

Relatório do Grau B - Publicação de Aplicação para Predição de Diagnóstico de Câncer

Bel Cogo¹, Bruno da Siqueira Hoffmann¹

¹Escola Politécnica – Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo – RS – Brazil

{belcogo,brunohoffmann2018}@unisinos.br

Resumo. *Este relatório descreve o desenvolvimento de uma aplicação capaz de prever o diagnóstico de câncer de um paciente com base em algumas de suas características e rotina. A partir disso, foi necessário realizar um treinamento de um modelo de aprendizagem de máquina, utilizando o algoritmo RandomForest, como também a criação de uma aplicação para internet capaz de capturar os dados do usuário. Além da aplicação desenvolvida, também foi realizada a publicação da mesma em um ambiente em nuvem AWS, usando o recurso EC2.*

1. Introdução

Para realização do presente trabalho foi utilizado os recursos do trabalho anterior, que seria um modelo capaz de prever se um paciente possui ou não diagnóstico de câncer com base em suas características e rotina. A partir disso, foi desenvolvida uma aplicação em *python*, usando a ferramenta *streamlit*, capaz de prever o diagnóstico de câncer de um paciente com base nas suas características. Além disso, após o desenvolvimento da aplicação o presente trabalho também teve como objetivo fazer a publicação desta aplicação em um ambiente na nuvem AWS.

2. Treinamento do Modelo

Para o treinamento do modelo foi utilizado um *dataset* contendo informações de 1500 pacientes que possuíam ou não diagnósticos de câncer, e tinham as seguintes informações: idade, gênero, IMC (Índice de massa corporal), tempo de atividade física, fumante, histórico de câncer, risco genético e consumo de unidades de álcool semanalmente. Além disso, o treinamento foi realizado a partir de um algoritmo *python* utilizando ferramentas como: *pandas* e *sklearn*.

A partir do *dataset* mencionado, foi necessário realizar a importação do arquivo para manipulação dos dados, onde foi utilizado de recursos da própria ferramenta *pandas* para importação do arquivo *csv* contendo os dados. Em seguida, foi necessário fazer o pré-processamento dos dados, realizando a execução do *One Hot Encoder* em cima dos dados da coluna risco genético, à medida que a mesma continha dados de natureza categórica. Adicionalmente, a execução do escalonador *StandardScaler* foi realizada, com o intuito de normalizar os dados de todas as colunas.

Com os dados pré-processados foi realizada a subdivisão dos conjuntos, onde foi utilizado a estratégia de separação dos dados de maneira randomizada, onde 30% dos dados foram utilizados para teste e os demais para treinamento. E em seguida, foi

realizado o treinamento do modelo, usando o algoritmo *RandomForest*, contendo os hiperparâmetros *n_estimators* como 10 e *criterion* como *entropy*.

Após o treinamento, o algoritmo desenvolvido fazia o armazenamento do modelo, do *encoder* e do *scaler* para arquivos da extensão *pkl*, utilizando a ferramenta *pickle* para que assim fosse possível utilizar o modelo e os *encoders* na aplicação posteriormente.

Por fim, ao executar o algoritmo era possível verificar na saída no console as métricas do treinamento, sendo elas a acurácia, a precisão, o *recall* e o *f1-score*, que continham respectivamente os valores, 89.3%, 92.6%, 78.9%, 85.2%. É possível verificar a saída do console na imagem abaixo.

```
brunohoffmann@MacBook-Pro-de-Bruno trab-1-big-data-gb % python3 ./src/train_random_forest.py
Acurácia: 89.33333333333333%
Precisão: 92.61744966442953%
F1-Score: 85.18518518518519%
Recall: 78.85714285714286%
```

Figura 1. Saída do Console com Informações sobre Métricas do Modelo.

3. Aplicação

Após o treinamento do modelo, foi criado uma aplicação, sendo ela capaz de fazer a predição se o paciente possui ou não diagnóstico de câncer com base nas suas características. A partir disso, o desenvolvimento da aplicação foi feito na linguagem *python*, utilizando a ferramenta *streamlit*. Essa aplicação pode ser encontrada no link: <https://github.com/BrunoHoffmann15/trab-1-big-data-gb>.

A aplicação contém uma tela de *login* onde o usuário consegue se autenticar, onde foi configurado uma senha fixa, sendo ela “bel_bruno@123”. Além disso, há outra tela contendo o formulário necessário para fazer a predição, conjunto com uma opção para visualizar os dados a partir de gráficos. Na figura abaixo é possível verificar a tela de *login*.

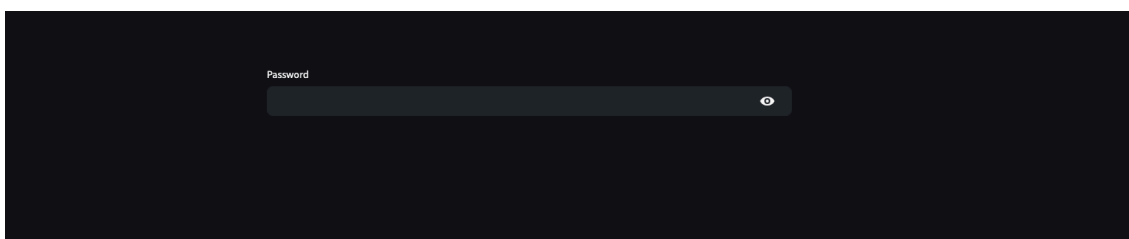


Figura 2. Tela de Login.

3.1. Visualização dos Dados

Para a visualização dos dados foram utilizadas boa parte das bibliotecas utilizadas na execução do trabalho da primeira metade do semestre, no *Google Colab*, sendo elas, *matplotlib*, *seaborn* e *plotly*, responsáveis pelo processamento dos dados e produção dos gráficos, em conjunto com a biblioteca *streamlit*, responsável pela renderização dos gráficos gerados pelas bibliotecas anteriores.

A visualização dos dados com gráficos foi utilizada para apresentar de forma simplificada e mais ilustrativa a divisão dos dados de acordo com algumas das *features* consideradas mais relevantes para o resultado do diagnóstico, assim como um comparativo dessa mesma divisão com a distribuição por diagnóstico de cada *feature*. Como por exemplo, nos gráficos, Gráfico 1 e Gráfico 2, abaixo demonstrando primeiramente a divisão referente ao gênero dos pacientes e em seguida a mesma divisão, porém identificando quantos pacientes de cada gênero foram ou não diagnosticados com câncer.

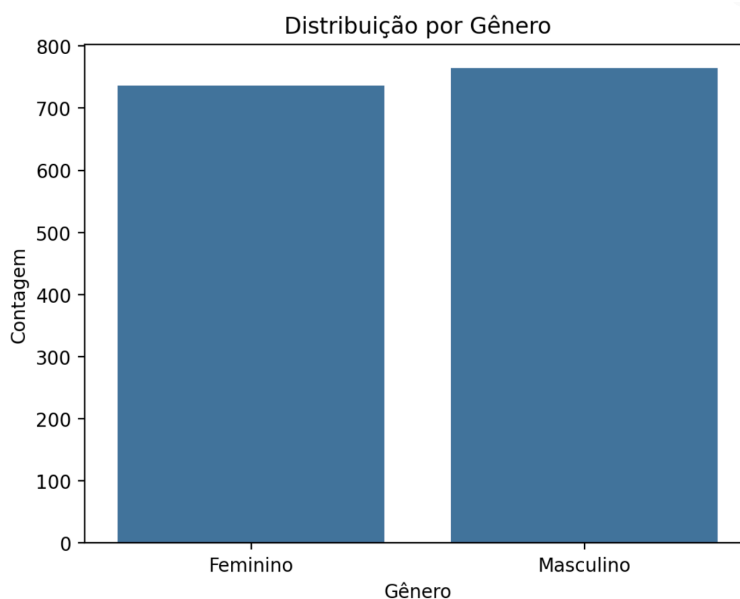


Gráfico 1 - Distribuição de pacientes por gênero.

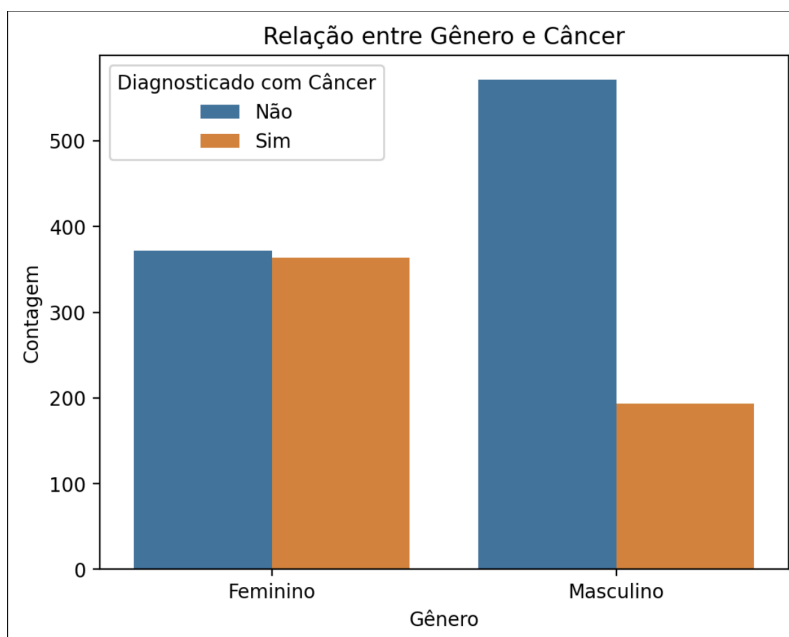


Gráfico 2 - Distribuição de pacientes por gênero e diagnóstico.

Além disso, para melhor compreensão, foram apresentados, em formato de texto, as porcentagens referentes aos valores representados nos gráficos, como é possível visualizar nas figuras, Figura 3 e Figura 4, abaixo.

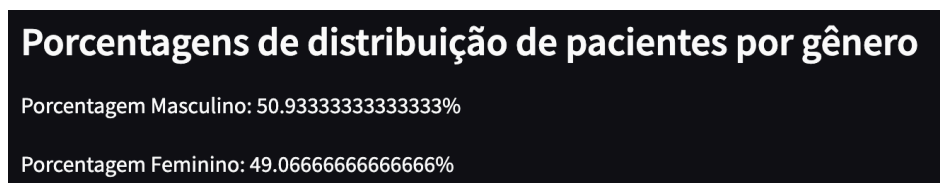


Figura 3 - Exemplo de dados apresentados sobre distribuição de pacientes por gênero.

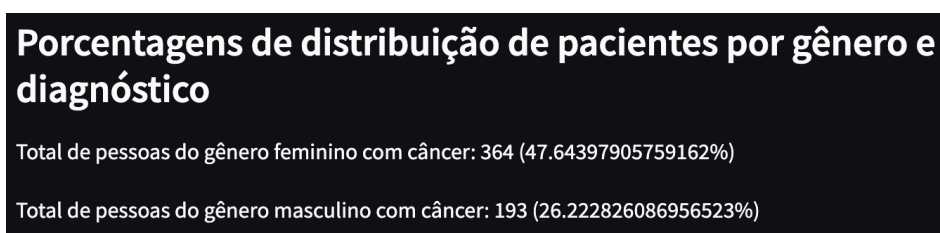


Figura 4 - Exemplo de dados apresentados sobre distribuição de pacientes por gênero e diagnóstico.

Para finalizar a seção de visualização de dados, foi renderizado o *treemap*, assim como no primeiro trabalho desenvolvido, por ser um gráfico muito fácil de compreender e que demonstra de forma simples quais *features* afetam mais o resultado do diagnóstico (Figura 5).

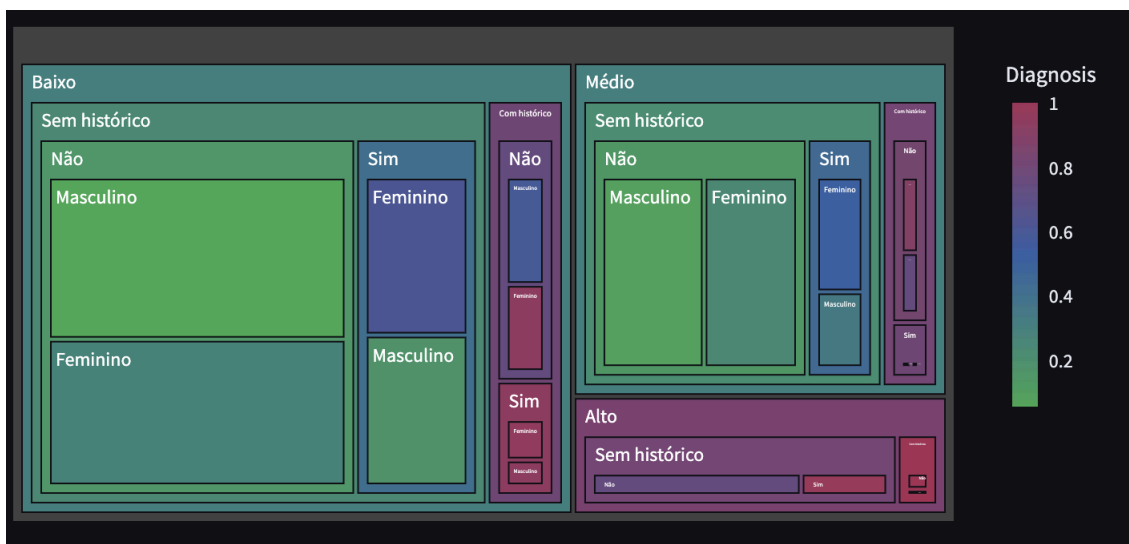


Figura 5 - Apresentação *treemap*.

3.2. Predição do Diagnóstico

Para a predição foi necessário criar uma tela contendo os campos: Idade em Anos, ICM, Tempo de Atividade Física Semanal (em Horas), Quantidade de Álcool Consumido na Semana, Possuí Risco Genético, Gênero, É fumante e Possui Histórico de Câncer. A partir desses campos o usuário poderia selecionar o botão “Predizer Diagnóstico”, que

iria verificar a predição do diagnóstico do paciente. Na figura abaixo é possível observar a tela contendo o formulário, e o resultado da predição.

App Predição de Câncer

☐ Exibir análise dos dados

Predição de Diagnóstico

Idade em Anos: 24 IMC: 23,00

Tempo de Atividade Física Semanal (em Horas): 2,00 Quantidade de Alcool Consumido na Semana: 0,00

Possui Risco Genético: ☒ Baixo, ☐ Médio, ☐ Alto

Gênero: ☐ Feminino, ☒ Masculino

☐ É fumante? ☐ Possui Histórico de Câncer?

Predizer Diagnóstico

Resultado da Predição: Não Diagnosticado

Figura 6. Tela de Predição de Diagnóstico de Câncer.

Para realizar a predição de fato, foi necessário importar o modelo antes treinado, além do *encoder* e do escalonador, dessa forma, foi possível importar esses usando a ferramenta *pickle*, fazendo a leitura dos arquivos *pkl* antes gerados.

Além disso, antes de executar a predição foi necessário realizar a normalização de alguns dados, como o caso de transformar dados booleanos em 0s e 1s (conforme esperado pelo modelo), e transformar alguns dados descritivos para seus respectivos valores em números, como o caso do risco genético e gênero. Adicionalmente, também foi necessário rodar o *encoder* para em cima da variável risco genético, e também o escalonador em todas as variáveis. A partir disso, foi possível chamar a função de predição do modelo, e com o resultado mostrar para o usuário se o paciente foi ou não diagnosticado.

4. Publicação do Modelo na AWS

Após a criação da aplicação foi realizada a publicação da mesma em um ambiente EC2 da AWS, para isso foi utilizado os recursos do *AWS Educate*, que disponibiliza um ambiente AWS para estudos. Na página do *AWS Educate*, foi necessário executar o laboratório, e selecionar a opção do *AWS Console*, que redireciona para uma página Web onde é possível fazer o uso dos serviços da AWS. Nas figuras abaixo é possível verificar tanto a tela de início de laboratório do *AWS Educate*, quanto a tela inicial da AWS.

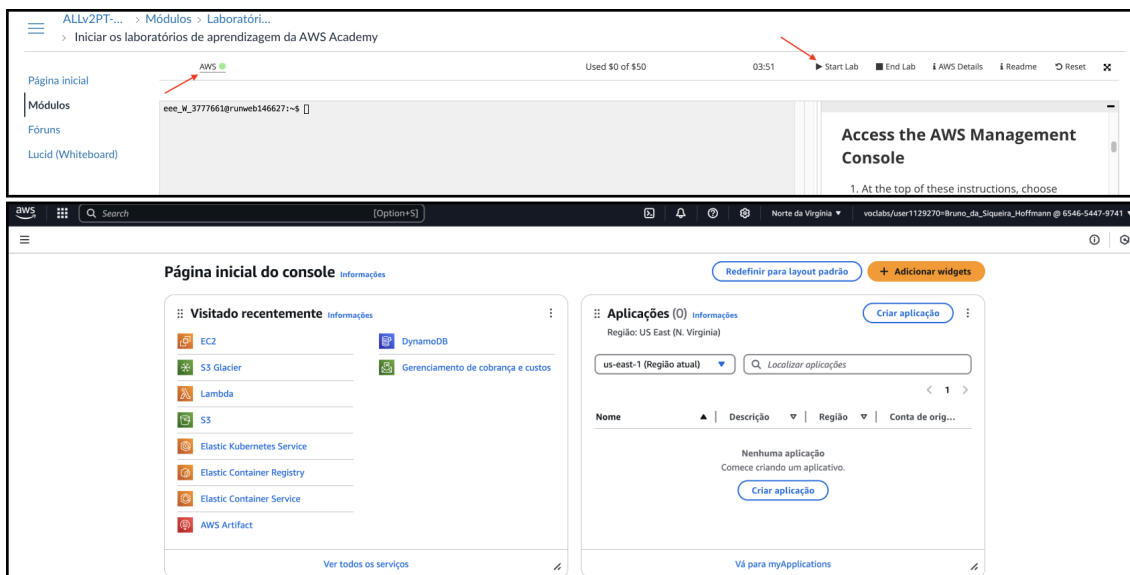


Figura 7. Tela do AWS Educate e Tela Inicial da AWS.

A partir da tela inicial foi selecionado o recurso *EC2*, redirecionando para a página específica do serviço. Assim, foi selecionado o menu “Instâncias”, com o intuito de criar uma nova instância, e a partir deste menu foi selecionada a opção “Executar Instâncias”. Na figura abaixo é possível verificar o submenu “Instâncias”.

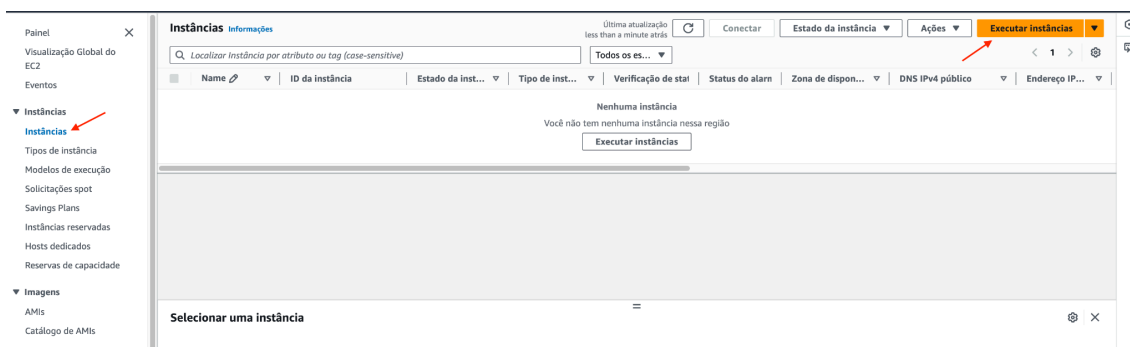


Figura 8. Tela do Submenu Instâncias.

Para configuração da instância, foi indicado o nome “trabalho_gb_bel_bruno”, selecionando a opção Ubuntu como Imagem para sistema operacional. Além disso, a opção de tipo de instância selecionada foi a t2.micro, que contém 1 vCPU e 1 GiB de memória, sendo qualificada para o nível gratuito. Para a opção par de chaves foi selecionada a opção “Prosseguir sem um par de chaves”. E as demais configurações, como rede e armazenamento se manteve as configurações default, sendo elas 8 GiB de armazenamento e Permitir tráfego SSH de Qualquer Lugar para configuração de rede. Com essas configurações foi criada a instância *EC2*, utilizando o botão “Executar Instância”. Na figura abaixo é possível observar as configurações feitas.

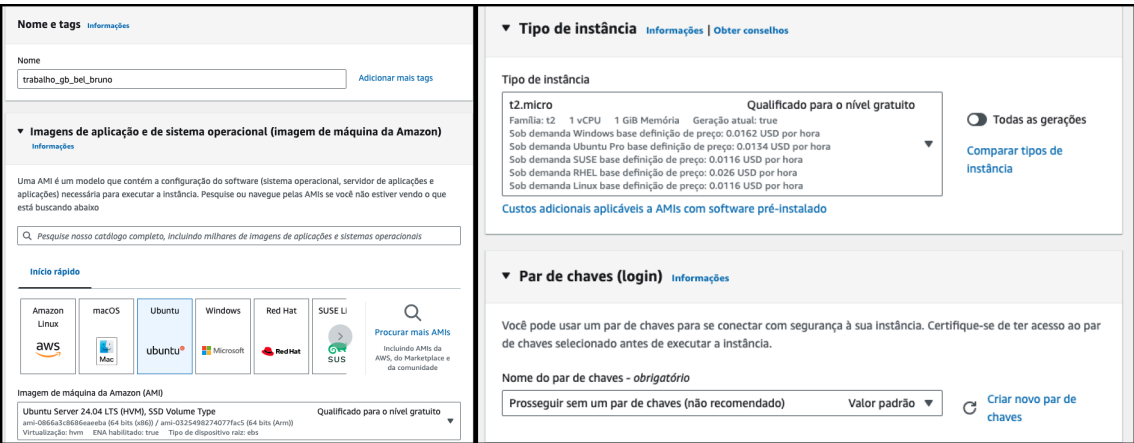


Figura 9. Tela de Configuração da Instância.

Com as configurações feitas foi necessário configurar a parte de segurança, de modo de disponibilizar o acesso para qualquer pessoa na internet para a aplicação desenvolvida. Dessa forma, foi necessário ir no submenu “Grupos de Segurança”, e selecionar o Grupo de Segurança gerado ao criar instância, sendo ele o “sg-007cff332b53bd0ff - launch-wizard-1”. Assim, foi feita a configuração de uma regra de entrada TCP personalizado para a porta 8501, sendo essa a porta utilizada pela a aplicação de predição de diagnóstico. Na figura abaixo é possível observar a regra de entrada criada para o grupo de segurança em questão.

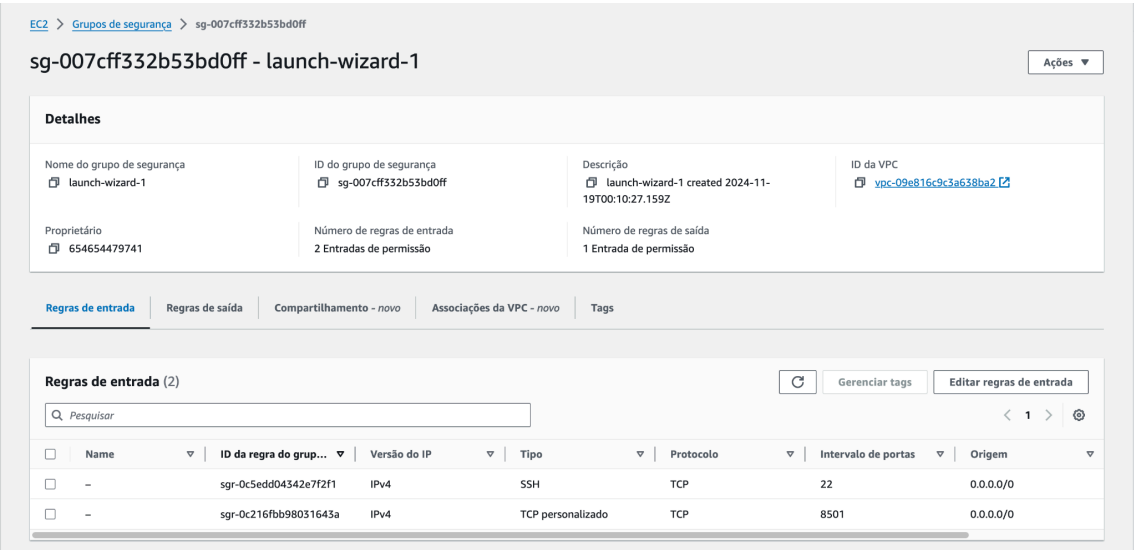


Figura 10. Tela de Configuração dos Grupos de Segurança.

Depois de configurar todo ambiente necessário para executar a aplicação, foi necessário configurar a aplicação de fato. Para isso, foi necessário conectar-se à instância através do *EC2 Instance Connect*, que disponibiliza um terminal na própria página *Web* para realizar os comandos. Na figura abaixo é possível observar a conexão com a instância através do *EC2 Instance Connect*.

EC2 > Instâncias > i-0bdb2bcd859ae6331 > Conectar-se à instância

Conectar-se à instância [Informações](#)

Conecte-se à sua instância i-0bdb2bcd859ae6331 (trabalho_gb_bel_bruno) usando qualquer uma destas opções


Conexão de instância do EC2

Gerenciador de sessões

Cliente SSH

Console de série do EC2


ID da instância

 i-0bdb2bcd859ae6331 (trabalho_gb_bel_bruno)

Tipo de conexão

☒ Conectar-se usando o EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.


☐ Conectar-se usando o endpoint do EC2 Instance Connect
Conecte-se usando o cliente baseado em navegador do EC2 Instance Connect, com um endereço IPv4 privado e um endpoint da VPC.

☒ Endereço IPv4 público
 54.81.66.6

☐ Endereço IPv6
—

Nome de usuário

Insira o nome de usuário definido na AMI usada para iniciar a instância. Se você não definiu um nome de usuário personalizado, use o nome de usuário padrão, ubuntu.

 **Observação:** na maioria dos casos, o nome de usuário padrão, ubuntu, está correto. No entanto, leia as instruções de uso da AMI para verificar se o proprietário da AMI alterou o nome de usuário da AMI padrão.

Cancelar

Conectar

Figura 11. Tela da Ferramenta *EC2 Instance Connect*.

A partir disso, foi necessário executar uma série de comandos, que são apresentados na tabela abaixo. Inicialmente foi feita a configuração do ambiente linux, baixando os pacotes referentes ao python. Em seguida, foi feito o download do repositório *git* contendo o código da aplicação. Também foi necessário executar comandos para criação de um ambiente virtual, e as dependências da aplicação. Por fim, foi executado o comando para executar a aplicação.

Configuração de pacotes do python

```
sudo apt update
```

```
sudo apt-get update
```

```
sudo apt upgrade -y
```

```
sudo apt install python3-pip
```



```
sudo apt install python3.12-venv
```

Download do repositório git

```
git clone https://github.com/BrunoHoffmann15/trab-1-big-data-gb.git
```

Acessar pastas

```
cd trab-1-big-data-gb
```

Configurar venv

```
python3 -m venv venv  
source venv/bin/activate
```

Download das dependências da aplicação

```
pip install -r src/requirements.txt
```

Executar a aplicação

```
streamlit run src/app.py
```

Tabela 1. Tabela com Comandos Executados.

Por fim, ao executar todos os comandos foi possível acessar a aplicação usando o endereço o endereço público mostrado no console ao executar o comando *streamlit run*. Na figura abaixo é possível verificar a aplicação sendo executada no ambiente *AWS*.

Not Secure 54.81.66.6:8501

App Predição de Câncer

☐ Exibir análise dos dados

Predição de Diagnóstico

Idade em Anos	IMC
45	20,00
Tempo de Atividade Física Semanal (em Horas)	Quantidade de Alcool Consumido na Semana
0,00	5,00
Possui Risco Genético	Gênero
<input type="radio"/> Baixo <input checked="" type="radio"/> Médio <input type="radio"/> Alto	<input checked="" type="radio"/> Feminino <input type="radio"/> Masculino
<input checked="" type="checkbox"/> É fumante?	<input checked="" type="checkbox"/> Possui Histórico de Câncer?

Predizer Diagnóstico

Resultado da Predição: **Diagnosticado**

Figura 11. Acesso a Aplicação Web via Internet.

5. Conclusão

O presente trabalho teve como objetivo o desenvolvimento de uma aplicação capaz de prever o diagnóstico de câncer a partir dos dados relacionados ao paciente, e também

a publicação desta aplicação em um ambiente nuvem *AWS EC2*. A partir disso, foi treinado um modelo usando uma base de dados de 1500 pacientes, e utilizando o algoritmo *Random Forest*. Antes de realizar o treinamento foi necessário fazer o tratamento dos dados, fazendo o escalonamento e o *encoder* das colunas.

Como o modelo treinado, foi realizado em seguida o desenvolvimento da aplicação capaz de obter os dados do usuário e fazer a predição. E, para consumo do modelo treinado, foi necessário fazer a exportação dos modelos e *encoders* para que esses estivessem disponíveis para aplicação desenvolvida. Assim, a partir dos dados de entrada do usuário era possível reutilizar esses *encoders* e o modelo para fazer a predição.

Por fim, após o desenvolvimento da aplicação foi possível fazer a publicação e configuração do ambiente utilizando o recurso *EC2* da *AWS*, de maneira bastante simples, precisando fazer as configurações relacionadas à instância *EC2*, os grupos de segurança da mesma e o download das dependências necessárias para poder executar a aplicação na máquina. Assim, foi possível ter a aplicação proposta na nuvem, sendo disponível para qualquer usuário da internet.