CENTRO UNIVERSITÁRIO SENAC

SANTO AMARO

PÓS-GRADUAÇÃO EM BIG DATA

Projeto Serviço de Inferência – Documentação:

Heart Attack Analysis

Alison da Silva Rodrigues

Merhi Mohamed Taha

Raelma Dionisio da Silva

Yan Oliveira Reis

Ygor Souza de Oliveira

Resumo

Este artigo apresenta um serviço de inferência web na área da saúde para classificar pacientes com chances de ataques cardíacos. A implementação inclui a utilização de serviços da nuvem AWS, técnicas sofisticadas para confecção de modelos preditivos e conta com uma base de dados disponibilizada pela Universidade da Califórnia e adaptada na plataforma Kaggle para melhor aderência ao processo Machine Learning.

Palavras-chave: Inferência, Saúde, Machine Learning, AWS.

São Paulo

2023

Sumário

1. Introdução

As informações introdutórias deste artigo encontram-se descritas nos subtópicos abaixo.

1.1 Contexto e motivação

Nas últimas décadas, tem sido observado um crescimento gradual nos casos de Infarto Agudo do Miocárdio (IAM). O aumento da expectativa de vida, embora seja um indicador positivo do progresso médico e social, traz consigo desafios significativos no campo da saúde cardiovascular. O aumento nas admissões de emergência devido à suspeita de IAM destaca a necessidade de métodos para prevenir e detectar precocemente esse evento crítico. No Brasil, o IAM representa a principal causa de mortes, com estimativas indicando a ocorrência de 300 mil a 400 mil casos anuais. Alarmantemente, a cada 5 a 7 casos, um resulta em óbito (Ministério da Saúde do Brasil, 2023).

O aumento constante nos atendimentos de emergência para casos de IAM destaca a urgência de encontrar maneiras eficientes de identificar fatores de risco e prever ataques cardíacos. Neste estudo, exploramos como a computação em nuvem pode ser uma solução inovadora para criar uma ferramenta acessível e eficaz. Atualmente, muitas pesquisas já são realizadas nesse contexto. O *Machine Learning* utilizado em clínicas têm o potencial de transformar os cuidados, tornando a prática clínica mais eficiente e rápida. A Inteligência Artificial surge como uma alternativa uma vez que tem o potencial de explorar grandes quantidades de informação de forma automática e sistemática, conforme apontado pelo estudo de Oliveira et al. (2023), que demonstrou que o uso de aprendizado de máquina pode prever a mortalidade em casos de Infarto Agudo do Miocárdio.

1.2 Objetivo do projeto

O objetivo do projeto é classificar o risco de ataque cardíaco com base em dados coletados de pacientes, tais como dados pessoais, índices de saúde cardiovascular e resultados de exames. Ao integrar tecnologias avançadas, busca-se não apenas armazenar e processar dados, mas também oferecer

uma solução prática para prevenir e detectar precocemente ataques cardíacos. Essa abordagem visa reduzir o impacto devastador dessa condição na sociedade brasileira e em outras comunidades.

2. Solução proposta

A solução adota os princípios da arquitetura REST, incorporando técnicas de *Machine Learning* supervisionadas para análise e classificação de pacientes, com ênfase na regressão logística. Utiliza o Framework Flask para facilitar o gerenciamento de requisições via URLs e aproveita os serviços da nuvem AWS. Entre esses serviços, destacam-se o Elastic Cloud Computing (EC2) para virtualização de máquinas, o Simple Service Storage (S3) para armazenamento do modelo pré-treinado e o Identity and Access Management (IAM) para a criação de usuários e credenciais de acesso ao modelo armazenado.

Cada serviço desempenha uma função específica, contribuindo para a eficiência global do projeto. A escolha por uma abordagem baseada em serviços em nuvem foi motivada não apenas pela oferta de funcionalidades essenciais ao projeto, mas também pela capacidade de transformar o desenvolvimento realizado em um produto utilizável e um serviço consumível. Na realidade, com a atual crescente dos dados e necessidade de serviço, é comum que soluções de *Machine Learning* utilizem serviços de nuvem. Já existem frentes de estudo que abordam temas como *Machine Learning as a Service (MLaaS)* dado a necessidade de disponibilização e reutilização de modelos em qualquer sistema, pois eles são um fator impulsionador na disseminação do conhecimento científico e na promoção da colaboração entre áreas de investigação. Ou seja, é importante que os modelos estejam acessíveis e sejam funcionais como um serviço, como é o caso de Álvaro López García et al. (2018) que propôs um framework baseado em nuvem para cargas de trabalho e aplicações de aprendizado de máquina.

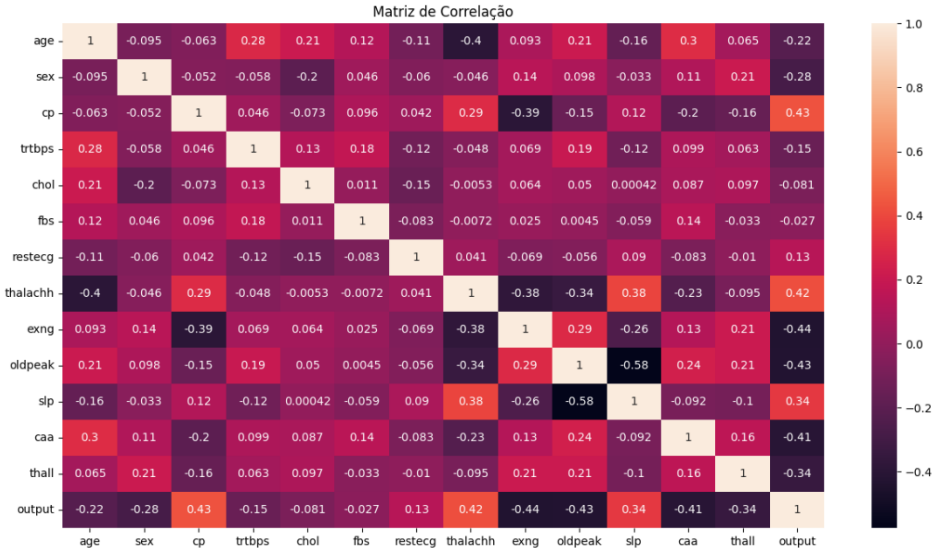
Antes de aprofundar adiante cada etapa da implementação e os serviços utilizados, é importante um contexto breve sobre o processo de criação do modelo de *Machine Learning*.

A base de estudo provêm da plataforma Kaggle e esta contém dados pessoais de pacientes, como idade e gênero, e informações de saúde cardiovascular, como pressão arterial, nível de colesterol, açúcar no sangue, tipo de dor no peito (angina), entre outros campos relevantes para o modelo (disponível em: https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset/data).

Foi utilizado uma abordagem de regressão logística para classificar os pacientes em dois grupos distintos: baixas chances de ataques cardíacos versus maiores chances de ataques cardíacos (0 ou 1). Para o desenvolvimento, foram utilizadas ferramentas como Jupyter Notebook e a linguagem Python, fazendo uso das bibliotecas Numpy, Pandas, Scikit Learning, JobLib e Boto3. O código completo da confecção do modelo implementado está disponível em neste notebook: https://colab.research.google.com/drive/1rDiiflbih_HPRhPChCzc_RZ-ciE5xc1t?usp=sharing

O procedimento de construção do modelo incorporou os 13 atributos da base de dados, com um particionamento de 80% para treinamento e 20% para teste. Na análise de correlação entre os atributos, destacaram-se dados como dores no peito (cp), frequência cardíaca máxima atingida (thalachh), angina induzida por exercícios (exng) e outras colunas específicas, conforme evidenciado na matriz de correlação.

Figura 1 – Matriz de Correlação



Fonte: Elaborado pelos autores

Sobre os resultados gerais obtidos, o modelo obteve como melhor desempenho uma acurácia de 82%, porém ela não deve ser considerada alta no âmbito da saúde visto que é uma inferência delicada e de valor muito alto que são as vidas de pacientes (precisaria ser mais assertivo). No entanto, para fins acadêmicos, não conseguimos melhorar a performance com a atual base. Seriam necessários mais atributos, volume de dados, dentre outras possíveis técnicas de ML.

Figura 2 – Resultados do modelo confeccionado

	precision	recall	f1-score	support
0	0.80	0.83	0.81	29
1	0.84	0.81	0.83	32
accuracy			0.82	61
macro avg	0.82	0.82	0.82	61
weighted avg	0.82	0.82	0.82	61

Fonte: Elaborado pelos autores

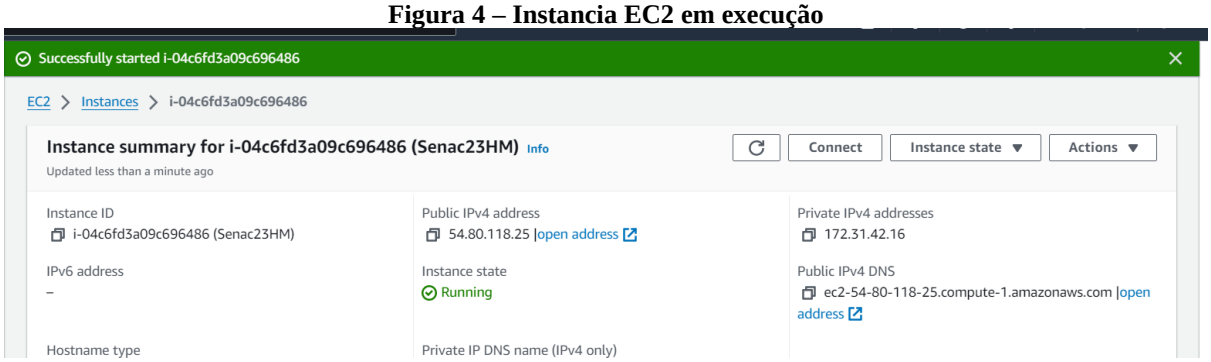
Com o modelo desenvolvido, foi dado início a utilização dos serviços de nuvem da AWS seguido da implementação do serviço de inferência.

Para a solução foi necessário utilizar 3 serviços pontuais da nuvem AWS e todos eles em nível gratuito (*free tier*): Elastic Cloud Computing (EC2) para virtualização de máquinas, Simple Service Storage (S3) para armazenamento do modelo pre-treinado e, por fim, o Identity and Acess Management (IAM) para criação de usuário e credenciais de acesso ao modelo armazenado.

Sobre o Elastic Cloud Computing (EC2): Este serviço permite virtualizar máquinas dando personalização ao número de processadores, armazenamentos e sistemas operacionais. Segundo Castaño-Diéz (2017), mediante o uso de uma conta na AWS, a utilização apenas dos recursos computacionais necessários para realizar a demanda específica permite o pagamento somente pelos recursos utilizados, o que no trabalho em questão se limitou as funcionalidades gratuitas. Por isso, o EC2 é focado para soluções que demandam tarefas escaláveis. Para efeito de projeto, utilizou-se de uma máquina virtual Ubuntu para preparação do ambiente necessário e que fosse capaz de executar a aplicação. A grande vantagem é não precisar se preocupar com a gestão de máquinas físicas alocadas em datacenters, pois com a abordagem do EC2, essa gestão passa a ser do fornecedor de nuvem, ou seja, da própria AWS. A conexão foi realizada via protocolo de segurança SSH a partir de uma

chave privada e, com a máquina conectada, iniciou-se a preparação do ambiente instalando o conda (sistema gerenciador de pacotes e ambientes) para configurar as bibliotecas necessárias. Também foram instalados o Jupyter Notebook, para desenvolver o modelo, e o Framework Flask, para gerenciamento das URLs. Em termos de máquinas virtuais para execução da aplicação, essas foram as configurações necessárias. Ela seria a responsável por executar o serviço de inferência fornecendo um ambiente preparado, escalável e pago apenas pelo consumo utilizado.

Figura 4 – Instancia EC2 em execução



Fonte: Elaborado pelos autores

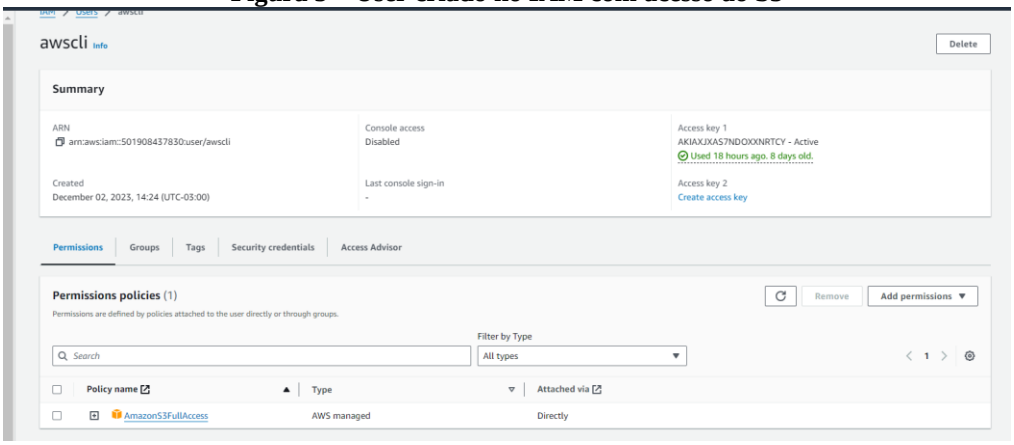
Sobre o Identity and Acess Management (IAM): Este serviço permite criar usuários que acessem os serviços da AWS, seja via linha de comando ou interface. O que foi importante entender durante o desenvolvimento do projeto é que para realizar tarefas que geralmente tornam-se repetitivas, é contraprodutivo fazer manualmente via interface. Para isso, existe a possibilidade de comunicação com os serviços da AWS a partir de um usuário e credenciais de acesso criadas. Conforme abordagem adotada por Shevrin (2013) em um artigo sobre ataques IAM via *Model Checking*, quando configuramos as credenciais, precisamos nos atentar detalhadamente sobre as permissões e credenciais que estão sendo configuradas, sendo cada permissão composta por 5 elementos básicos:

1) Efeito: Permitido ou Negado;
2) Ação: Define quais ações podem ou não podem ser realizadas;
3) Recurso: Define em quais recursos uma determinada ação pode ou não ser realizada;
4) Entidade Principal: Define por quais entidades principais as ações podem ou não serem efetivadas;

5) Condição: Define condições adicionais dependendo da ação.

Para o projeto foi realizada a criação de um usuário (Command Line Interface), e a partir de suas credenciais (Acess Key e Secrect Acess Key) foi possível permitir o acesso total no serviço S3. Vale comentar que geralmente não é liberado *Full Acess* nas políticas do IAM, pois é necessário compreender e liberar apenas o mínimo necessário para o usuário.

Figura 5 – User criado no IAM com acesso ao S3



Fonte: Elaborado pelos autores

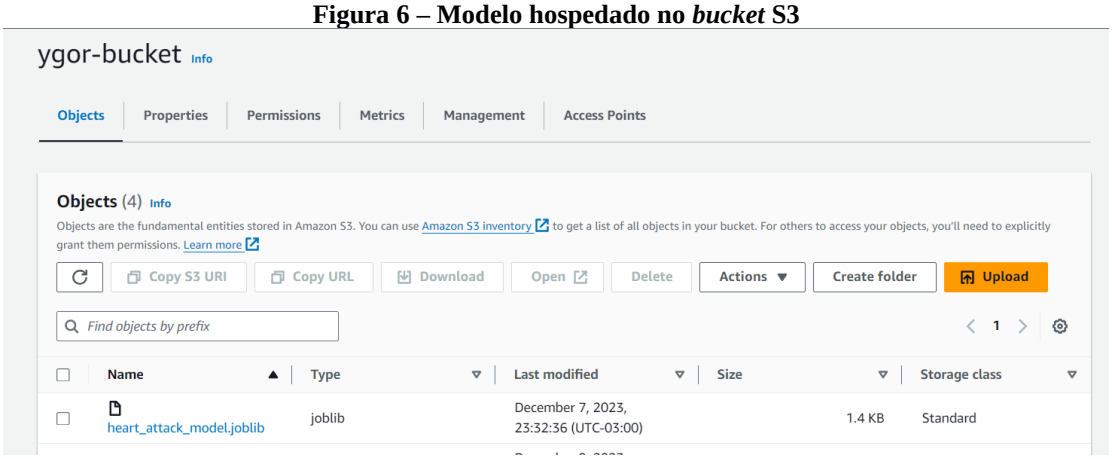
Sobre o Simple Service Storage (S3): Ele é um serviço de armazenamento projetado para ser altamente durável e disponível, sendo uma solução popular para armazenamento de objetos, como imagens, vídeos, arquivos de backup, e outros tipos de dados. Ele é uma plataforma na nuvem que armazena diversos objetos de dados organizados em "buckets". Os usuários têm a flexibilidade de criar esses *buckets* em qualquer uma das regiões disponíveis na AWS, atribuindo para eles nomes globalmente únicos, independentemente da região escolhida. A ferramenta também possui controle de versão e políticas de acesso.

O ponto mais interessante deste serviço é a facilidade que proporciona, fazendo com que uma requisição acesse rapidamente algum objeto armazenado. Diferente de um banco de dados que possui esquemas, dentre diversas configurações para obtenção de um dado, o S3 desempenha um papel mais ágil. Um ponto de atenção importante é sobre a segurança do *bucket e seus objetos armazenados.* Políticas de segurança podem (e devem) ser anexadas aos objetos e bucket como um todo manualmente. Um artigo bem legal escrito por Continella et al. (2018), comenta a respeito da automação dessa configuração de políticas e propõe uma extensão de

navegador que evita o carregamento de recursos hospedados em *buckets* não seguros, destinados tanto a usuários finais, como uma mitigação contra sites vulneráveis e para desenvolvedores e testadores de software, como forma de verificar se há configurações incorretas.

Para o projeto, foi utilizado um *bucket* não público para armazenar o modelo já treinado. A ideia principal é que, com a máquina virtual em execução, o usuário criado no serviço IAM consiga pegar o modelo disponível no *bucket* S3 de forma rápida e eficaz, esse gerando uma resposta de inferência ao cliente.

Figura 6 – Modelo hospedado no *bucket* S3



Fonte: Elaborado pelos autores

Com o modelo treinado e armazenado no bucket S3, e a máquina virtual em execução, foi necessário desenvolver um código Python capaz de interpretar as requisições do cliente, buscar o modelo armazenado no S3 e gerar uma nova inferência com base nos dados fornecidos pelo cliente. Em outras palavras, o código lida com solicitações do cliente e fornece respostas do servidor.

Para que esse código funcionasse, seguimos uma arquitetura amplamente utilizada na construção de sistemas distribuídos na web, chamada REST (Representational State Transfer). Em resumo, a arquitetura REST utiliza recursos identificados por URIs e manipulados por meio de representações, como JSON ou XML. Guinard et al. (2012) explicam que na REST, as operações usam URLs para identificar recursos, e os métodos HTTP padrão (POST, GET, PUT etc.) servem como a interface de serviço permitindo comunicação entre cliente e servidor.

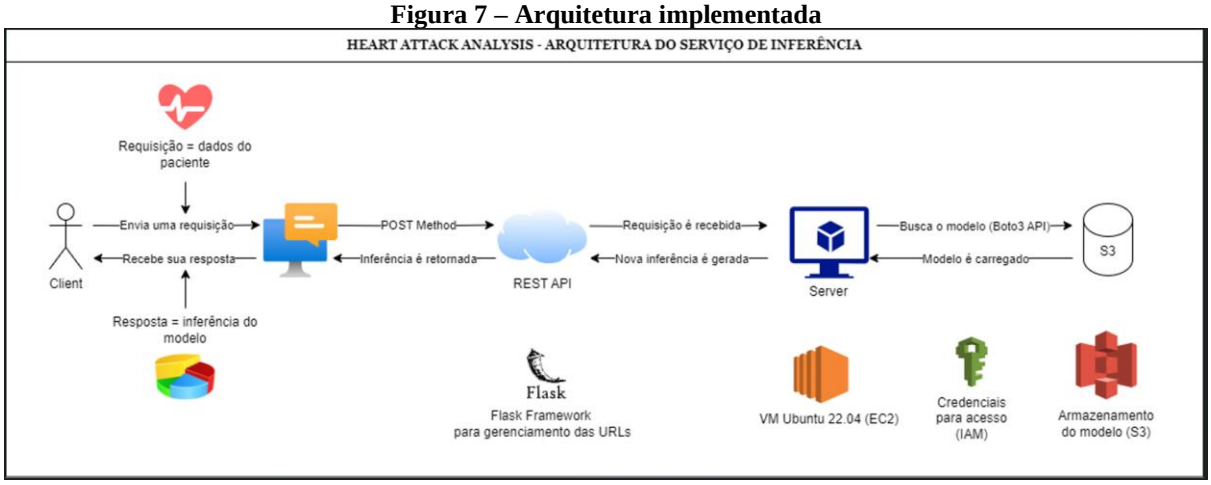
Uma das características fundamentais da REST é que a comunicação é *stateless*, ou seja, cada requisição de um cliente para um servidor contém todas as

informações necessárias para entender e processar a solicitação. O servidor não mantém nenhum estado sobre o cliente entre requisições. Cada requisição é independente e autocontida. Esses princípios tornam a arquitetura REST eficiente, escalável e amplamente utilizada em serviços web e APIs.

Para implementar os princípios da arquitetura REST no projeto, optamos por utilizar o framework Flask, o qual simplifica consideravelmente a organização das rotas de acesso às URLs, eliminando a necessidade de configurar vários arquivos, como XML. No próprio código Python, é possível definir qual URL será acessada, por qual método (POST ou GET), e especificar o que a rota realizará e qual resposta será gerada. Essencialmente, esse fluxo é encapsulado em funções Python correspondentes a cada rota configurada.

Dessa forma, para o serviço de inferência, foram desenvolvidas duas formas de requisição, ambas utilizando o método POST, ou seja, capturando dados do corpo da página em vez de parâmetros de URL. A primeira abordagem envolve o uso de um formulário HTML (interface), enquanto a segunda utiliza JSON (código). Mais abaixo no texto estão apresentadas algumas figuras que evidenciam o funcionamento do serviço de inferência. Logo abaixo está evidente a arquitetura final do projeto.

Figura 7 – Arquitetura implementada



Fonte: Elaborado pelos autores

Seguindo o fluxograma apresentado na figura, o processo inicia-se com o cliente enviando uma requisição contendo os dados do paciente. Essa requisição é transmitida ao servidor por meio do método POST, sendo os dados incorporados no corpo da página. O servidor, ao receber essa requisição, realiza a leitura dos dados

fornecidos, carrega o modelo armazenado no S3 e executa uma nova inferência com base nessas informações.

Ao finalizar o processo de inferência, o servidor retorna a resposta ao cliente. Essa resposta inclui a classificação do paciente, indicando se há baixas ou maiores chances de um ataque cardíaco, além das probabilidades associadas a cada classe. Essa abordagem fornece ao cliente informações valiosas para a tomada de decisões clínicas, permitindo uma análise mais precisa e rápida dos riscos cardíacos do paciente. O uso combinado do método POST e da arquitetura REST proporciona uma comunicação eficiente entre cliente e servidor, garantindo a integridade e segurança dos dados transmitidos durante o processo.

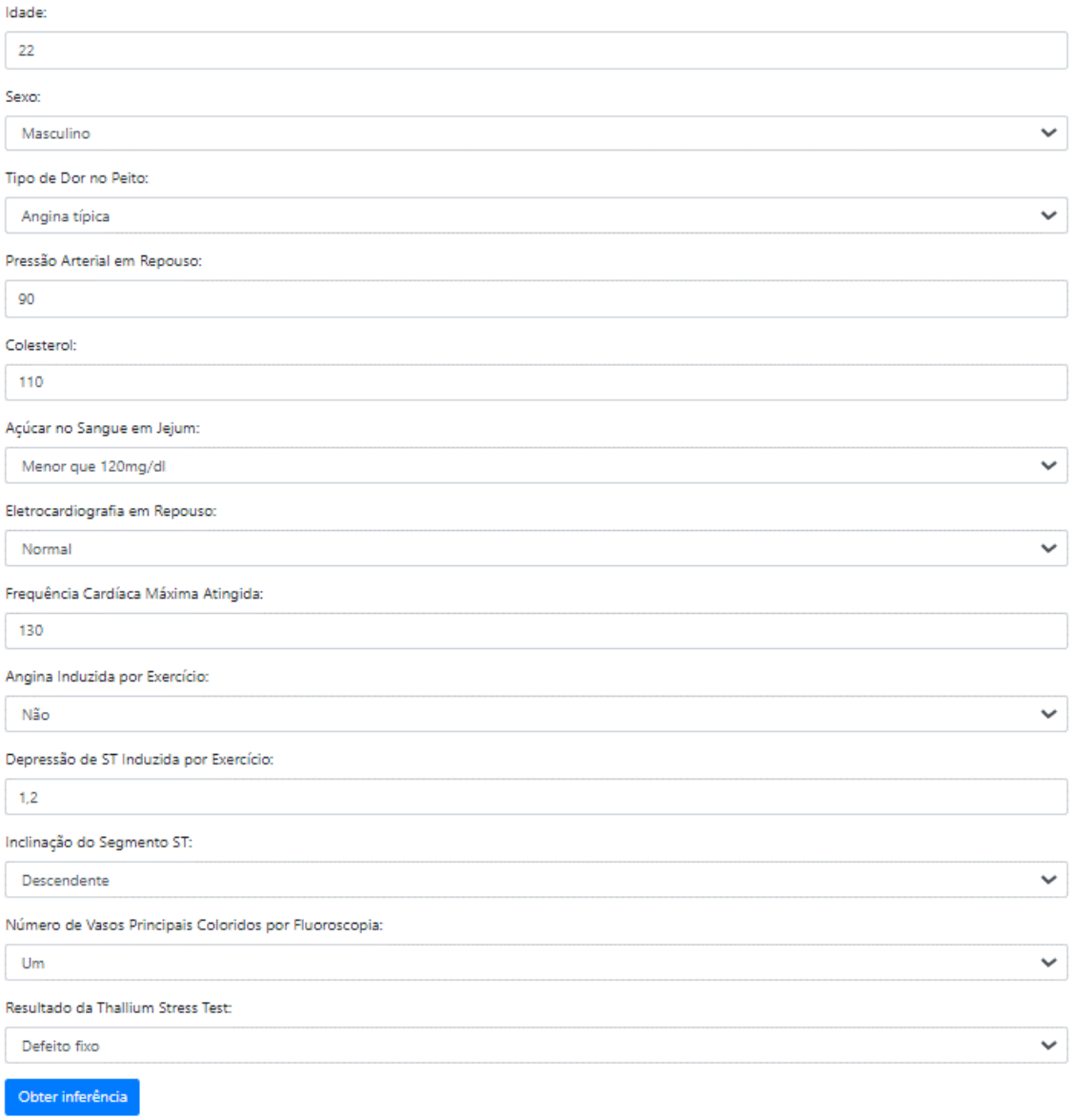
Figura 8 – Código utilizando o Framework Flask

```
from flask import request
@app.route('/model', methods=['POST']) #Block GET requests
def model():
  if request.method == 'POST':

    # Buscando o modelo disponível no s3
    s3_client = boto3.client(service_name='s3')
    s3_client.download_file("ygor-bucket", "heart_attack_model.joblib", "./heart_attack_model.joblib")

    # Com download feito carregamos o modelo para o código
    modelo = load('./heart_attack_model.joblib')

    # Coletando dados do formulário
    dados_cliente = pd.DataFrame({
    'age': [request.form['age']],
    'sex': [request.form['sex']],
    'cp': [request.form['cp']],
    'trtbps': [request.form['trtbps']],
    'chol': [request.form['chol']],
    'fbs': [request.form['fbs']],
    'restecg': [request.form['restecg']],
    'thalachh': [request.form['thalachh']],
    'exng': [request.form['exng']],
    'oldpeak': [request.form['oldpeak']],
    'slp': [request.form['slp']],
    'caa': [request.form['caa']],
    'thall': [request.form['thall']]
    })

    # Gera a inferência
    nova_predicao = modelo.predict(dados_cliente)
    probabilidades_novo_dado = modelo.predict_proba(dados_cliente) * 100

    # Montando apresentação gráfica
    labels = ['Classe 0', 'Classe 1']
    sizes = [probabilidades_novo_dado[0, 0], probabilidades_novo_dado[0, 1]]

    fig, ax = plt.subplots()
    ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
    ax.axis('equal')

    ax.text(0.5, 1.1, f"Resultado da inferência: {nova_predicao}", ha='center', va='center', transform=ax.transAxes)

    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode()

    resposta = f"<img src='{plot_url}' alt='Probabilidades do Modelo'>"

    plt.close()

    return resposta
```

Fonte: Elaborado pelos autores

Na próxima figura está apresentada a interface visual para obter inferências a serem realizadas pelo modelo. O código pega os dados do corpo da página usando o método POST (pega os dados preenchidos no formulário).

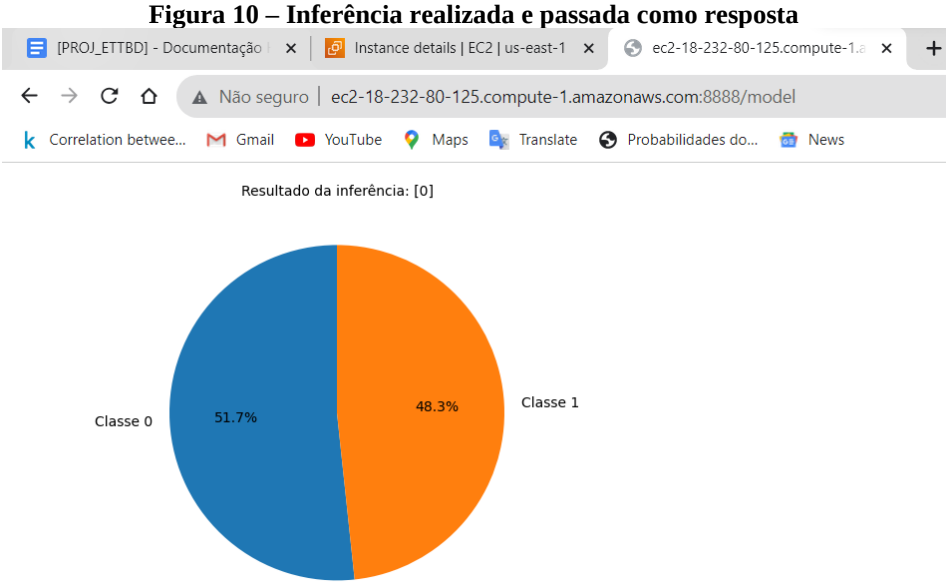
Figura 9 – Formulário HTML para realizar requisição



Fonte: Elaborado pelos autores

Após preenchimento do formulário, ao clicar em obter inferência o código faz *download* do modelo disponível no serviço S3 e gera a nova predição com os dados do paciente. Como resposta é passado a classificação (0 ou 1) como também as probabilidades de cada classe.

Figura 10 – Inferência realizada e passada como resposta



Fonte: Elaborado pelos autores

Esse processo pode ser feito através dessa interface HTML ou via código JSON que realizará o mesmo processo.

Figura 11 – Serviço de inferência com requisição em JSON

Fonte: Elaborado pelos autores

Conclusões

A escolha de usar computação em nuvem foi fundamental para o projeto. Isso se encaixa bem na ideia de MÜLLER & GUIDO (2023) sobre a integração bem-sucedida de plataformas de nuvem para soluções de *Machine Learning*. Essa abordagem se conecta ao modelo construído, que mostrou ser bem preciso em prever ataques cardíacos com base nas variáveis que analisamos.

Ao colocar esse modelo no *bucket* S3 da AWS e disponibilizar este serviço de inferência na web, faz com que mais pessoas possam o utilizar, incluindo profissionais de saúde e pesquisadores. A mistura entre a análise de dados, a criação do modelo e o uso da nuvem oferece uma solução completa para prever ataques cardíacos, o que pode trazer um impacto positivo para a área da saúde.

Ao combinar a eficácia do modelo com a flexibilidade da nuvem, nosso projeto se destaca como uma iniciativa interessante para melhorar as estratégias de prevenção e tratamento de problemas cardíacos. A união entre avanços tecnológicos e cuidados médicos reflete um passo importante em direção a soluções inovadoras na área da saúde cardíaca.

A disponibilidade fácil do modelo na AWS não apenas facilita o acesso, mas também incentiva a colaboração entre diferentes áreas, permitindo que especialistas em saúde e pesquisadores usem a tecnologia para melhorar suas abordagens. Essa integração bem-sucedida entre tecnologia e medicina não apenas oferece uma ferramenta de inferência valiosa, mas também estabelece bases para futuros avanços na prevenção e tratamento de doenças cardíacas. Em resumo, a conexão entre análise de dados, criação de modelos e uso da nuvem não apenas atende aos objetivos do projeto, mas também abre portas para melhorias significativas na saúde cardiovascular.

4. Referências

MINISTÉRIO DA SAÚDE DO BRASIL. Infarto Agudo do Miocárdio. Disponível em: <https://www.saude.gov.br/>. Acesso em: 9 dez. 2023.

OLIVEIRA, M. et al. Machine learning prediction of mortality in Acute Myocardial Infarction. PubMed Central, 18 abr. 2023. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10111317/. Acesso em: 9 dez. 2023.

ALVÁRO, G. et al. A Cloud-Based Framework for Machine Learning Workloads and Applications. IEEE Xplore, 06 jan. 2020. Disponível em: https://ieeexplore.ieee.org/document/8950411. Acesso em: 9 dez. 2023.

CASTAÑO-DIÉZ, D. **The Dynamo package for tomography and subtomogram averaging:** components for MATLAB, GPU computing and EC2 Amazon Web Services. 73. ed. Basel: Acta Cryst., 28 fev. 2017. Disponível em: https://www.scienceopen.com/document?vid=afa707d2-8a7b-425d-a243-c26f41077797. Acesso em: 9 dez. 2023.

SHEVRIN, I. **Detecting Multi-Step IAM Attacks in AWS Environments via Model Checking**. 32. ed. Anaheim: USENIX Association, 9 ago. 2023. Disponível em: https://www.usenix.org/system/files/usenixsecurity23-shevrin.pdf. Acesso em: 9 dez. 2023.

CONTINELLA, A. et al. **There's a Hole in that Bucket! A Large-scale Analysis of Misconfigured S3 Buckets**. San Juan: ACSAC, 3 dez. 2018. Disponível em: https://re.public.polimi.it/bitstream/11311/1065367/1/2018-continella-bucketsec.pdf. Acesso em: 9 dez. 2023.

GUINARD, D. et al. **In Search of an Internet of Things Service Architecture**: REST or WS-*? A Developers' Perspective. 104. ed. Zurich: LNICST, 2012. Disponível em: https://eudl.eu/pdf/10.1007/978-3-642-30973-1_32. Acesso em: 9 dez. 2023.

MÜLLER, Andreas C.; GUIDO, Sarah. **Introduction to Machine Learning with Python**: A Guide for Data Scientists. 1ª ed. Nova York: O'Reilly Media, 2023. 400 p. ISBN 978-1492041412. Disponível em:https://learning.oreilly.com/library/view/introduction-to-machine/9781449369880/?sso_link=yes&sso_link_from=Oracle. Acesso em: 9 dez. 2023.