV-2, sendo a doença resultante denominada COVID-19 (GUO et al., 2020). A primeira notificação de transmissão local no Brasil foi declarada em 5 de março de 2020, seguida pela confirmação da primeira morte na cidade do Rio de Janeiro (RJ) em 20 de março de 2020. Entre os estados brasileiros, o Rio de Janeiro ocupa uma posição central nessa estatística, apresentando a segunda maior taxa de mortalidade (BERNARDO; ROSARIO; CONTE-JUNIOR, 2021). Até 18 de Julho de 2023, o Brasil registrou 37.693.506 casos confirmados e 704.320 óbitos relacionados à COVID-19 (GOV, 2023).

Os sintomas da COVID-19 englobam distúrbios sistêmicos, como febre, tosse e distúrbios respiratórios, incluindo pneumonia, que são semelhantes aos beta-coronavírus humanos anteriores (BERNARDO; ROSARIO; CONTE-JUNIOR, 2021). Conforme um estudo liderado pela equipe do Prof. Nan-Shan Zhong, que envolveu a análise de 1.099 casos confirmados em laboratório, foi observado que as manifestações clínicas comuns da COVID-19 incluíam febre (88,7%), tosse (67,8%), fadiga (38,1%), produção de escarro (33,4%), falta de ar (18,6%), dor de garganta (13,9%) e dor de cabeça (13,6%). Além disso, uma parcela dos pacientes também apresentou sintomas gastrointestinais, como diarreia (3,8%) e vômitos (5,0%). Os idosos e pessoas com condições subjacentes, como hipertensão, doença pulmonar obstrutiva crônica, diabetes e doença cardiovascular, tinham maior probabilidade de desenvolver rapidamente síndrome do desconforto respiratório agudo, choque séptico, acidose metabólica de difícil correção, disfunção da coagulação e

evolução para óbito (GUO et al., 2020).

Durante a pandemia de COVID-19, uma forma de desinformação amplamente disseminada foi a propagação de campanhas antivacinas (BARCELOS et al., 2021). A resposta ineficaz e tardia ao COVID-19 no Brasil foi surpreendente, considerando o histórico do país em responder de forma eficiente e ágil a políticas e serviços bem-sucedidos, bem como no controle de riscos à saúde relacionados ao tabagismo, HIV/AIDS e, mais recentemente, o Zika vírus. Apesar da postura frágil das autoridades federais, diversos estados brasileiros, como Bahia e São Paulo, e cidades como Belo Horizonte e Niterói, lideraram uma série de medidas não farmacológicas para lidar com as epidemias, incluindo bloqueios totais ou parciais, isolamento social, divulgação de informações consistentes e controle do distanciamento seguro em locais públicos (POZ; LEVCOVITZ; BAHIA, 2021).

1.1 Justificativa

A vacinação em massa é uma das principais estratégias para combater a propagação da COVID-19 e reduzir o número de óbitos causados pela doença. Com a introdução de doses adicionais da vacina, surge a necessidade de investigar a eficácia dessas doses extras e seu impacto na saúde dos pacientes. Este estudo é relevante, pois busca explorar a relação entre a terceira dose da vacina da COVID-19 e os desfechos clínicos dos pacientes no estado do Rio de Janeiro. A análise desses dados pode fornecer informações valiosas para apoiar a tomada de decisões por parte das autoridades de saúde, permitindo uma melhor compreensão sobre a eficácia e a segurança das doses adicionais da vacina.

1.2 Motivação

Este trabalho visa investigar a relação entre a evolução da COVID-19 e a vacinação por meio da análise de dados disponibilizados pelo e-SUS no estado do Rio de Janeiro. A motivação para este estudo baseia-se na importância de aprimorar os métodos de classificação de pacientes que receberam a terceira dose da vacina contra a COVID-19. Ao construir um modelo de aprendizado de máquina capaz de realizar a classificação com base em uma base de dados balanceada, espera-se obter resultados mais confiáveis e preci-

sos. Com a utilização de técnicas de transferência de aprendizado, é possível aproveitar o conhecimento prévio adquirido. Isso contribuirá para a melhoria da eficiência das campanhas de vacinação, permitindo uma identificação mais rápida e precisa dos pacientes que estão em maior risco de complicações graves. Em resumo, este trabalho busca contribuir para a área de saúde pública ao fornecer um modelo de classificação capaz de auxiliar na tomada de decisões e no monitoramento da eficácia das doses adicionais da vacina da COVID-19 no estado do Rio de Janeiro.

1.3 Objetivo

O objetivo deste trabalho é realizar a classificação dos pacientes que receberam a terceira dose da vacina contra a COVID-19 no estado do Rio de Janeiro, utilizando a técnica de aprendizado por transferência de uma base de dados balanceada. Serão treinadas duas bases de dados balanceadas: uma contendo 438 casos de óbito e 438 casos de cura de pacientes que receberam a primeira e segunda dose da vacina, e outra contendo 6.126 casos de cura e 6.106 casos de óbito de pacientes não vacinados. O modelo mais adequado, resultante do treinamento com essas duas bases, será utilizado para classificar os pacientes da base que receberam a terceira dose e que está desbalanceada, com o propósito de determinar se seus casos podem evoluir para recuperação ou óbito.

2 Fundamentação Teórica

Este capítulo aborda diferentes conceitos e técnicas relacionadas a aprendizado de máquina. Ele discute o uso de algoritmos de aprendizado supervisionado, para classificar dados em diferentes categorias. Também aborda o problema de bases de dados desbalanceadas e sugere maneiras de contornar esse problema. Além disso, o texto explora a técnica de transferência de aprendizado, que permite aproveitar conhecimentos prévios de modelos treinados em domínios relacionados para melhorar o desempenho em novos domínios.

2.1 Aprendizado de Máquina

As técnicas de Aprendizado de Máquina (AM) são fundamentadas no princípio de inferência chamado indução, que consiste em obter conclusões gerais a partir de um conjunto específico de exemplos. No contexto do aprendizado supervisionado, um professor externo fornece o conhecimento do ambiente por meio de conjuntos de exemplos contendo entradas e saídas desejadas. O algoritmo de AM utiliza esses exemplos para extrair uma representação do conhecimento (ALBUQUERQUE, 2019). São utilizadas técnicas estatísticas e matemáticas para extrair padrões dos dados, e com isso o modelo pode servir como uma ferramenta de previsões (TAN, 2021). O objetivo é que essa representação seja capaz de produzir saídas corretas para novas entradas não vistas anteriormente (ALBUQUERQUE, 2019).

De acordo com a literatura, no campo da inteligência artificial, o AM combina estatística e ciência da computação para desenvolver algoritmos que se tornam eficientes (ALBUQUERQUE, 2019). Uma representação visual dessa abordagem pode ser observada na Figura 2.1, em que os classificadores são considerados como uma funçãao f(x) que recebe um conjunto de valores X e produz uma predição Y (ALBUQUERQUE, 2019).

O modelo utilizado descreve que, dada uma entrada de dados de X1 a Xn, em que cada dado Xi possui m atributos representados como Xi = (Xi1, ..., Xim), as variáveis

Yi representam as classes (LORENA; CARVALHO, 2007).

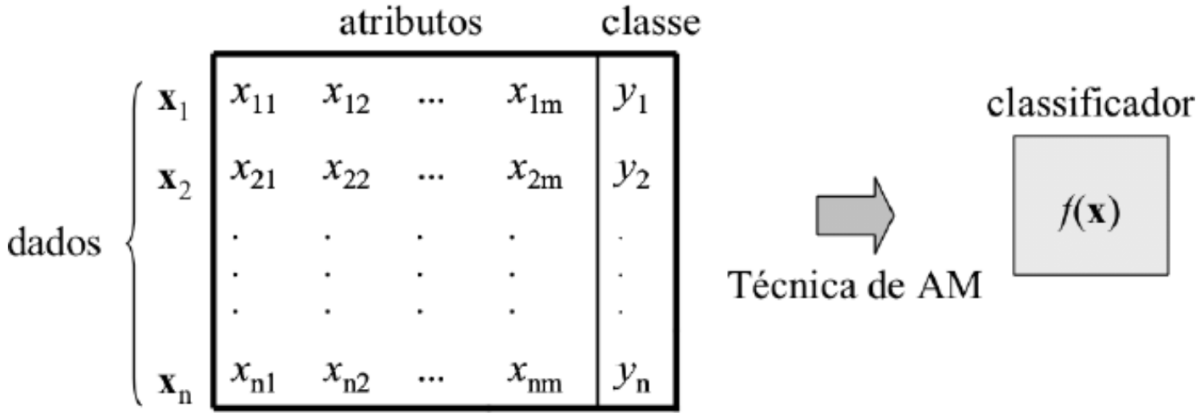

Figura 2.1: Criação de um classificador em aprendizado supervisionado (LORENA; CARVALHO, 2007)

2.1.1 Algoritmos de Classificação

Os algoritmos de classificação são utilizados para identificar classes (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Segundo a literatura, a classificação dos dados pode ser feita levando em consideração suas características, as quais podem ser binárias, categóricas ou contínuas (TAN, 2021). O modelo é treinado utilizando uma parte da base de dados que está sendo analisada para treinar o modelo e depois a sua acurácia é avaliada utilizando os dados da outra parte da base (KOTSIANTIS, 2007).

Uma abordagem comumente utilizada é a do aprendizado supervisionado, na qual o algoritmo é treinado com exemplos que possuem rótulos conhecidos, representando as classes corretas correspondentes (TAN, 2021). A principal finalidade dos algoritmos supervisionados é desenvolver um modelo capaz de atribuir rótulos semelhantes aos dados de treinamento e, ao mesmo tempo, ser capaz de generalizar efetivamente para dados não vistos. Isso significa que esses algoritmos são projetados para aprender com exemplos rotulados e utilizar esse conhecimento para fazer previsões precisas em novos dados não rotulados (DAI; ZHANG; WU, 2016).

Formalmente, o modelo deve aprender um mapeamento de entradas x para saídas y, onde $y \in \{1, ..., A\}$, sendo A o número de classes. Quando $A = 2$, isso significa que é uma classificação binária e na maioria das vezes y será um valor verdadeiro ou falso. Se

$A > 2$, então a classificação será multiclasse (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.1.2 O Problema das Bases de Dados Desbalanceadas para Algoritmos de Classificação

Entende-se como uma base de dados desbalanceada, aquelas que apresentam maior quantidade de amostras de uma classe quando comparada com as outras demais classes, impactando de forma negativa na acurácia dos resultados gerados pelo modelo treinado pois o modelo tende a ser tendencioso para a classe majoritária (ALPAYDIN, 2010). Esse é um dos problemas de algoritmo de classificação pois bases de dados desbalanceadas prejudicam algoritmos de classificação devido a três razões principais: viés na aprendizagem, baixa sensibilidade às classes minoritárias e influência dos custos de classificação. Isso leva a menor precisão nas classes minoritárias, dificuldade em identificar seus exemplos corretamente e negligência das consequências graves da classificação errônea dessas classes.

Duas maneiras mais comuns de contornar esse problema podem ser usar métodos que possam lidar com conjuntos de dados desbalanceados ou pré-processar os dados para ajustar esse desbalanceamento, duplicando as classes de menores instâncias ou removendo instâncias da classe majoritária (ALPAYDIN, 2010).

2.1.3 Árvores de Decisões

A árvore de decisão é um algoritmo de classificação que apresenta várias vantagens. Ele é capaz de gerar regras compreensíveis e lidar bem com diferentes tipos de dados, evitando cálculos complexos (ALBUQUERQUE, 2019). A construção de uma árvore de decisão requer a escolha criteriosa de uma avaliação lógica ou atributo adequado. Para um conjunto de exemplos, existem várias opções possíveis. No entanto, pesquisas indicam que, em geral, árvores menores apresentam melhor capacidade de previsão. O segredo para construir uma árvore de decisão mais compacta está em selecionar o ramo apropriado com base no atributo escolhido (DAI; ZHANG; WU, 2016). As árvores de decisão são amplamente utilizadas devido à sua análise simples e capacidade de fornecer precisão em

diferentes tipos de dados (JIJO; ABDULAZEEZ, 2021).

Uma árvore de decisão classifica uma instância ao analisar características decisivas e percorrer um caminho da raiz até as folhas para determinar sua classe ou categoria (BREIMAN, 2001). Cada árvore de decisão é composta por nós e ramos, em que cada nó representa características relacionadas a uma categoria a ser classificada e cada subconjunto define um valor possível para o nó (JIJO; ABDULAZEEZ, 2021). Cada ramo é um valor que pode ser assumido por aquele nó. A partir do nó raiz, a instância é classificada conforme a ordenação dos seus valores e características. O nó raiz é a característica que melhor divide os dados de treinamento. Existem diferentes métodos para encontrar a melhor característica mas não há um único melhor método (ALBUQUERQUE, 2019).

No entanto, é importante considerar algumas desvantagens, como a dificuldade de prever valores contínuos e a necessidade de um pré-processamento cuidadoso em dados cronológicos. Adicionalmente, quando há um grande número de categorias, os erros podem aumentar de forma mais rápida (ALBUQUERQUE, 2019). Podem enfrentar dificuldades em lidar com obstáculos que requerem particionamento diagonal, ou seja, quando os dados possuem uma relação não linear entre as características de entrada e a classe de saída. Essa situação ocorre quando os padrões e relações entre as variáveis não podem ser representados de forma eficiente por linhas retas ou planos. Portanto, em tais casos, as árvores de decisão tradicionais podem não ser eficazes (TAN, 2021).

2.1.4 Floresta aleatória

A floresta aleatória(ou *random forest*) faz parte dos algoritmos de classificação baseados em árvores de decisões e propõe a criação de várias árvores de decisão usando subconjuntos aleatórios de dados (DAI; ZHANG; WU, 2016). Essa abordagem possibilita a classificação de uma instância não apenas com base em uma única árvore, mas sim em uma coleção de árvores, formando assim uma estrutura chamada de floresta (ALBUQUERQUE, 2019). Durante a construção de cada árvore, é selecionado aleatoriamente um subconjunto de m variáveis a partir do conjunto original de variáveis, sendo utilizada a melhor divisão baseada nessas m variáveis (LIPARAS et al., 2014). Quando essas árvores são combinadas, formam o classificador floresta aleatória. (TAN, 2021).

Como podemos ver na Figura 2.2, quando usamos o algoritmo de floresta aleatória para resolver problemas de classificação, cada árvore gerada vai informar uma classe de saída. (DAI; ZHANG; WU, 2016). Cada árvore na floresta aleatória emite seu próprio voto para determinar a classe mais provável. Esses votos são contabilizados e a classe com o maior número de votos é escolhida como a classificação final para a amostra (TAN, 2021).

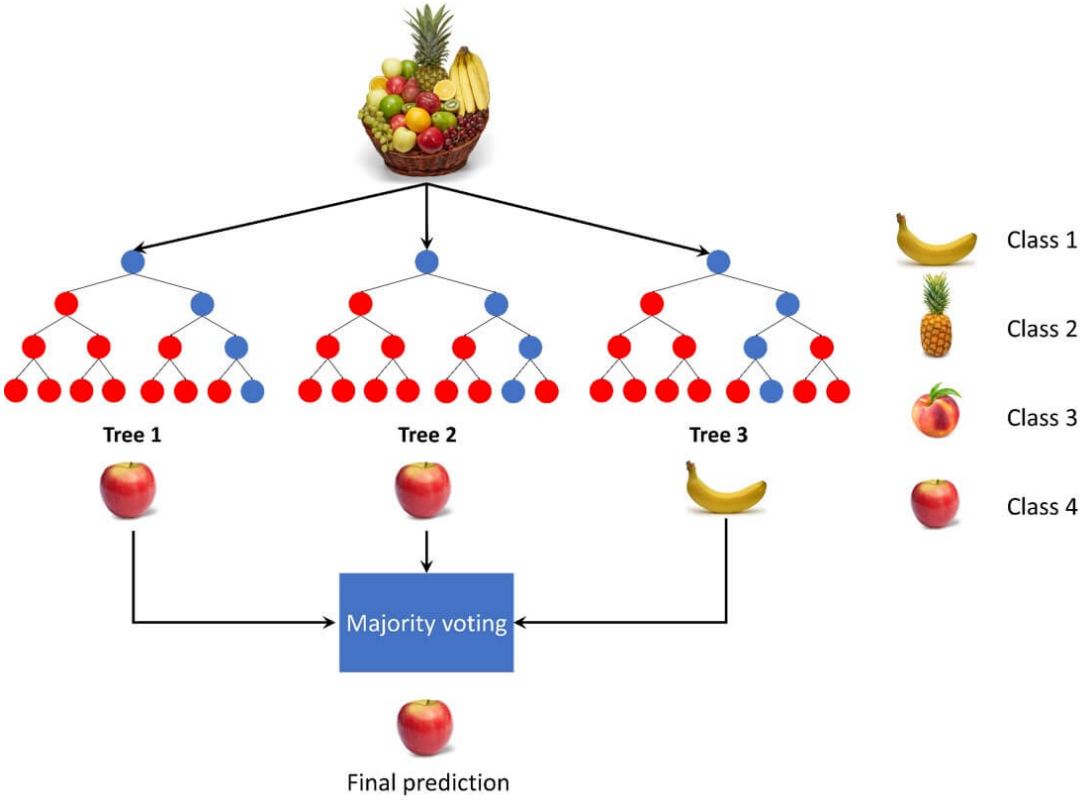

Figura 2.2: Como Funciona o Algoritmo de Floresta Aleatória (CONTRIBUTOR, 2022)

2.1.5 XGBoost(*Extreme Gradient Boosting*)

O *XGBoost* é uma biblioteca de aprendizado de máquina que implementa uma extensão do algoritmo de *boosting* de gradientes clássico, que combina várias árvores de decisão fracas para criar um modelo forte. Ele usa a técnica de boosting para iterativamente adicionar árvores ao modelo, onde cada nova árvore é ajustada para corrigir os erros residuais das árvores anteriores. Dessa forma, o *XGBoost* melhora o desempenho do modelo ao aprender com os erros anteriores e focar nas áreas mais difíceis para o algoritmo aprender (CHEN; GUESTRIN, 2016).

O *XGBoost* é popular devido a várias razões. Primeiro, ele é altamente eficiente e rápido, sendo otimizado para lidar com grandes conjuntos de dados por meio de paralelização de cálculos e amostragem eficiente. Segundo, incorpora técnicas de regularização para evitar *overfitting* e controlar a complexidade do modelo. Terceiro, o XGBoost lida com características ausentes sem pré-processamento e suporta diferentes tipos de características, como numéricas e categóricas. Além disso, fornece uma medida de importância das características para entender sua relevância. Por fim, o *XGBoost* é versátil e pode ser usado em várias tarefas de aprendizado de máquina (CHEN; GUESTRIN, 2016).

2.1.6 KNN(*K-Nearest Neighbors*)

O KNN é um algoritmo de aprendizado de máquina muito utilizado para classificação. Ele é baseado no princípio de que objetos semelhantes estão próximos uns dos outros e atribui a um ponto de amostra não classificado a classificação do vizinho mais próximo de um conjunto de pontos previamente classificados (COVER; HART, 1967).

No KNN, a escolha do número de vizinhos (representado por K) é essencial. Para classificação, os K vizinhos mais próximos são selecionados com base em uma medida de distância, como a distância euclidiana. A classe mais frequente entre os vizinhos é atribuída a amostra (COVER; HART, 1967). Podemos ver na Figura 2.3 dois exemplos do funcionamento do algoritmo para k=3 e para k=5.

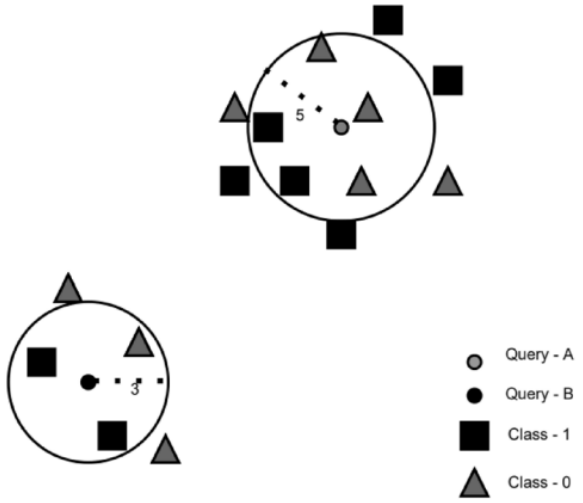

Figura 2.3: Ilustração visual do algoritmo KNN (UDDIN et al., 2022).

Uma vantagem do KNN é a sua simplicidade. No entanto, ele pode ser computacionalmente intensivo, principalmente quando o conjunto de dados de treinamento é grande. Além disso, a escolha adequada do valor de K e a seleção de uma medida de distância apropriada são importantes para o desempenho do algoritmo (COVER; HART, 1967).

2.1.7 MLP(*Multilayer Perceptron*)

O MLP é uma arquitetura de rede neural artificial que consiste em múltiplas camadas de neurônios interconectados, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio em uma camada está conectado aos neurônios na camada seguinte por meio de pesos sinápticos (HAYKIN, 2008), como podemos ver um exemplo na Figura 2.4.

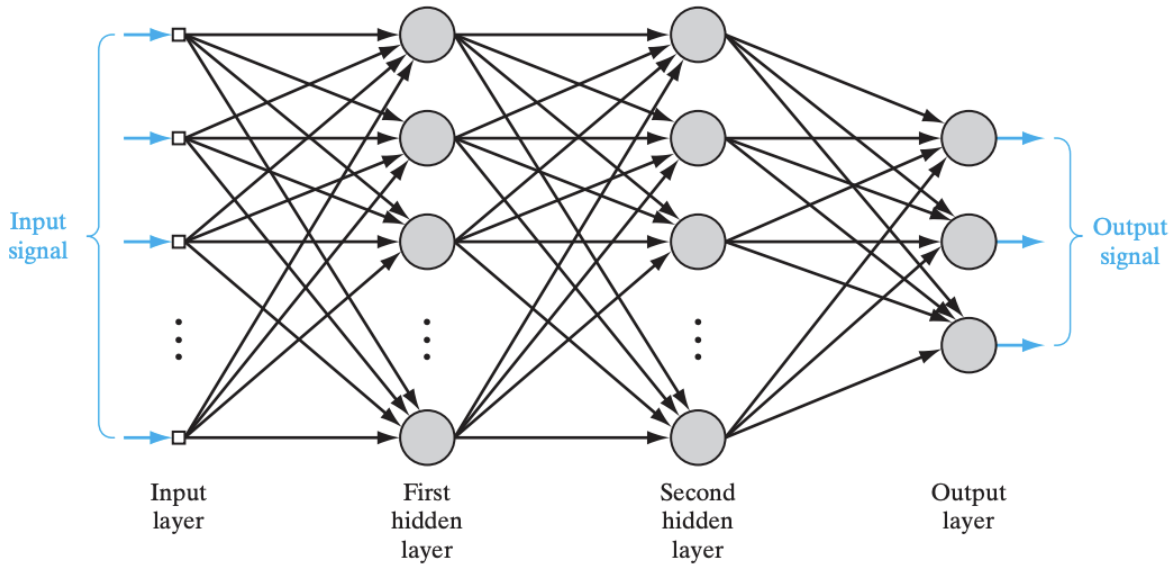

Figura 2.4: Exemplo de MLP com duas camadas ocultas (HAYKIN, 2008).

O objetivo do MLP para classificação é aprender um mapeamento entre os recursos de entrada e as classes de saída. Durante o treinamento, os pesos sinápticos da rede são ajustados com base no erro entre as classes previstas e as classes reais, utilizando o algoritmo de retropropagação de erro (HAYKIN, 2008).

Uma vez treinado, o MLP pode ser usado para classificar novos exemplos, onde os valores dos recursos de entrada são alimentados na rede, propagados pelas camadas ocultas e a saída é obtida na camada de saída. A classe prevista é geralmente determinada

pelo neurônio com o maior valor de ativação na camada de saída (HAYKIN, 2008).

2.1.8 Validação Cruzada

A validação cruzada (*cross-validation*) é uma técnica usada para avaliar o desempenho de modelos de aprendizado de máquina. Ela envolve dividir o conjunto de dados em partes menores chamadas *folds* e treinar o modelo várias vezes, usando diferentes combinações de treinamento e teste. Após cada iteração, o desempenho é avaliado usando métricas definidas nos parâmetros. Ao final, a média das métricas é calculada para obter uma estimativa mais robusta do desempenho do modelo. Essa abordagem é adotada para evitar o overfitting e ajudar na seleção de hiperparâmetros e escolha do melhor modelo para o problema em questão (SCIKIT-LEARN, Acesso em 1 de julho de 2023).

2.1.9 Grid Search

O *Grid Search* é uma técnica de busca em grade para encontrar a combinação ideal de hiperparâmetros de um modelo de aprendizado de máquina, evitando a necessidade de ajuste manual e a tentativa e erro. Essa técnica permite definir uma grade de valores para cada hiperparâmetro que se deseja otimizar e então realiza uma busca exaustiva por todas as combinações possíveis desses valores. No final, ele retorna a combinação de hiperparâmetros que obteve o melhor desempenho (SCIKIT-LEARN, Acesso em 9 de julho de 2023).

2.1.10 Transferência de Aprendizado

O aprendizado de máquina tradicional é caracterizado treinando e testando entradas com os mesmos dados de distribuição (BREIMAN, 2001). De acordo com a literatura, a transferência de aprendizado tem como objetivo aprimorar a compreensão da tarefa atual estabelecendo conexões com outras tarefas realizadas em momentos diferentes, mas dentro de um domínio de origem relacionado (HOSNA et al., 2022). A transferência de aprendizado pode ser utilizada para aprimorar os resultados de um modelo de destino, capitalizando a relação existente entre os domínios considerados (WEISS; KHOSHGOFTAAR; WANG, 2016). Ela é utilizada com o objetivo de melhorar um modelo em um

domínio específico, através da transferência de informações provenientes de um domínio relacionado (WEISS; KHOSHGOFTAAR; WANG, 2016). Essa abordagem tem se mostrado eficaz quando os recursos são gerais, ou seja, adequados tanto para as tarefas base quanto para as tarefas de destino, em vez de serem específicos apenas para a tarefa base (YOSINSKI et al., 2014).

A transferência de aprendizado serve para solucionar problemas de classificação que são similares e pode ser aplicado como uma alternativa para melhorar o desempenho do modelo que está sendo treinado para um determinado domínio, baseado em um treinamento já realizado anteriormente para um domínio diferente(KONONENKO; KUKAR, 2007). Podemos recorrer a dois exemplos do mundo real para compreender a transferência de aprendizado. Um exemplo é o reconhecimento de objetos em imagens, em que um modelo pré-treinado em um grande conjunto de dados, como o ImageNet, pode ser transferido para tarefas específicas, como o reconhecimento de objetos em um conjunto de dados menor e mais especializado.

O modelo pré-treinado serve como ponto de partida para a tarefa de aprendizado na nova base de dados, permitindo aproveitar o conhecimento prévio e acelerar o processo de aprendizado nesse novo contexto, como podemos observar na Figura 2.5.

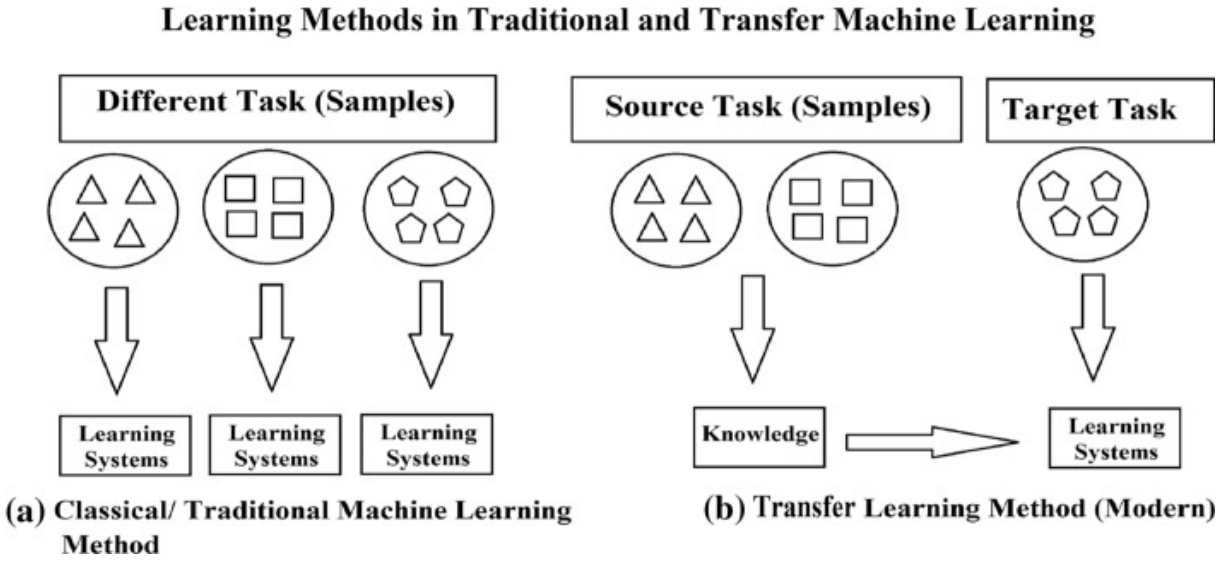

Figura 2.5: Aprendizado de Máquina Clássico x Transferência de Aprendizado (HOSNA et al., 2022)

Devido à escassez, inacessibilidade, alto custo e desafios associados à coleta de dados no mundo contemporâneo, as pessoas enfrentam dificuldades em investir recursos

nessa atividade (HOSNA et al., 2022). Em certos cenários, é desafiador e custoso obter dados de treinamento que estejam alinhados com o espaço de características e a distribuição dos dados de teste. Como resultado, existe uma demanda por modelos de alto desempenho que possam ser treinados em um domínio-fonte relacionado e aplicados em um domínio-alvo específico (WEISS; KHOSHGOFTAAR; WANG, 2016). Assim, a transferência de conhecimento entre tarefas surge como uma abordagem mais eficiente e viável como alternativa à coleta de dados (HOSNA et al., 2022).

2.2 Análise dos dados

As métricas utilizadas para avaliar o desempenho dos modelos desenvolvidos neste TCC são apresentadas a seguir.

2.2.1 Matriz de Confusão

A matriz de confusão é uma tabela que mostra as predições do modelo em relação às classes reais. Ela permite analisar os resultados em termos de verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN), como pode ser visto na Tabela 2.1.

Isso significa que se uma classe tem como saída um valor falso e o modelo classifica essa classe como falsa, ele fez uma classificação da classe falsa corretamente e por isso esse resultado entra na matriz como TN. Se uma classe tem como saída um valor falso e o modelo classifica essa classe como verdadeira, ele fez uma classificação da classe falsa incorretamente e por isso esse resultado entra na matriz como FP. Se uma classe tem como saída um valor verdadeiro e o modelo classifica essa classe como verdadeira, ele fez uma classificação da classe verdadeira corretamento e por isso esse resultado entra na matriz como TP. E por fim, se uma classe tem como saída um valor verdadeiro e o modelo classifica essa classe como falsa, ele fez uma classificação da classe verdadeira incorretamente e por isso esse resultado entra na matriz como FN.

Tabela 2.1: Matriz de Confusão

	Predição Negativa	Predição Positiva
Classe Negativa	TN	FP
Classe Positiva	FN	TP

2.2.2 Precisão

A precisão é uma métrica que representa a proporção de verdadeiros positivos em relação ao total de predições positivas feitas pelo modelo.

$$Precisão = \frac{\text{TP}}{\text{TP + FP}}$$

2.2.3 Revocação (*Recall*)

A revocação ou *recall* é a taxa de verdadeiros positivos, que representa a proporção de verdadeiros positivos em relação ao total de amostras da classe positiva.

$$Revocação = \frac{\text{TP}}{\text{TP + FN}}$$

2.2.4 *F1-Score*

O *F1-Score* é uma métrica que combina a precisão e a revocação em uma única medida, fornecendo uma média harmônica entre as duas.

$$F1 - Score = 2 * \frac{\text{Precisão * Revocação}}{\text{Precisão + Revocação}}$$

2.2.5 Área sob a Curva ROC

A AUC-ROC é uma métrica utilizada principalmente em problemas de classificação binária para avaliar o desempenho do modelo em diferentes limiares de classificação. Ela mede a capacidade do modelo de distinguir entre classes positivas e negativas.

2.2.6 Acurácia

A acurácia é uma métrica simples e amplamente utilizada para problemas de classificação. Ela representa a proporção de predições corretas em relação ao total de predições feitas pelo modelo.

$$Acurácia = \frac{\text{Número de Previsões Corretas}}{\text{Total de Previsões}}$$

3 Metodologia

3.1 Base de dados

A base de dados utilizada para a realização deste estudo foi retirada do e-SUS, com dados de notificação relativos ao estado do Rio de Janeiro. Os municípios envolvidos foram: Angra dos reis, Aperibé, Araruama, Areal, Armação de Búzios, Arraial do cabo, Barra do Piraí, Barra Mansa, Belford Roxo, Bom Jesus do Itabapoana, Cabo Frio, Cachoeiras de Macacu, Campos dos Goytacazes, Cantagalo, Carapebus, Cardoso Moreira, Carmo, Casimiro de Abreu, Cordeiro, Duque de Caxias, Engenheiro Paulo de Frontin, Guapimirim, Itaboraí, Itaguaí, Italva, Itaocara, Itaperuna, Itatiaia, Macaé, Macuco, Magé, Mangaratiba, Maricá, Mendes, Miguel Pereira, Nilópolis, Niterói, Nova Friburgo, Nova Iguaçu, Paracambi, Petrópolis, Pinheiral, Piraí, Porciúncula, Porto Real, Queimados, Quissamã, Resende, Rio Bonito, Rio Claro, Rio das Ostras, Rio de Janeiro, Santo Antônio de Pádua, São Francisco de Itabapoana, São Gonçalo, São João da Barra, São João de Meriti, São José de Ubá, São José do Vale do Rio Preto, Sapucaia, Saquarema, Seropédica, Silva Jardim, Sumidouro, Tanguá, Teresópolis, Trajano de Moraes, Valença, Varre-sai, Vassouras, Volta Redonda.

Essa base de dados contém informações pessoais e relacionadas ao estado de saúde do paciente, tais como idade, distúrbios gustativos, presença de sintomas, incluindo tosse, coriza, dor de cabeça, dispneia, febre, distúrbios olfativos, dor de garganta, doença respiratória crônica, doença renal crônica, doença cromossômica, doença cardíaca crônica, puérpera, imunossupressão, diabetes, gestação, obesidade, outras condições e um valor de verdadeiro ou falso para a evolução da doença, informando se o paciente se recuperou ou não, onde 1 indica a cura do paciente e 0 indica óbito.

Para realizar o treinamento dos modelos utilizamos uma base com 438 casos de óbito e 438 casos de sobrevivência, ou seja, era uma base balanceada que continha pacientes que tomaram a primeira e a segunda dose da vacina e outra base de dados também balanceada contendo 6106 casos de óbito e 6126 casos de sobrevivência que continha pa-

cientes que não tomaram nenhuma dose da vacina. Após a análise dos resultados, os melhores modelos de cada base foram utilizados para realizar a transferência de aprendizagem para a classificação de uma outra base de dados que continha 61.288 casos de cura e 106 casos de óbito de pacientes que tomaram até a terceira dose da vacina.

Os dados dessas bases foram extraidos da base de dados armazenados no sistema da Secretária de saúde do Estado do Rio de Janeiro, com data de 02/02/2023 contendo 12,9 GB, com 11.152.822 linhas e 144 colunas com permissões de uso regidas pelo convenio entre a secretária de saúde do Rio de janeiro e a UERJ. A base passou por várias etapas de pré-processamento para garantir a proteção dos dados sensíveis dos pacientes e de acordo com o que é estabelecido pela LGPD(LGPD, 2023), fornecer um conjunto de dados mais adequado para análise.

Para melhorar a qualidade e a utilidade dos dados, foram executadas diversas etapas de limpeza e transformação. Os registros duplicados foram removidos, garantindo a existência de registros únicos na base de dados. Além disso, foram aplicados filtros para selecionar apenas casos confirmados de COVID-19, descartando notificações de casos que não estavam realmente infectados.

Uma coluna adicional denominada idade foi calculada com base nas datas de nascimento dos pacientes, permitindo uma análise mais detalhada da distribuição etária dos casos. Além disso, a característica evolucaoCaso foi utilizada para identificar os desfechos dos pacientes, com foco nos casos de cura e óbito.

Da mesma forma, a coluna dosesVacina foi tratada para gerar colunas separadas para cada dose de vacina mencionada. Por fim, a característica sintomas foi analisada e processada para criar novas colunas indicando a presença ou ausência de cada sintoma relatado pelos pacientes.

3.2 Modelos de Classificação Aplicados

Neste estudo, todos os modelos foram treinados utilizando a linguagem Python no ambiente do Jupyter Notebook, executado em uma máquina local com um processador Intel Core i7 de 2,6 GHz e 6 núcleos. Para a manipulação dos dados, utilizamos as bibliotecas numpy para transformar os dados em estruturas de matrizes e pandas (MCKINNEY

et al., 2010) para realizar a leitura dos dados no formato Excel. A principal biblioteca utilizada para o aprendizado de máquina foi a Scikit-learn (PEDREGOSA et al., 2011), que oferece algoritmos para as principais tarefas padrões do Python e permite avaliar as métricas dos modelos treinados.

Foram treinados e avaliados diferentes modelos de classificação. Os modelos utilizados foram: KNN (K-Nearest Neighbors) com até 10 vizinhos (sendo exibido apenas o melhor resultado entre os modelos 10 treinados), MLP (Multi-Layer Perceptron), Floresta Aleatória e XGBoost.

Para realizar o treinamento desses modelos foi utilizado o GridSearchCV, que é uma técnica que encontra os melhores hiperparâmetros para um modelo por meio de várias combinações possíveis, automatizando o processo de busca e otimização.

A etapa de testes e validação dos modelos foram realizadas utilizando validação cruzada com 5 *KFolds*, ou seja, o conjunto de dados foi dividido em 5 partes iguais, sendo cada parte usada uma vez como conjunto de teste e as demais partes como conjunto de treinamento. Após as 5 iterações, foram obtidas estimativas do desempenho do modelo com base nos resultados de teste em cada *fold*.

Para cada modelo, foram calculadas diversas métricas de avaliação, incluindo acurácia, precisão, recall, F1-score e AUC-ROC. Inicialmente, os modelos foram treinados com bases de dados balanceadas. O objetivo foi analisar tanto os casos positivos quanto os casos negativos previstos pelos modelos. A métrica principal utilizada para avaliar o desempenho dos modelos treinados com bases balanceadas foi a acurácia.

4 Estudo de caso

Neste capítulo, apresentamos os resultados obtidos para cada modelo, considerando as bases com pacientes sem nenhuma dose e com a primeira e segunda doses.

4.1 Avaliação dos Hiperparâmetros e Melhores Parâmetros Encontrados

Parâmetros são valores ajustáveis que definem o comportamento e a configuração de um modelo. Eles variam de acordo com o algoritmo e o tipo de modelo de aprendizado de máquina e representam as características e propriedades específicas do modelo. Eles são ajustados durante a etapa de treinamento para minimizar o erro ou maximizar o desempenho do modelo.

O ajuste adequado dos parâmetros pode fazer a diferença entre um modelo de aprendizado de máquina preciso e um modelo com baixo desempenho. A seleção adequada dos parâmetros é um desafio importante em aprendizado de máquina e eles podem ser ajustados manualmente ou de forma automatizada usando técnicas como busca em grade ou otimização bayesiana. Exemplos de hiperparâmetros incluem a taxa de aprendizado, o número de camadas em uma rede neural e o número de árvores em um modelo de floresta aleatória.

Os hiperparâmetros escolhidos para treinar os modelos deste projeto foram escolhidos com base em outros projetos pessoais já desenvolvidos e em alguns exemplos encontrados em *notebooks* na internet.

4.1.1 Base de Dados Sem Vacinação

A Tabela 4.1 apresenta os hiperparâmetros utilizados na configuração de um classificador Floresta Aleatória e os melhores resultados obtidos para cada um desses parâmetros. Vamos entender o significado de cada hiperparâmetro:

max_depth: Este parâmetro define a profundidade máxima das árvores na floresta. Uma maior profundidade permite que as árvores tenham mais decisões e capturam relações mais complexas nos dados, mas também aumenta o risco de *overfitting*.

min_samples_split: Determina o número mínimo de amostras necessárias para que um nó seja dividido em dois nós filhos durante a construção da árvore. Um valor menor desse parâmetro permite divisões mais frequentes, levando a árvores mais complexas e detalhadas, mas também pode aumentar o risco de *overfitting*.

n_estimators : Este parâmetro define a quantidade de árvores na floresta aleatória. Ter mais árvores geralmente resulta em um modelo mais robusto e com melhor desempenho, embora o aumento do número de árvores também possa aumentar o tempo de treinamento.

criterion: Refere-se à medida de qualidade usada para avaliar a qualidade das divisões nos nós das árvores. Alguns exemplos comuns são o índice de Gini e a entropia. Essa medida é usada para determinar como as divisões nos nós são feitas para maximizar a precisão do modelo.

max_features: Este parâmetro limita o número de variáveis consideradas em cada divisão durante a construção das árvores. Controlar o número máximo de características pode ajudar a evitar a superespecialização do modelo e melhorar sua capacidade de generalização.

min_samples_leaf: Define o número mínimo de amostras necessárias para um nó ser considerado uma folha na árvore. Ter um valor maior nesse parâmetro evita que os nós finais da árvore sejam muito pequenos e específicos para poucas amostras, ajudando a evitar overfitting.

Tabela 4.1: Avaliação de Hiperparâmetros Para o Modelo Floresta Aleatória

Parâmetro	Valores	Melhor Resultado
max_depth	[3,4, 5, 6, 7, 8, 9, 10]	10
min_samples_split	[2, 4, 8, 12, 16]	2
n_estimators	[100]	100
criterion	[gini, entropy]	gini
max_features	[auto, 2, 3, 4, 6, 8,10,12,14,16,18]	3
min_samples_leaf	[2,3,4,5,6,7,8]	3

A Tabela 4.2 apresenta os hiperparâmetros utilizados na configuração do classificador XGBoost e os melhores resultados obtidos para cada um desses parâmetros. Vamos entender o significado de cada hiperparâmetro:

max_depth : Controla a profundidade máxima das árvores no modelo *XGBoost*. Um valor maior permite que as árvores sejam mais complexas e capturam relações mais detalhadas nos dados, mas também aumenta o risco de *overfitting*.

learning_rate: É a taxa de aprendizado usada para atualizar os pesos do modelo em cada iteração do treinamento. Um valor menor resulta em ajustes mais conservadores, enquanto um valor maior pode levar a ajustes mais agressivos. A escolha da taxa de aprendizado correta é importante para encontrar o equilíbrio entre precisão e eficiência do modelo.

n_estimators: Define o número de estimadores, ou seja, o número de árvores a serem construídas no modelo *XGBoost*. Ter mais árvores geralmente melhora o desempenho, mas também aumenta o tempo de treinamento.

random_state: É um parâmetro que controla a aleatoriedade do processo de treinamento. Ao definir um valor específico para o random_state, é possível reproduzir os mesmos resultados em diferentes execuções do modelo.

booster: Indica o tipo de booster utilizado no *XGBoost*. O *XGBoost* oferece opções como "gbtree"para árvores de decisão e "gblinear"para modelos lineares generalizados.

gamma: Define a redução mínima de perda necessária para fazer uma nova divisão em uma árvore. Um valor maior de gamma torna o modelo mais conservador, evitando divisões desnecessárias e aumentando a eficiência.

base_score: É o valor inicial de previsão para todas as amostras. Pode ser usado para ajustar a taxa de aprendizado e compensar a distribuição desigual dos rótulos.

A Tabela 4.3 descreve os hiperparâmetros utilizados na configuração de um classificador MLP e os melhores resultados obtidos para cada um desses parâmetros. Vamos entender o significado de cada um deles:

Tabela 4.2: Avaliação de Hiperparâmetros Para o Modelo XGBoost

Parâmetro	Valores	Melhor Resultado
max_depth	[5,6,7]	5
learning_rate	[0.1,0.01,0.05,0.0005]	0.05
n_estimators	[100,280,300,320]	320
random_state	[1,42,50,80,101]	1
booster	[gbtree]	gbtree
gamma	[0]	0
base_score	[0.2,0.5,0.7]	0.5

hidden_layer_sizes: Esses parâmetros indicam o número e o tamanho das camadas ocultas na rede neural. Por exemplo, um valor de (100, 50) indicaria uma rede com duas camadas ocultas, a primeira com 100 neurônios e a segunda com 50 neurônios.

activation: Refere-se à função de ativação utilizada nos neurônios da rede. As funções de ativação determinam a saída dos neurônios e sua importância na propagação dos dados pela rede. Exemplos comuns são a função sigmoide, a função ReLU e a função tangente hiperbólica.

solver: É o algoritmo de otimização usado para ajustar os pesos da rede durante o treinamento. Existem vários algoritmos disponíveis, como "adam", "lbfgs"e "sgd", cada um com suas características e eficácia em diferentes situações.

alpha: É um parâmetro de regularização que controla a penalidade aplicada aos pesos da rede para evitar o *overfitting*. O valor de *alpha* afeta a complexidade do modelo e a magnitude dos pesos atribuídos a cada neurônio.

learning_rate_init e *learning_rate*: São as taxas de aprendizado usadas pelos algoritmos de otimização. A taxa de aprendizado determina o tamanho dos ajustes feitos nos pesos da rede em cada iteração do treinamento. O valor de *learning_rate_init* é a taxa de aprendizado inicial.

momentum e *nesterovs_momentum*: Esses parâmetros estão relacionados à técnica de momentum, que ajuda a acelerar o processo de treinamento. O momentum controla a contribuição dos gradientes anteriores no ajuste dos pesos, enquanto o nesterovs_momentum indica se a técnica de Nesterov deve ser utilizada.

beta_1 e *beta_2*: São parâmetros específicos do otimizador Adam, que é um algoritmo de otimização popular. Eles influenciam o cálculo dos momentos de primeira e

segunda ordem usados pelo Adam para atualizar os pesos da rede.

Tabela 4.3: Avaliação de Hiperparâmetros Para o Modelo MLP

Parâmetro	Valores	Melhor Resultado
hidden_layer_sizes	[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]	80
activation	['relu','tanh']	tanh
solver	['adam', 'sgd','lbfgs'],	sgd
alpha	[0.001, 0.05]	0.05
learning_rate	['constant','adaptive']	adaptive
nesterovs_momentum	[True, False]	True
beta_1	[0.95]	0.95
beta_2	[0.999]	0.999
learning_rate_init	[0.0001, 0.005, 0.001, 0.05]	0.0001
momentum	[0.9, 0.95]	0.95

A Tabela 4.4 apresenta os parâmetros passados para a configuração do classificador KNN e o melhor resultado obtido. O K é o número de vizinhos mais próximos a serem considerados na classificação. O melhor resultado encontrado pode ser visto na terceira coluna da Tabela 4.4.

Tabela 4.4: Avaliação de Parâmetros Para o Modelo KNN

Parâmetro	Valores	Melhor Resultado
K	1 até 10	9

4.1.2 Base de Dados Com Primeira e Segunda Doses

A Tabela 4.5 apresenta os hiperparâmetros utilizados na configuração de um classificador Floresta Aleatória e os melhores resultados obtidos para cada um desses parâmetros. Vamos entender o significado de cada hiperparâmetro:

max_depth: Este parâmetro define a profundidade máxima das árvores na floresta. Uma maior profundidade permite que as árvores tenham mais decisões e capturam relações mais complexas nos dados, mas também aumenta o risco de *overfitting*.

min_samples_split: Determina o número mínimo de amostras necessárias para que um nó seja dividido em dois nós filhos durante a construção da árvore. Um valor menor desse parâmetro permite divisões mais frequentes, levando a árvores mais complexas e detalhadas, mas também pode aumentar o risco de *overfitting*.

n_estimators : Este parâmetro define a quantidade de árvores na floresta aleatória. Ter mais árvores geralmente resulta em um modelo mais robusto e com melhor desempenho, embora o aumento do número de árvores também possa aumentar o tempo de treinamento.

criterion: Refere-se à medida de qualidade usada para avaliar a qualidade das divisões nos nós das árvores. Alguns exemplos comuns são o índice de Gini e a entropia. Essa medida é usada para determinar como as divisões nos nós são feitas para maximizar a precisão do modelo.

max_features: Este parâmetro limita o número de variáveis consideradas em cada divisão durante a construção das árvores. Controlar o número máximo de características pode ajudar a evitar a superespecialização do modelo e melhorar sua capacidade de generalização.

min_samples_leaf: Define o número mínimo de amostras necessárias para um nó ser considerado uma folha na árvore. Ter um valor maior nesse parâmetro evita que os nós finais da árvore sejam muito pequenos e específicos para poucas amostras, ajudando a evitar overfitting.

Tabela 4.5: Avaliação de Hiperparâmetros Para o Modelo Floresta Aleatória

Parâmetro	Valores	Melhor Resultado
max_depth	[3,4, 5, 6, 7, 8, 9, 10]	5
min_samples_split	[2, 4, 8, 12, 16]	12
n_estimators	[100]	100
criterion	[gini, entropy]	entropy
max_features	[auto, 2, 3, 4, 6, 8,10,12,14,16,18]	3
min_samples_leaf	[2,3,4,5,6,7,8]	2

A Tabela 4.6 apresenta os hiperparâmetros utilizados na configuração do classificador XGBoost e os melhores resultados obtidos para cada um desses parâmetros. Vamos

entender o significado de cada hiperparâmetro:

max_depth : Controla a profundidade máxima das árvores no modelo *XGBoost*. Um valor maior permite que as árvores sejam mais complexas e capturam relações mais detalhadas nos dados, mas também aumenta o risco de *overfitting*.

learning_rate: É a taxa de aprendizado usada para atualizar os pesos do modelo em cada iteração do treinamento. Um valor menor resulta em ajustes mais conservadores, enquanto um valor maior pode levar a ajustes mais agressivos. A escolha da taxa de aprendizado correta é importante para encontrar o equilíbrio entre precisão e eficiência do modelo.

n_estimators: Define o número de estimadores, ou seja, o número de árvores a serem construídas no modelo *XGBoost*. Ter mais árvores geralmente melhora o desempenho, mas também aumenta o tempo de treinamento.

random_state: É um parâmetro que controla a aleatoriedade do processo de treinamento. Ao definir um valor específico para o random_state, é possível reproduzir os mesmos resultados em diferentes execuções do modelo.

booster: Indica o tipo de booster utilizado no *XGBoost*. O *XGBoost* oferece opções como "gbtree"para árvores de decisão e "gblinear"para modelos lineares generalizados.

gamma: Define a redução mínima de perda necessária para fazer uma nova divisão em uma árvore. Um valor maior de gamma torna o modelo mais conservador, evitando divisões desnecessárias e aumentando a eficiência.

base_score: É o valor inicial de previsão para todas as amostras. Pode ser usado para ajustar a taxa de aprendizado e compensar a distribuição desigual dos rótulos.

Tabela 4.6: Avaliação de Hiperparâmetros Para o Modelo XGBoost

Parâmetro	Valores	Melhor Resultado
max_depth	[5,6,7]	5
learning_rate	[0.1,0.01,0.05,0.0005]	0.01
n_estimators	[100,280,300,320]	300
random_state	[1,42,50,80,101]	1
booster	[gbtree]	gbtree
gamma	[0]	0
base_score	[0.2,0.5,0.7]	0.7

A Tabela 4.7 descreve os hiperparâmetros utilizados na configuração de um classificador MLP e os melhores resultados obtidos para cada um desses parâmetros. Vamos entender o significado de cada um deles:

hidden_layer_sizes: Esses parâmetros indicam o número e o tamanho das camadas ocultas na rede neural. Por exemplo, um valor de (100, 50) indicaria uma rede com duas camadas ocultas, a primeira com 100 neurônios e a segunda com 50 neurônios.

activation: Refere-se à função de ativação utilizada nos neurônios da rede. As funções de ativação determinam a saída dos neurônios e sua importância na propagação dos dados pela rede. Exemplos comuns são a função sigmoide, a função ReLU e a função tangente hiperbólica.

solver: É o algoritmo de otimização usado para ajustar os pesos da rede durante o treinamento. Existem vários algoritmos disponíveis, como "adam", "lbfgs"e "sgd", cada um com suas características e eficácia em diferentes situações.

alpha: É um parâmetro de regularização que controla a penalidade aplicada aos pesos da rede para evitar o *overfitting*. O valor de *alpha* afeta a complexidade do modelo e a magnitude dos pesos atribuídos a cada neurônio.

learning_rate_init e *learning_rate*: São as taxas de aprendizado usadas pelos algoritmos de otimização. A taxa de aprendizado determina o tamanho dos ajustes feitos nos pesos da rede em cada iteração do treinamento. O valor de *learning_rate_init* é a taxa de aprendizado inicial.

momentum e *nesterovs_momentum*: Esses parâmetros estão relacionados à técnica de momentum, que ajuda a acelerar o processo de treinamento. O momentum controla a contribuição dos gradientes anteriores no ajuste dos pesos, enquanto o nesterovs_momentum indica se a técnica de Nesterov deve ser utilizada.

beta_1 e *beta_2*: São parâmetros específicos do otimizador Adam, que é um algoritmo de otimização popular. Eles influenciam o cálculo dos momentos de primeira e segunda ordem usados pelo Adam para atualizar os pesos da rede.

A Tabela 4.8 apresenta os parâmetros passados para a configuração do classifi-

Tabela 4.7: Avaliação de Hiperparâmetros Para o Modelo MLP

Parâmetro	Valores	Melhor Resultado
hidden_layer_sizes	[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]	80
activation	['relu','tanh']	tanh
solver	['adam', 'sgd','lbfgs'],	sgd
alpha	[0.001, 0.05]	0.05
learning_rate	['constant','adaptive']	adaptive
nesterovs_momentum	[True, False]	True
beta_1	[0.95]	0.95
beta_2	[0.999]	0.999
learning_rate_init	[0.0001, 0.005, 0.001, 0.05]	0.0001
momentum	[0.9, 0.95]	0.95

cador KNN e o melhor resultado obtido. O K é o número de vizinhos mais próximos a serem considerados na classificação. O melhor resultado encontrado pode ser visto na terceira coluna da Tabela 4.8.

Tabela 4.8: Avaliação de Parâmetros Para o Modelo KNN

Parâmetro	Valores	Melhor Resultado
K	1 até 10	8

4.2 Análise dos Modelos

4.2.1 Resultados do Treinamento Com a Base de Dados Sem Vacinação

Para avaliar o desempenho dos modelos treinados, utilizaremos as métricas: acurácia, precisão, revocação, f1-score e AUC-ROC. Na Tabela 4.9, referente aos resultados dos treinamentos, é possível observar que os modelos de Floresta Aleatória e *XGBoost* obtiveram os melhores resultados em termos da maioria das métricas analisadas.

Tabela 4.9: Métricas de Validação dos Modelos Treinados Com a Base de Dados Sem Vacinação

	Acurácia	Precisão	Revocação	F1-Score	AUC-ROC
Floresta Aleatória	**0.873**	0.**873**	**0.873**	**0.873**	**0.940**
XGBoost	**0.871**	0.864	**0.882**	**0.873**	**0.943**
KNN(9 vizinhos)	0.854	0.844	0.870	0.857	0.922
MLP	0.826	**0.884**	0.751	0.812	0.911

4.2.2 Resultados do Treinamento Com a Base de Dados Com Primeira e Segunda Doses

Para avaliar o desempenho dos modelos treinados, utilizaremos as métricas: acurácia, precisão, revocação, f1-score e AUC-ROC. Na Tabela 4.10, referente aos resultados dos treinamentos, é possível observar que os modelos de Floresta Aleatória e *XGBoost* obtiveram os melhores resultados em termos da maioria das métricas analisadas.

Tabela 4.10: Métricas de Validação dos Modelos Treinados Com a Base de Dados Com Primeira e Segunda Doses

	Acurácia	Precisão	Revocação	F1-Score	AUC-ROC
Floresta Aleatória	**0.867**	**0.843**	**0.901**	**0.871**	**0.914**
XGBoost	**0.853**	0.829	**0.888**	**0.857**	**0.912**
KNN(8 vizinhos)	0.837	**0.846**	0.826	0.836	0.890
MLP	0.792	0.809	0.774	0.778	0.866

4.2.3 Resultados dos Testes Com os Melhores Resultados dos Modelos Treinados

Na Tabela 4.11 e 4.12 referente aos resultados dos testes, é possível observar que os modelos de Floresta Aleatória e *XGBoost* obtiveram os melhores resultados.

Tabela 4.11: Métricas de Testes dos Modelos Treinados Com a Base de Dados de Pacientes Que Não Tomaram Nenhuma Dose

	Acurácia	Precisão	Revocação	F1-Score	AUC-ROC
Floresta Aleatória	**0.877**	**0.870**	**0.877**	**0.877**	**0.88**
XGBoost	0.875	0.865	0.875	0.875	0.87

Tabela 4.12: Métricas de Testes dos Modelos Treinados Com a Base de Dados Com Primeira e Segunda Doses

	Acurácia	Precisão	Revocação	F1-Score	AUC-ROC
Floresta Aleatória	**0.875**	**0.884**	**0.875**	**0.875**	**0.88**
XGBoost	0.864	0.848	0.864	0.864	0.86

4.3 Transferência de Aprendizado

Conforme observado na seção 4.2.3, os modelos Floresta Aleatória e *XGBoost* demonstraram um desempenho superior em relação às métricas de avaliação em ambas as bases de dados. Por essa razão, serão os modelos e hiper parâmetros escolhidos para a transferência de aprendizado das pessoas que tomaram a terceira dose da vacina, que é uma base de dados desbalanceada. O objetivo será avaliar se essa abordagem resulta em bons resultados de classificação para a base desbalanceada, considerando a capacidade dos modelos em lidar com os casos da classe minoritária e majoritária.

4.3.1 Resultados Das Transferências de Aprendizados dos Modelos Treinados Com a Base de Dados Sem Nenhuma Dose

Floresta Aleatória

Na Tabela 4.13 apresentamos os resultados da transferência de aprendizado do modelo de Floresta Aleatória, observe que a acurácia foi de 77%. Na Figura 4.1 representando a matriz de confusão, vemos que, dos casos em que a classe verdadeira era falsa, o modelo classificou corretamente 78% como falsa e 22% como verdadeira. Para os casos em que a classe verdadeira era verdadeira, o modelo classificou corretamente 77% como verdadeira e 23% como falsa. Além disso, a AUC-ROC deste modelo foi de 78%, conforme demonstrado na Figura 4.2.

Tabela 4.13: Métricas da Transferência de Aprendizado do Modelo Floresta Aleatória

Acurácia	Precisão	Revocação	F1-Score
0.772	0.999	0.772	0.869

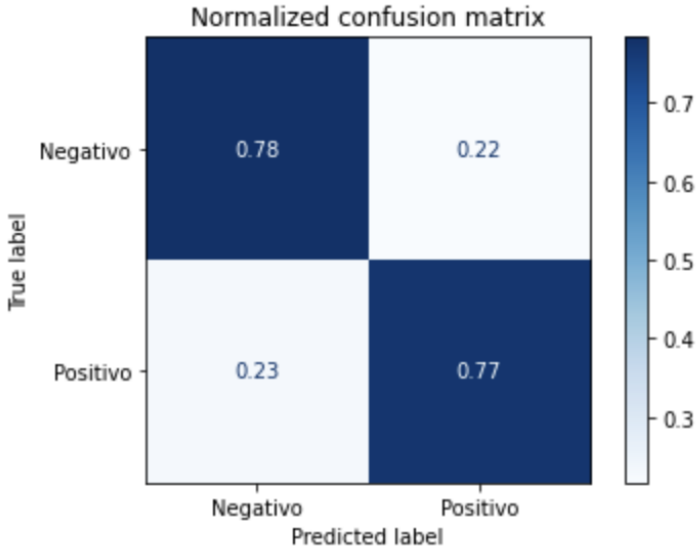

Figura 4.1: Matriz de Confusão Modelo Floresta Aleatória - Transferência de Aprendizado

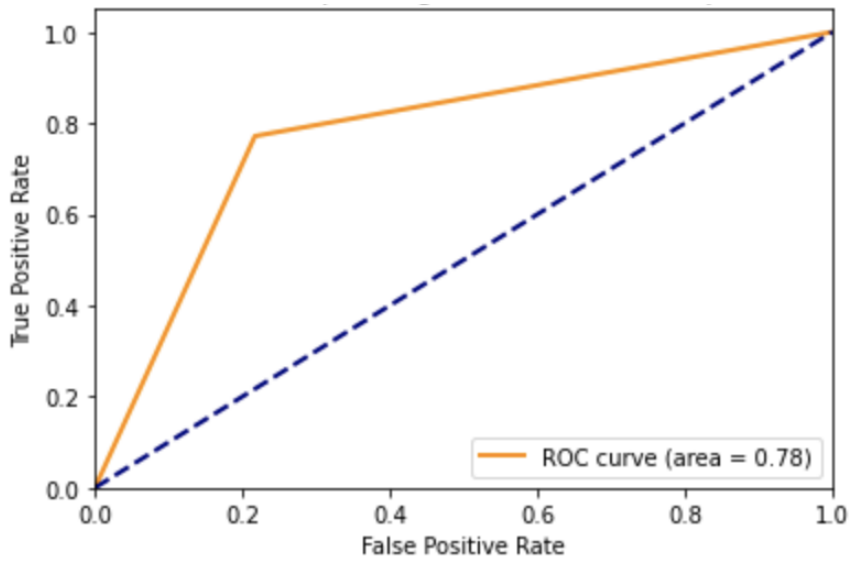

Figura 4.2: Curva ROC Modelo Floresta Aleatória - Transferência de Aprendizado

XGBoost

Na Tabela 4.15 apresentamos os resultados da transferência de aprendizado do modelo de *XGBoost*, observe que a acurácia foi de 79%. Na Figura 4.3 representando a matriz de confusão, vemos que, dos casos em que a classe verdadeira era falsa, o modelo classificou corretamente 73% como falsa e 27% como verdadeira. Para os casos em que a classe verdadeira era verdadeira, o modelo classificou corretamente 79% como verdadeira e 21% como falsa. Além disso, a AUC-ROC deste modelo foi de 76%, conforme demonstrado na Figura 4.4.

Tabela 4.14: Métricas da Transferência de Aprendizado do Modelo XGBoost

Acurácia	Precisão	Revocação	F1-Score
0.792	0.999	0.792	0.883

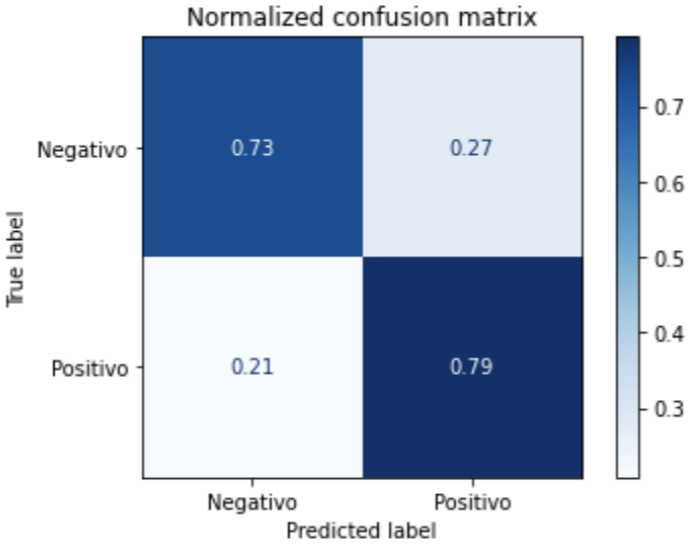

Figura 4.3: Matriz de Confusão Modelo XGBoost - Transferência de Aprendizado

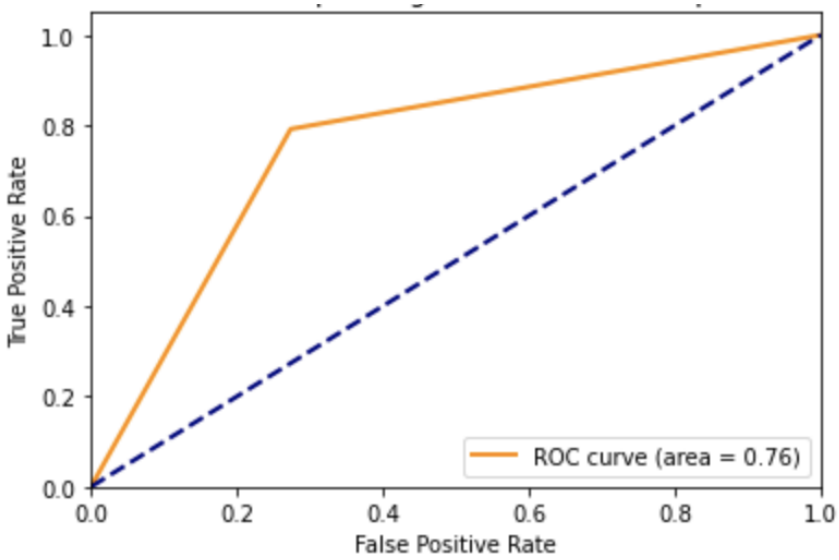

Figura 4.4: Curva ROC Modelo XGBoost - Transferência de Aprendizado

4.3.2 Resultados Das Transferências de Aprendizados dos Modelos Treinados Com a Base de Dados Com Primeira e Segunda Doses

Floresta Aleatória

Na Tabela 4.15 apresentamos os resultados da transferência de aprendizado do modelo de Floresta Aleatória, podemos observar que a acurácia foi de 72%. Na Figura 4.5 representando a matriz de confusão, vemos que, dos casos em que a classe verdadeira era falsa, o modelo classificou corretamente 75% como falsa e 25% como verdadeira. Para os casos em que a classe verdadeira era verdadeira, o modelo classificou corretamente 72% como verdadeira e 28% como falsa. Além disso, a AUC-ROC deste modelo foi de 74%, conforme demonstrado na Figura 4.6.

Tabela 4.15: Métricas da Transferência de Aprendizado do Modelo Floresta Aleatória

Acurácia	Precisão	Revocação	F1-Score
0.725	0.999	0.725	0.840

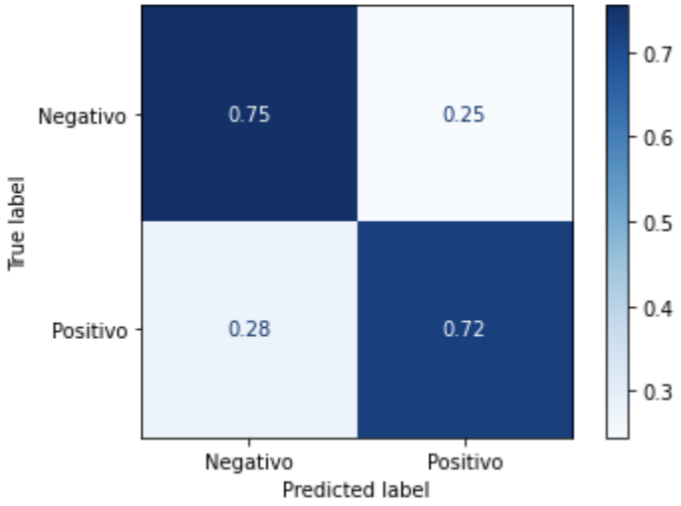

Figura 4.5: Matriz de Confusão Modelo Floresta Aleatória - Transferência de Aprendizado

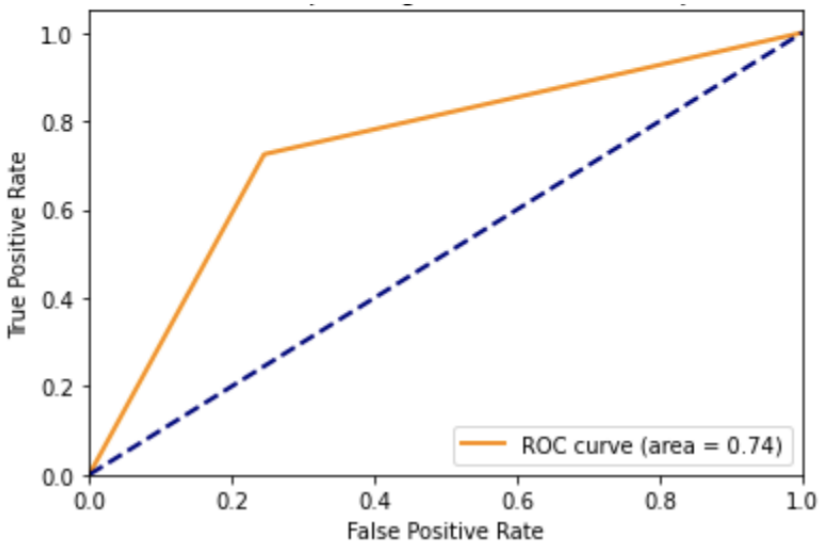

Figura 4.6: Curva ROC Modelo Floresta Aleatória - Transferência de Aprendizado

XGBoost

Na Tabela 4.16 apresentamos os resultados da transferência de aprendizado do modelo de *XGBoost*, podemos observar que a acurácia foi de 69%. Na Figura 4.7 representando a matriz de confusão, vemos que, dos casos em que a classe verdadeira era falsa, o modelo classificou corretamente 77% como falsa e 23% como verdadeira. Para os casos em que a classe verdadeira era verdadeira, o modelo classificou corretamente 69% como verdadeira e 31% como falsa. Além disso, a AUC-ROC deste modelo foi de 73%, conforme demonstrado na Figura 4.8.

Tabela 4.16: Métricas da Transferência de Aprendizado do Modelo XGBoost

Acurácia	Precisão	Revocação	F1-Score
0.691	0.999	0.691	0.817

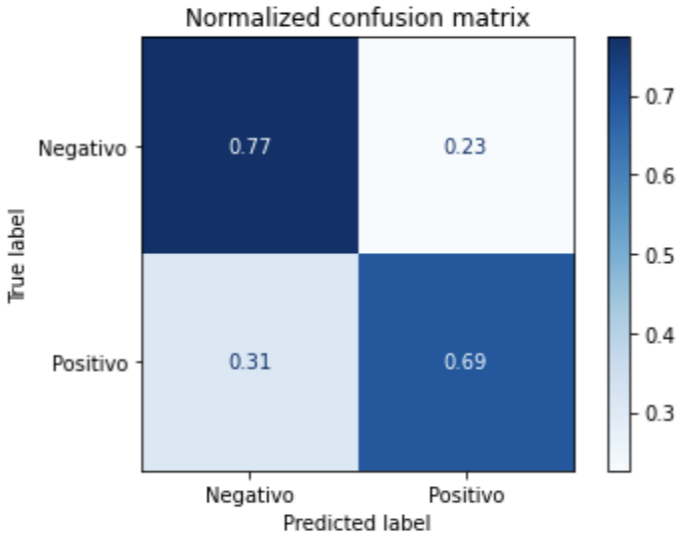

Figura 4.7: Matriz de Confusão Modelo XGBoost - Transferência de Aprendizado

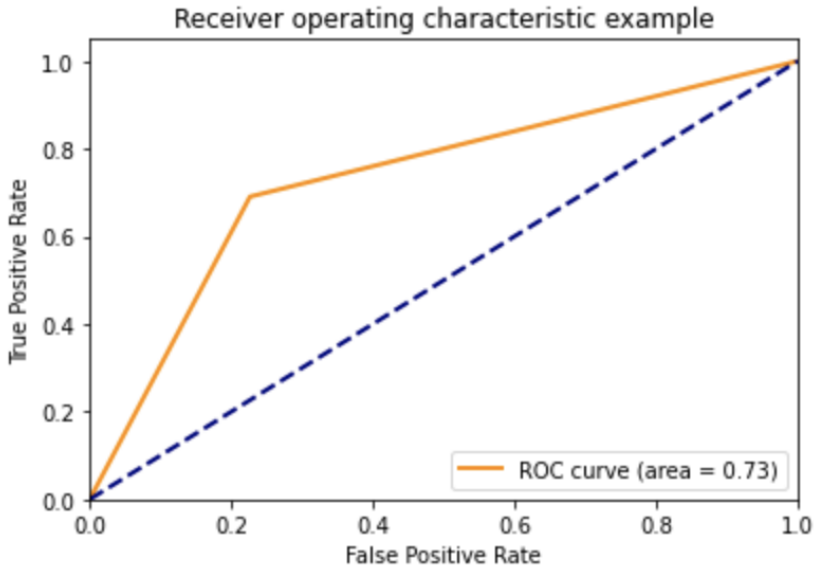

Figura 4.8: Curva ROC Modelo XGBoost - Transferência de Aprendizado

Observe que em praticamente todos os casos, analisando a matriz de confusão, foi possível avaliar mais de 70% dos casos tanto da classe minoritária quanto da classe majoritária. A base de dados com pacientes que não tomaram nenhuma dose da vacina, obteve os melhores resultados em relação as métricas da acurácia e a AUC-ROC.

5 Conclusão e Trabalhos Futuros

O objetivo deste trabalho é classificar os pacientes que receberam a terceira dose da vacina COVID-19, no estado do Rio de Janeiro, através do aprendizado por transferência dos modelos treinados com bases de dados balanceadas de pacientes vacinados com a primeira e segunda doses da vacina e com pacientes não vacinados.

Utilizamos a AUC_ROC como principal métrica para avaliação dos resultados, pois queremos que o modelo consiga distinguir tanto entre as classes positivas quanto as negativas. Com isso, o modelo de melhor desempenho foi o Random Forest, treinado com a base de dados balanceada com pacientes que não tomaram nenhuma dose da vacina. Esse modelo conseguiu classificar corretamente 77% dos casos positivos e 78% dos casos negativos da base de pacientes que tomaram a terceira dose da vacina.

O modelo treinado com mais dados parece ter sido capaz de calibrar melhor os padrões e obteve melhores resultados na transferência de aprendizado. Isso ressalta a influência da quantidade de dados disponíveis no treinamento do modelo. Os resultados obtidos evidenciam a importância de uma base de dados balanceada e a relevância da quantidade de dados disponíveis para o treinamento adequado dos modelos.

Um trabalho futuro interessante seria explorar outros modelos de classificação e experimentar diferentes hiperparâmetros para tentar alcançar uma acurácia melhor nos treinamentos dos modelos, e então, tentar melhorar a transferência de aprendizado.

Bibliografia

ALBUQUERQUE, R. D. R. d. *Estudo Comparativo de Algoritmos de Classificação Supervisionada para Classificação de Polaridade em Análise de Sentimentos*. 2019. UFRPE. Disponível em: ⟨https://repository.ufrpe.br/handle/123456789/2150⟩.

ALPAYDIN, E. *Introduction to Machine Learning*. 2nd. ed. Cambridge, MA: MIT Press, 2010.

BARCELOS, T. N. et al. Análise de fake news veiculadas durante a pandemia de covid-19 no brasil. *Rev Panam Salud Publica*, v. 45, p. e65, 2021. Disponível em: ⟨https://doi.org/10.26633/RPSP.2021.65⟩.

BERNARDO, Y.; ROSARIO, D. do; CONTE-JUNIOR, C. Covid-19 pandemic in rio de janeiro, brazil: A social inequality report. *Medicina (Kaunas)*, v. 57, n. 6, p. 596, Jun 2021.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, 2001.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: ACM. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.], 2016. p. 785–794.

CONTRIBUTOR datahacker. *Machine Learning – Introduction to Random Forest*. 2022. ⟨https://datahacker.rs/012-machine-learning-introduction-to-random-forest/⟩.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, v. 13, n. 1, p. 21–27, 1967.

DAI, Q.-y.; ZHANG, C.-p.; WU, H. Research of decision tree classification algorithm in data mining. *International Journal of Database Theory and Application*, v. 9, n. 5, p. 1–8, 2016. Disponível em: ⟨http://dx.doi.org/10.14257/ijdta.2016.9.5.01⟩.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.

GOV. 2023. ⟨https://infoms.saude.gov.br/extensions/covid-19_html/covid-19_html.html⟩. Acesso em: 18 de julho de 2023.

GUO, Y. et al. The origin, transmission and clinical therapies on coronavirus disease 2019 (covid-19) outbreak - an update on the status. *Mil Med Res*, BioMed Central, v. 7, n. 1, p. 11, Mar 2020.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. [S.l.]: Springer, 2009.

HAYKIN, S. *Neural Networks and Learning Machines Third Edition*. [S.l.]: Pearson, 2008. 122-229 p.

HOSNA, A. et al. Transfer learning: a friendly introduction. *Journal of Big Data*, v. 9, n. 1, p. 1–16, 2022.

JIJO, B.; ABDULAZEEZ, A. M. Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, v. 2, p. 20–28, 01 2021.

KONONENKO, I.; KUKAR, M. *Machine Learning and Data Mining: Introduction to Principles and Algorithms*. [S.l.]: Horwood Publishing, 2007.

KOTSIANTIS, S. B. Supervised machine learning: A review of classification techniques. *Informatica*, v. 31, n. 3, p. 249–268, 2007.

LGPD. 2023. ⟨https://www.gov.br/ans/pt-br/assuntos/noticias/lei-geral-de-protecao-de-dados-lgpd⟩. Acesso em: 1 de julho de 2023.

LIPARAS, D. et al. News articles classification using random forests and weighted multimodal features. In: . [S.l.: s.n.], 2014. v. 8849. ISBN 978-3-319-12978-5.

LORENA, A.; CARVALHO, A. d. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007.

MCKINNEY, W. et al. Data structures for statistical computing in python. In: AUSTIN, TX. *Proceedings of the 9th Python in Science Conference*. [S.l.], 2010. v. 445, p. 51–56.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

POZ, M. D.; LEVCOVITZ, E.; BAHIA, L. Brazil's fight against covid-19. *Am J Public Health*, American Public Health Association, v. 111, n. 3, p. 390–391, Mar 2021.

SCIKIT-LEARN. *Cross-validation*. Acesso em 1 de julho de 2023. ⟨https://scikit-learn.org/stable/modules/cross_validation.html⟩.

SCIKIT-LEARN. *Grid Search*. Acesso em 9 de julho de 2023. ⟨https://scikit-learn.org/stable/modules/grid_search.html⟩.

TAN, H. Machine learning algorithm for classification. *Journal of Physics: Conference Series*, v. 1994, n. 1, p. 012016, 2021. Disponível em: ⟨https://iopscience.iop.org/article/10.1088/1742-6596/1994/1/012016/pdf⟩.

UDDIN, S. et al. Comparative performance analysis of k-nearest neighbour (knn) algorithm and its different variants for disease prediction. *Scientific Reports*, v. 12, n. 6256, 2022. Disponível em: ⟨https://doi.org/10.1038/s41598-022-10358-x⟩.

WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. A survey of transfer learning. *Journal of Big Data*, Springer, v. 3, n. 1, p. 9, 2016. Disponível em: ⟨https://doi.org/10.1186/s40537-016-0043-6⟩.

YOSINSKI, J. et al. *How transferable are features in deep neural networks?* 2014.