Capítulo 6

Desenvolvimento de uma aplicação web para o diagnóstico da COVID-19 usando conceitos, técnicas e ferramentas de Inteligência Artificial

Giovanni L. F. da Silva, Emanuel L. C. de S. Filho, Paulo G. S. Magno, Camyla J. P. Santos, João O. B. Diniz, Jonnison L. Ferreira, Caio E. F. Matos e Aristófanes C. Silva

Abstract

The pandemic caused by COVID-19 has drastically affected the health and well-being of the world's population. However, early detection is essential not only for the treatment and cure of patients, but also for public health, ensuring the isolation of patients. So far, several screening methods have been introduced for the diagnosis of COVID-19, including the analysis of medical images. Recent studies prove that Artificial Intelligence can be useful for a fast and accurate diagnosis. Therefore, the mini-course aims to develop a web application for the diagnosis of COVID-19, through chest radiography images, using concepts, techniques and tools of Artificial Intelligence and Digital Image Processing, being able to provide a second opinion to specialists.

Resumo

A pandemia causada pela COVID-19 afetou drasticamente a saúde e o bem-estar da população mundial. No entanto, a detecção precoce é essencial não só para o tratamento e cura dos pacientes, mas também para a saúde pública, garantindo o isolamento dos pacientes. Até o momento, vários métodos de triagem foram introduzidos para o diagnóstico da COVID-19, incluindo a análise de imagens médicas. Estudos recentes comprovam que a Inteligência Artificial pode ser útil para um diagnóstico rápido e preciso. Portanto, o minicurso tem como objetivo desenvolver uma aplicação web para o diagnóstico da COVID-19, por meio de imagens de radiografia do tórax, usando conceitos, técnicas e ferramentas de Inteligência Artificial e Processamento de Imagens Digitais, podendo fornecer uma segunda opinião aos especialistas.

6.1. Introdução

Em dezembro de 2019, a Organização Mundial da Saúde (OMS) relatou problemas de uma pneumonia não identificada surgindo em Wuhan, China [Zhang et al. 2020]. Posteriormente, o vírus foi nomeado como síndrome respiratória aguda grave coronavírus 2 (SARS-CoV-2) e a doença ficou conhecida como COVID-19, tornando-se em pouco tempo uma pandemia de grandeza mundial, totalizando mais de 578 milhões de infectados e 6,4 milhões de óbitos [Chaudhary and Pachori 2021]. Os sintomas mais frequentes da COVID-19 incluem febre, tosse seca, dificuldade na respiração ou falta de ar, dor no peito, perda de movimentos e a perda de paladar ou olfato [Ghassemi et al. 2021, Chaudhary and Pachori 2021].

A detecção precoce da COVID-19 é um fator crucial não só para o tratamento e cura dos pacientes acometidos, mas também para a saúde pública, garantindo o isolamento dos pacientes e, desta forma, controlando a proliferação da doença [Ghaderzadeh and Asadi 2021]. Assim, as modalidades de imagem médica, como a radiografia do tórax (raio-X), podem desempenhar um papel importante no diagnóstico de pacientes com alta dúvida de infecção de acordo com os sintomas [Huang et al. 2021]. Os benefícios da imagem de raio-X, em comparação com outras modalidades de imagem, incluem fácil acesso, baixo custo e baixo risco de radiação, uma vez que são perigosos para a saúde humana [Ghassemi et al. 2021, Ghaderzadeh and Asadi 2021].

Com o avanço da tecnologia nos últimos tempos, principalmente com o uso de aplicações baseadas em Inteligência Artificial [da Silva et al. 2018, Souza et al. 2019, da Silva et al. 2020, Diniz et al. 2021], tornou-se possível o desenvolvimento de sistemas computacionais que auxiliam o especialista na tarefa de análise e interpretação de imagens, conhecidos como sistemas CAD/CADx [Dias et al. 2021, Chaudhary and Pachori 2021]. Uma vez que o computador consegue interpretar de maneira rápida e ininterrupta várias imagens, estes sistemas podem melhorar a eficiência e aumentar a precisão do diagnóstico da COVID-19 e, consequentemente, fornecer uma segunda opinião aos especialistas [Diniz et al. 2021, Huang et al. 2021].

Neste contexto, o objetivo principal deste minicurso é apresentar o desenvolvimento de uma aplicação web para o diagnóstico da COVID-19, por meio de imagens de radiografia do tórax, usando conceitos, técnicas e ferramentas de Inteligência Artificial. Para isso, a Seção 6.2 apresenta os trabalhos relacionados ao tema abordado disponíveis na literatura, Seção 6.1 descreve os materiais e o método proposto para o diagnóstico da COVID-19 por meio de base de imagens de raio-X públicas. Em seguida, a Seção 6.4 detalha o desenvolvimento da aplicação com a utilização da linguagem de programação Python e apresenta os resultados obtidos pelo método proposto, por fim, a Seção 6.5 retrata as considerações finais e propõe os trabalhos futuros.

6.2. Trabalhos relacionados

Os sistemas CAD/CADx baseados em técnicas de Inteligência Artificial são essenciais na facilitação da triagem da COVID-19 por meio das imagens médicas. Neste sentido, vários trabalhos foram recentemente publicados na literatura visando aumentar a precisão e a agilidade no diagnóstico dos pacientes acometidos pela doença. [Al-Bawi et al. 2020] apresenta um método automático usando a variação da rede neural convolucional

conhecida como VGG aplicada a imagens de raio-X, alcançando uma acurácia de 98,52%. Já [Ghassemi et al. 2021] propõe um método automático para o diagnóstico da COVID-19 usando a rede neural CycleGAN em imagens de tomografia computadorizada (TC), obtendo uma acurácia de 99,60%.

O trabalho proposto por [Dias et al. 2021] demonstra a aplicabilidade das *deep features* e o algoritmo XGBoost no diagnóstico automático da COVID-19 em imagens de raio-X, obtendo uma acurácia de 98,17%. [Chaudhary and Pachori 2021] propõe um método automático utilizando a decomposição baseada em expansão da série de Fourier-Bessel e a rede neural convolucional, alcançando uma acurácia de 100%. No geral, os trabalhos publicados apresentam resultados satisfatórios, tanto em imagens de raio-X quanto em TC, usando bases de imagens públicas na Internet. Uma vez que o objetivo do minicurso é apresentar o desenvolvimento de uma aplicação web para o diagnóstico da COVID-19, o método proposto neste trabalho utiliza técnicas convencionais do Processamento de Imagens Digitais e Inteligência Artificial.

6.3. Materiais e método

O método proposto neste minicurso para o desenvolvimento de uma aplicação web capaz de diagnosticar COVID-19 por meio de imagens de raio-X, ilustrado na Figura 6.1, está dividido em quatro etapas. Em resumo, a primeira etapa consiste na aquisição dos materiais utilizados, como a base de imagens públicas dos exames de radiografia do tórax. Na segunda etapa, a extração de característica é realizada usando descritores de textura, a terceira etapa apresenta o treinamento do modelo preditivo e, por fim, na última etapa, os resultados obtidos pelo método são avaliados.

Figura 6.1. Fluxograma do método proposto para o diagnóstico da COVID-19 em imagens de raio-X.

6.3.1. Materiais

A base de dados utilizada neste minicurso consiste em uma pequena amostra das bases de imagens *COVID-19 Radiography Database* e a *COVID-QU-Ex Dataset*, ambas disponíveis na Internet. A amostra utilizada consiste em 100 imagens de raio-X de pacientes acometidos pela COVID-19 e 100 imagens de raio-X de pacientes saudáveis, totalizando 200 imagens ao todo. Todas as imagens foram padronizadas para a resolução espacial 256 x 256. A Figura 6.2 apresenta exemplos de imagens utilizadas pelo método proposto.

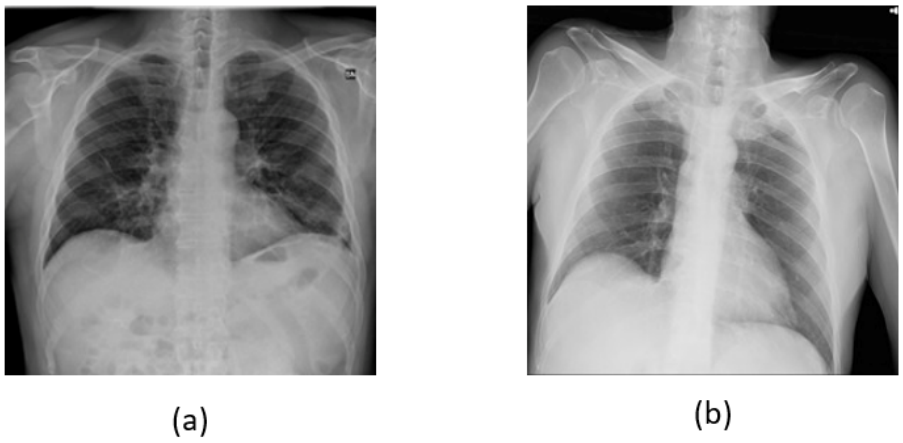

Figura 6.2. Exemplos de imagens de raio-X utilizadas no método proposto.

A radiografia do tórax costuma ser o primeiro exame de imagem usado para ajudar no diagnóstico de doenças e no tratamento de emergência, pois é rápido, fácil e de baixo custo. A partir do exame, pode ser revelado muitas coisas dentro do corpo, incluindo a condição dos pulmões, insuficiência cardíaca, vasos pulmonares, enfisema, COVID-19, o tamanho e o contorno do coração, fraturas, alterações pós-operatórias e posicionamento de dispositivos médicos [Huang et al. 2021].

6.3.2. Extração de características

Uma das tarefas mais complexas presentes na área de Processamento de Imagens Digitais está na definição de um conjunto de características capazes de descrever de maneira efetiva a imagem de entrada [Pedrini and Schwartz 2008]. Segundo [Haralick 1979], a textura pode ser definida como a característica de uma região relacionada a coeficientes de uniformidade, densidade, aspereza, regularidade, intensidade, entre outras características da imagem.

A análise de textura é importante em imagens médicas, pois concedem informações importantes ao realizar a discriminação [Pedrini and Schwartz 2008]. Uma forma clássica de quantificação da textura numa imagem em níveis de cinza é a abordagem estatística, a qual possibilita a descrição da textura através das regras estatísticas que regem tanto a distribuição quanto à relação entre os diferentes níveis de cinza presentes em uma imagem. Portanto, neste minicurso, foram extraídas 14 características de textura proposta por [Haralick 1979].

6.3.3. Modelo preditivo

A etapa de treinamento do modelo preditivo consiste na utilização de algoritmos de Aprendizado de Máquina para a tarefa de reconhecimento de padrões baseado em um conjunto de dados históricos [Janiesch et al. 2021]. Após a etapa de extração de características, os dados que antes estavam no formato não estruturado (imagens) passam a estar estruturado (tabular), portanto, estão aptos para a utilização de técnicas de Aprendizado de Máquina tradicionais, tais como a Máquina de Vetores de Suporte, a Árvore de Decisão, a Regressão Logística, K-Vizinhos Mais Próximos, entre outras.

Os tipos de aprendizado por partes dos algoritmos de Aprendizado de Máquina são compostos por dois grupos principais, supervisionado e não-supervisionado. O aprendizado supervisionado consiste em um processo prévio de treinamento de um classificador para entender os padrões desejados. Posteriormente, o classificador é capaz de identificar o rótulo de qualquer objeto desconhecido na mesma população de treinamento. No aprendizado não-supervisionado, não há informação prévia sobre os rótulos aos quais os padrões de amostras pertencem [Mitchell and Mitchell 1997].

Neste minicurso, o algoritmo XGBoost proposto por [Chen and Guestrin 2016] foi utilizado para o treinamento do modelo preditivo. A escolha pelo XGBoost foi devido ao seu desempenho, em comparação com outros algoritmos disponíveis. Além disso, o XGBoost tem como vantagens a velocidade de predição, escalabilidade do modelo preditivo, baixo consumo de memória e recursos de hardware, sendo um dos algoritmos mais utilizados na literatura atual, alcançando o estado da arte em diversas aplicações [Chen and Guestrin 2016].

O algoritmo XGBoost é baseado no algoritmo de árvores de decisão com aumento de gradiente, do inglês *gradient boosting decision tree*. O algoritmo *boosting* é uma técnica de *ensemble* onde novos modelos são adicionados sequencialmente para corrigir os erros obtidos pelos modelos anteriores, até que não haja melhora no resultado. O *boosting* é constituído essencialmente de três etapas:

1. A partir de um modelo inicial F_0, uma entrada de dados X e a classe y, o erro residual desse modelo é dado por:

$$E_0 = y - F_0(X) \tag{1}$$

   onde $F_0(X)$ é a saída do modelo inicial.

2. Um novo modelo h_1 é treinado então apresentando como rótulo o erro residual do modelo anterior (no caso E_0).

3. Assim, F_0 e h_1 são combinados para gerar F_1:

$$F_1(X) = F_0(X) + h_1(X) \tag{2}$$

   em que o novo erro será dado por:

$$E_1 = y - F_1(X) \tag{3}$$

Esse processo pode ser repetido para a inserção de vários modelos.

Já a técnica de aumento de gradiente, do inglês *gradient boosting*, utiliza um algoritmo de gradiente descendente para minimizar a função de perda ao adicionar novos modelos [Chen and Guestrin 2016]. No caso do algoritmo XGBoost, os modelos utilizados pelo algoritmo são árvores de decisão, porém, a implementação do algoritmo foi feita para ter a maior eficiência possível no uso de recursos de memória e processamento.

A função objetivo do XGBoost é dada por:

$$\ell^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{4}$$

onde $\ell^{(t)}$ é a função de objetivo na iteração t, $\hat{y}_i^{(t-1)}$ é a predição da i-ésima instância na iteração $t-1$, y_i é a classe, $\Omega(f_t)$ é o termo de regularização do modelo, $f_t(x_i)$ é a saída da árvore na iteração t, e g_i e h_i são os gradientes de primeira e segunda ordem da função de perda [Chen and Guestrin 2016].

6.3.4. Validação do modelo preditivo

Em um sistema CAD/CADx, os resultados obtidos pelo modelo preditivo em relação ao diagnóstico podem ser divididos em quatro grupos:

- O teste é positivo e o paciente tem a doença – Verdadeiro Positivo (VP);
- O teste é positivo e o paciente não tem a doença – Falso Positivo (FP);
- O teste é negativo e o paciente tem a doença – Falso Negativo (FN);
- O teste é negativo e o paciente não tem a doença – Verdadeiro Negativo (VN).

Para avaliar o desempenho do modelo preditivo, é comum utilizar o cálculo de algumas estatísticas como a acurácia, *recall*, especificidade e a precisão [Hossin and Sulaiman 2015], obtidos pela matriz de confusão ilustrada na Figura 6.3.

A acurácia representa a probabilidade de pacientes classificados corretamente sobre o total de pacientes existentes, como descrita na Equação 5.

$$ACU = \frac{VP + VN}{VP + VN + FP + FN} \tag{5}$$

O *recall* representa a probabilidade de verdadeiros positivos sobre o total de pacientes acometidos pela COVID-19 existentes, como descrita na Equação 6.

$$REC = \frac{VP}{VP + FN} \tag{6}$$

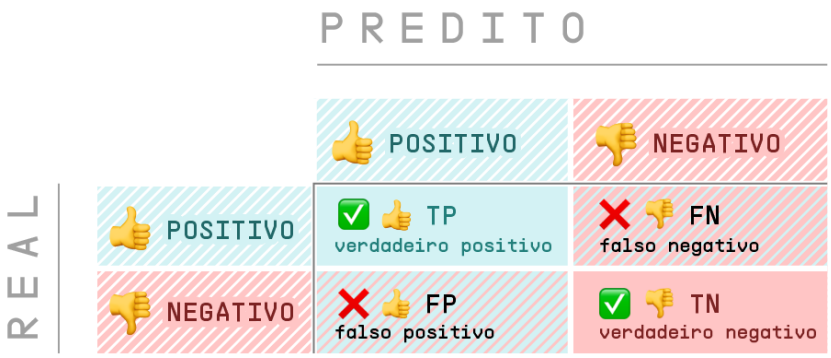

Figura 6.3. Matriz de confusão.

A especificidade representa a probabilidade de verdadeiros negativos sobre o total de pacientes saudáveis existentes, como descrita na Equação 7.

$$ESP = \frac{VN}{VN + FP} \tag{7}$$

A precisão representa a probabilidade de verdadeiros positivos sobre o total de pacientes classificados como acometidos pela COVID-19, como descrita na Equação 8.

$$PREC = \frac{VP}{VP + FP} \tag{8}$$

A acurácia, *recall*, especificidade e precisão foram empregadas neste minicurso para avaliar o desempenho da método proposto para o diagnóstico da COVID-19 por meio de imagens de raio-X, considerando pacientes acometidos pela COVID-19 corretamente classificados como verdadeiros positivos.

6.4. Resultados e discussão

Esta seção detalha o desenvolvimento da aplicação web para apresentação e interpretação dos resultados obtidos usando o método proposto descrito na Seção 6.1. Para isso, toda a aplicação foi desenvolvida na linguagem de programação Python [Millman and Aivazis 2011], usando as bibliotecas OpenCV [Bradski 2000], Mahotas [Coelho 2012], XGBoost [Chen and Guestrin 2016], Imblearn [Mishra 2017], LIME [Ribeiro et al. 2016] e Streamlit [Shukla et al. 2021].

Todo o projeto está disponível no repositório público [1], dividido em três pastas principais: etc, images e src. A pasta etc consiste nos arquivos gerados durante o minicurso, tais como o arquivo de características no formato csv e o modelo preditivo no formato dat. A pasta images contém as 200 imagens de raio-X organizadas em duas subpastas, normal e covid. Por fim, a pasta src contém o *scripts* necessários para o desenvolvimento da aplicação web para o diagnóstico da COVID-19.

[1] https://github.com/giiovannilucca/covid_diagnostic_app.git

A Figura 6.4 apresenta o passo-a-passo para reprodução do ambiente utilizado para o desenvolvimento do minicurso. O arquivo *environment.yml* contém todas as bibliotecas necessárias. Para o melhor entendimento da aplicação desenvolvida de ponta-a-ponta, todos os *scripts* são explicados linha por linha.

```
#Passo 1 - Clonar o repositório no Github

>> git clone https://github.com/giiovannilucca/covid_diagnostic_app.git

#Passo 2 - Instalar as bibliotecas usando o Anaconda

>> conda env create -f environment.yml

#Passo 3 - Ativar o ambiente covid no Anaconda

>> conda activate covid

#Passo 4 - Extrair as características de textura

>> python src/feat_extraction.py

#Passo 5 - Treinar o modelo preditivo

>> python src/model_training.py

#Passo 6 - Executar a aplicação web

>> streamlit run src/web_application.py
```

Figura 6.4. Passo-a-passo para reprodução do ambiente utilizado para o desenvolvimento do minicurso.

6.4.1. Script *feat_extraction.py*

A Figura 6.5 detalha a importação das bibliotecas necessárias para a extração das características das imagens de raio-X (linhas 1 à 5). Entre as bibliotecas mencionadas, destacam-se a Mahotas e a OpenCV. A biblioteca Mahotas disponibiliza os descritores de textura utilizados para descrever os padrões das imagens de raio-X e a OpenCV proporciona diversas operações em imagens digitais, desde a leitura até a aplicação de técnicas de melhoramento de imagens avançadas. A linha 7 inicializa uma lista com os nomes dos rótulos das imagens e a linha 9 inicializa uma lista vazia que armazenará as características extraídas de todas as imagens.

A Figura 6.6 descreve o processo de leitura das imagens e extração das características de textura. O primeiro laço (linha 11) itera sobre a lista dos rótulos, portanto, na primeira iteração lê todas as imagens de raio-X rotuladas como "Normal"e, em seguida, lê todas as imagens de raio-X rotuladas como "COVID". Ao todo, o minicurso foi desenvolvido com 200 imagens, sendo 100 imagens de raio-X de pacientes saudáveis e 100 imagens de raio-X de pacientes acometidos pela COVID-19.

Ainda na Figura 6.6, a linha 19 detalha a leitura da imagem de raio-X usando a biblioteca OpenCV. Para isso, foram passados dois parâmetros para o método *cv.imread*, o primeiro consiste no caminho relativo da imagem de entrada e o segundo consiste no formato de leitura, sistema de cores original ou escala de cinza. O valor zero indica que

```
1 import mahotas
2 import pandas
3 import numpy
4 import glob
5 import cv2
6
7 labels = ['normal', 'covid']
8
9 features_list = list()
```

Figura 6.5. Importação das bibliotecas necessárias para a extração das características.

a imagem foi lida em escala de cinza. A linha 23 ilustra a extração das características de textura usando a biblioteca Mahotas. Por fim, a linha 26 adiciona o rótulo da imagem de entrada, ou seja, o diagnóstico comprovado clinicamente com as demais características de textura e a linha 28 insere as características extraídas a lista que contém todas as demais.

A Figura 6.7 apresenta o processo de exportação da lista de características no formato csv. As linhas 29 e 31 viabilizam a criação do cabeçalho do arquivo csv, ou seja, detalha o nome dos descritores de características extraídas juntamente com a coluna para o rótulo. Posteriormente, as linhas 33 e 35 descrevem a exportação das características para o arquivo chamado *features* usando a biblioteca de análise de dados Pandas [Nelli 2018].

Ao final da execução desde *script*, espera-se que um arquivo chamado *features* no formato csv seja criado na pasta etc do projeto. O arquivo conterá as características para todas as 200 imagens de raio-X utilizadas no minicurso, além do cabeçalho com o nome de todos os descritores de textura. Ressalta-se que o quantitativo de imagens pode

```
11 for label in labels:
12
13     path = './images/' + label
14
15     images_list = glob.glob(path + '/*.png')
16
17     for image_path in images_list:
18
19         img = cv2.imread(image_path, 0)
20
21         label = 1 if 'covid' in image_path else 0
22
23         features_img = mahotas.features.haralick(img, compute_14th_feature = True,
24                                                  return_mean = True)
25
26         features_img = numpy.append(features_img, label)
27
28         features_list.append(features_img)
```

Figura 6.6. Extração das características de textura.

```
features_names = mahotas.features.texture.haralick_labels

features_names = numpy.append(features_names, 'Label')

df = pandas.DataFrame(data = features_list, columns = features_names)

df.to_csv('./etc/features.csv', index = False, sep = ';')
```

Figura 6.7. Exportação da lista de características no formato csv.

ser aumentado facilmente com a inserção de novas imagens nas pastas images/normal e images/covid.

6.4.2. Script *model_training.py*

A Figura 6.8 ilustra as bibliotecas necessárias para o treinamento do modelo preditivo usando o algoritmo XGBoost e a técnica de aumento de dados SMOTE [Chawla et al. 2002]. Para isso, foram utilizadas as bibliotecas Sklearn, Imblearn, Xgboost, Pandas e Pickle. A linha 1 indica a importação do método *train_test_split* necessário para separação do conjunto de treinamento e teste. A linha 2 ilustra a importação do método *confusion_matrix* fundamental para o cálculo das métricas de desempenho.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from imblearn.over_sampling import SMOTE
import xgboost
import pandas
import pickle
```

Figura 6.8. Importação das bibliotecas necessárias para o treinamento do modelo preditivo.

Além disso, na Figura 6.8, a linha 3 indica a importação do método SMOTE usado para aumentar de forma sintética a quantidade de dados no conjunto de treinamento, visando resultados melhores no conjunto de teste. A linha 4 importa a biblioteca XGBoost para a construção do modelo preditivo, a linha 5 indica a importação da biblioteca Pandas para a leitura do arquivo no formato csv e, por fim, a linha 6 apresenta a importação da biblioteca Pickle para o armazenamento e leitura do modelo preditivo em disco local.

A Figura 6.9 apresenta uma função para o cálculo das métricas de desempenho do modelo preditivo descrita na Seção 6.3.4, tais como a acurácia, *recall*, especificidade e a precisão. Todas estas métricas podem ser obtidas com a utilização da matriz de confusão

(linha 9). Neste minicurso, a principal métrica de desempenho é o *recall*, pois representa a taxa de acertos do modelo preditivo para as imagens de raio-X de COVID-19 classificadas corretamente.

```
 8 def get_metrics(y_true, y_pred):
 9     vn, fp, fn, vp = confusion_matrix(y_true, y_pred).ravel()
10     accuracy = (vp + vn) / (vp + fp + fn + vn)
11     recall = vp / (vp + fn)
12     specificity = vn / (vn + fp)
13     precision = vp / (vp + fp)
14
15     return {
16         'accuracy': accuracy,
17         'specificity': specificity,
18         'recall': recall,
19         'precision': precision,
20     }
```

Figura 6.9. Função para o cálculo das métricas de desempenho do modelo preditivo.

A Figura 6.10 apresenta a etapa de separação dos conjuntos de treinamento e teste usando a técnica *hold-out* [Hossin and Sulaiman 2015]. A linha 22 descreve a leitura do arquivo *features* usando a biblioteca Pandas. As linhas 24 e 25 apresentam a separação das características e dos rótulos. Em seguida, a linha 27 separa de forma mutuamente exclusiva os dados de treinamento e teste, usando a proporção de 20% para o conjunto de teste. Por fim, as linhas 29 e 30 ilustram a aplicação da técnica SMOTE no conjunto de treinamento para aumento dos dados. O objetivo é criar novas instâncias afim de garantir uma melhor generalização no modelo preditivo.

```
22 df = pandas.read_csv('./etc/features.csv', delimiter = ';')
23
24 X = df.drop('Label', axis = 1)
25 y = df['Label']
26
27 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
28
29 balanced = SMOTE(random_state=42)
30 X_train, y_train = balanced.fit_resample(X_train, y_train)
```

Figura 6.10. Separação dos conjuntos de treinamento e teste.

A Figura 6.11 detalha a etapa de construção do modelo preditivo usando o algoritmo XGBoost. Para isso, a linha 32 apresenta os hiperparâmetros utilizados no XGBoost. A linha 36 apresenta o treinamento do modelo através do método *fit*, em seguida, a linha 38 armazena as predições após o treinamento do modelo preditivo para o conjunto de teste. A linhas 40 e 42 apresentam os resultados obtidos pelo método

proposto, alcançando uma acurácia de 85%, *recall* de 89,47%, especificidade de 80,95% e precisão de 80,95%.

```
32 model = xgboost.XGBClassifier(objective = "binary:logistic", random_state = 42, max_depth = 9,
                                 colsample_bytree = 0.4033, min_child_weight = 6, gamma = 0.429,
33                                 eta = 0.5995, n_estimators = 1000, use_label_encoder=False,
                                     eval_metric='merror')
34
35 model.fit(X_train, y_train)
36
37 y_pred = model.predict(X_test)
38
39 metrics = get_metrics(y_test, y_pred)
40
41 print(metrics)
42
43 pickle.dump(model, open('./etc/xgb_model.dat', 'wb'))
```

Figura 6.11. Treinamento do modelo preditivo usando o algoritmo XGBoost.

6.4.3. Script *web_application.py*

A Figura 6.12 apresenta as bibliotecas necessárias para a aplicação web usando a biblioteca Streamlit e LIME. De bibliotecas novas, apenas as linhas 2, 3 e 4. O Streamlit é uma biblioteca de código-aberto que facilita a criação e o compartilhamento de aplicativos web personalizados e bonitos para as áreas de Inteligência Artificial e Ciência de Dados. A linha 3 indica a importação da biblioteca LIME para interpretação das predições do modelo preditivo.

```
1 from sklearn.model_selection import train_test_split
2 import streamlit.components.v1 as components
3 import lime.lime_tabular
4 import streamlit
5 import mahotas
6 import pickle
7 import pandas
8 import numpy
9 import cv2
```

Figura 6.12. Importação das bibliotecas necessárias para a aplicação web.

A Figura 6.13 ilustra duas funções auxiliares no desenvolvimento da aplicação web. A primeira consiste na função para o carregamento do modelo preditivo que está localizado na pasta etc (linhas 12 e 13). A criação do modelo preditivo é resultado da execução do *script* anterior, descrito na Seção 6.4.2. A segunda função consiste na conversão da imagem de raio-X de bytes para a biblioteca OpenCV (linhas 15 a 20).

```
12 def get_model():
13     return pickle.load(open('./etc/xgb_model.dat', 'rb'))
14
15 def convert_byteio_image(string):
16     array = numpy.frombuffer(string, numpy.uint8)
17     image = cv2.imdecode(array, flags=1)
18     image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
19
20     return image
```

Figura 6.13. Funções auxiliares na aplicação web.

A Figura 6.14 apresenta a configuração da barra lateral da aplicação web. A linha 22 expõe o título da aplicação e a linha 23 inicializa a barra lateral. Em seguida, a linha 26 apresenta a funcionalidade de *upload* da imagem de entrada, a imagem precisa estar no formato png e o *upload* não aceita mais de uma imagem por vez. Por fim, na linha 28 o modelo preditivo é carregado em memória para a predição dos novos dados, ou seja, as novas imagens de raio-X.

```
22 streamlit.markdown("<h1 style='text-align: center; color: black;'>Web application for the diagnosis of
                       COVID-19 using X-ray images</h1>", unsafe_allow_html=True)
23
24 streamlit.sidebar.title('Settings')
25
26 uploaded_image = streamlit.sidebar.file_uploader("Choose an image", type = 'png',
                                                    accept_multiple_files = False)
27
28 model = get_model()
```

Figura 6.14. Configuração da barra lateral da aplicação web.

A Figura 6.15 demonstra a conversão da imagem em bytes para OpenCV e, em seguida, a extração de características de textura. A linha 30 verifica se alguma imagem foi carregada durante o *upload*, caso tenha acontecido, a imagem passa pelo processo de conversão nas linhas 32 e 34. Posteriormente, a linha 36 verifica a resolução espacial da imagem de entrada, caso ela seja diferente de 256 x 256, a imagem é redimensionada, conforme ilustrado na linha 37. Finalmente, a linha 39 apresenta o processo de extração de características.

A Figura 6.16 apresenta a predição do modelo preditivo (linhas 41 e 43) usando as características de textura extraídas na linha 39. Além disso, as linhas 47 e 49 exibem a imagem de raio-X centralizada, possibilitando a visualização do exame para um especialista. Por último, as linhas 51 e 53 formatam a predição do modelo preditivo em probabilidades, ou seja, além de informar a predição final, o modelo preditivo é capaz de indicar o quão convicto está em sua predição.

```
30 if uploaded_image is not None:
31
32     bytes_data = uploaded_image.getvalue()
33
34     image = convert_byteio_image(bytes_data)
35
36     if (image.shape != (256, 256)):
37         image = cv2.resize(image, (256, 256))
38
39     features = mahotas.features.haralick(image, compute_14th_feature = True,
40                                          return_mean = True).reshape(1, 14)
```

Figura 6.15. Leitura da imagem de entrada e extração de características.

O último trecho do *script* da aplicação web, ilustrado na Figura 6.17, demonstra a utilização da biblioteca LIME para o entendimento dos fatores que levaram ao modelo preditivo tomar aquela decisão em sua predição, ou seja, quais características contribuíram de forma positiva e negativa para a predição do modelo preditivo (linhas 68 a 73). Uma vez que o algoritmo LIME precisa do conjunto de treinamento para a sua interpretação local, foi necessário a leitura do conjunto de treinamento novamente (linhas 61 a 66).

A Figura 6.18 apresenta a interface gráfica da aplicação web desenvolvida com a parte da visualização da imagem de raio-X. Após o carregamento da imagem de entrada, todo o processo de extração de característica e predição do modelo preditivo é iniciado em *background*, assim, que o processo estiver concluído, um texto abaixo da imagem de entrada indica a predição juntamente com a probabilidade assegurada pelo modelo preditivo. Por fim, a Figura 6.19 demonstra a interpretação da predição obtida usando a biblioteca LIME, no qual, as características ressaltadas com a cor laranja indica uma

```
41 pred = model.predict(features)
42
43 probs = model.predict_proba(features)
44
45 streamlit.markdown("<h3 style='text-align: center; color: black;'>Image</h3>",
                        unsafe_allow_html=True)
46
47 col1, col2, col3 = streamlit.columns([0.2, 5, 0.2])
48
49 col2.image(image, use_column_width=True)
50
51 pred_output = "Patient affected by COVID-19 with {:.2%} certainty".format(probs[0]
                 [1]) if pred[0] == 1 else "Patient within the normal range with
                     {:.2%} certainty".format(probs[0][0])
52
53 streamlit.markdown("<h4 style='text-align: center; color: black;'>" + pred_output + "
                        </h4>", unsafe_allow_html=True)
```

Figura 6.16. Predição e visualização da imagem de entrada.

```
55 if streamlit.sidebar.button("Explain prediction"):
56
57     streamlit.markdown("<h3 style='text-align: center; color:
           black;'>Interpretation</h3>", unsafe_allow_html=True)
58
59     with streamlit.spinner('Calculating...'):
60
61         df = pandas.read_csv('./etc/features.csv', delimiter = ';')
62
63         X = df.drop('Label', axis = 1)
64         y = df['Label']
65
66         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                             random_state = 42)
67
68         explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values,
                        feature_names = X.columns,
69                      class_names = ['Normal', 'COVID-19'],
70                      feature_selection = 'lasso_path',
71                      discretize_continuous = True)
72
73         exp = explainer.explain_instance(features.reshape(14,), model.predict_proba,
                                             num_features = 14)
74
75         components.html(exp.as_html(predict_proba = False), height = 800)
76
```

Figura 6.17. Interpretação das predições do modelo preditivo.

contribuição positiva para o rótulo COVID-19. Por outro lado, a cor azul indica uma contribuição positiva para o rótulo normal.

6.5. Considerações finais

Este capítulo de livro apresentou o desenvolvimento de uma aplicação web para o diagnóstico da COVID-19 por meio de imagens de raio-X usando conceitos, técnicas e ferramentas de Inteligência Artificial. Com esse intuito, foram utilizadas imagens de raio-X públicas, disponíveis na Internet. O método proposto usando os descritores de textura de Haralick e o algoritmo XGBoost apresentou resultados satisfatórios, obtendo uma acurácia de 85%, *recall* de 89,47%, especificidade de 80,95% e precisão de 80,95%.

Uma vez que o objetivo principal do minicurso foi apresentar o desenvolvimento de ponta-a-ponta de uma aplicação web baseada em Inteligência Artificial, não foi necessário um grande quantitativo de imagens para validação do método proposto e nem um pipeline de técnicas inovadoras. No entanto, como trabalhos futuros, destacam-se: 1) aplicação de técnicas de melhoramento de imagens, 2) utilização de outros descritores de textura, juntamente com uma seleção de características, 3) execução de outros algoritmos de Aprendizado de Máquina e, por fim, 4) *deploy* da aplicação em um servidor web.

Referências

[Al-Bawi et al. 2020] Al-Bawi, A., Al-Kaabi, K., Jeryo, M., and Al-Fatlawi, A. (2020). Ccblock: an effective use of deep learning for automatic diagnosis of covid-19 using

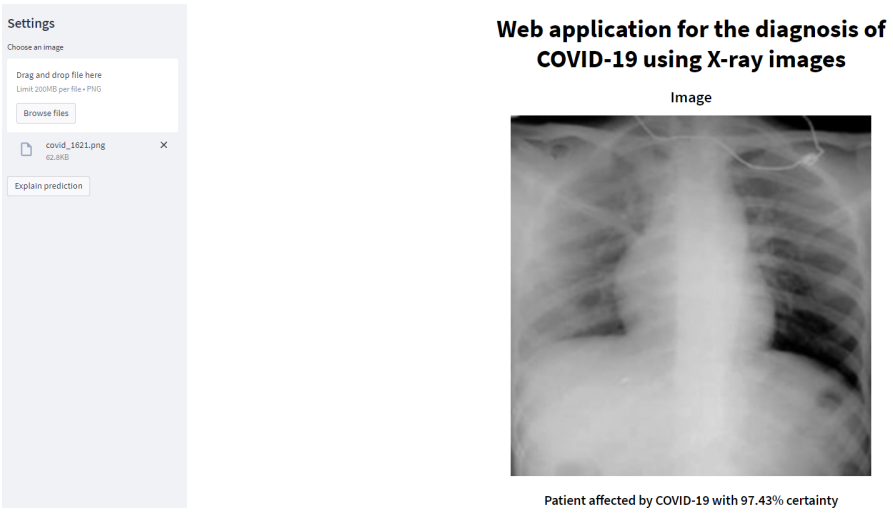

Figura 6.18. Interface gráfica da aplicação web com a visualização da imagem de raio-X.

Figura 6.19. Interface gráfica da aplicação web com a interpretação da predição obtida.

x-ray images. *Research on Biomedical Engineering*, pages 1–10.

[Bradski 2000] Bradski, G. (2000). The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123.

[Chaudhary and Pachori 2021] Chaudhary, P. K. and Pachori, R. B. (2021). Fbsed based automatic diagnosis of covid-19 using x-ray and ct images. *Computers in Biology and Medicine*, 134:104454.

[Chawla et al. 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[Chen and Guestrin 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

[Coelho 2012] Coelho, L. P. (2012). Mahotas: Open source software for scriptable computer vision. *arXiv preprint arXiv:1211.4907*.

[da Silva et al. 2020] da Silva, G. L., Diniz, P. S., Ferreira, J. L., Franca, J. V., Silva, A. C., de Paiva, A. C., and de Cavalcanti, E. A. (2020). Superpixel-based

deep convolutional neural networks and active contour model for automatic prostate segmentation on 3d mri scans. *Medical & Biological Engineering & Computing*, 58(9):1947–1964.

[da Silva et al. 2018] da Silva, G. L. F., Valente, T. L. A., Silva, A. C., de Paiva, A. C., and Gattass, M. (2018). Convolutional neural network-based pso for lung nodule false positive reduction on ct images. *Computer methods and programs in biomedicine*, 162:109–118.

[Dias et al. 2021] Dias, D. A. J., da Cruz, L. B., Diniz, J. O. B., da Silva, G. L. F., Junior, G. B., Silva, A. C., de Paiva, A. C., Nunes, R. A., and Gattass, M. (2021). Automatic method for classifying covid-19 patients based on chest x-ray images, using deep features and pso-optimized xgboost. *Expert Systems with Applications*, 183:115452.

[Diniz et al. 2021] Diniz, J. O., Quintanilha, D. B., Santos Neto, A. C., da Silva, G. L., Ferreira, J. L., Netto, S., Araújo, J. D., Da Cruz, L. B., Silva, T. F., da S Martins, C. M., et al. (2021). Segmentation and quantification of covid-19 infections in ct using pulmonary vessels extraction and deep learning. *Multimedia Tools and Applications*, 80(19):29367–29399.

[Ghaderzadeh and Asadi 2021] Ghaderzadeh, M. and Asadi, F. (2021). Deep learning in the detection and diagnosis of covid-19 using radiology modalities: a systematic review. *Journal of healthcare engineering*, 2021.

[Ghassemi et al. 2021] Ghassemi, N., Shoeibi, A., Khodatars, M., Heras, J., Rahimi, A., Zare, A., Pachori, R. B., and Gorriz, J. M. (2021). Automatic diagnosis of covid-19 from ct images using cyclegan and transfer learning. *arXiv preprint arXiv:2104.11949*.

[Haralick 1979] Haralick, R. M. (1979). Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804.

[Hossin and Sulaiman 2015] Hossin, M. and Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1.

[Huang et al. 2021] Huang, S., Yang, J., Fong, S., and Zhao, Q. (2021). Artificial intelligence in the diagnosis of covid-19: Challenges and perspectives. *International Journal of Biological Sciences*, 17(6):1581.

[Janiesch et al. 2021] Janiesch, C., Zschech, P., and Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3):685–695.

[Millman and Aivazis 2011] Millman, K. J. and Aivazis, M. (2011). Python for scientists and engineers. *Computing in Science & Engineering*, 13(2):9–12.

[Mishra 2017] Mishra, S. (2017). Handling imbalanced data: Smote vs. random undersampling. *Int. Res. J. Eng. Technol*, 4(8):317–320.

[Mitchell and Mitchell 1997] Mitchell, T. M. and Mitchell, T. M. (1997). *Machine learning*, volume 1. McGraw-hill New York.

[Nelli 2018] Nelli, F. (2018). The pandas library—an introduction. In *Python Data Analytics*, pages 87–139. Springer.

[Pedrini and Schwartz 2008] Pedrini, H. and Schwartz, W. R. (2008). *Análise de imagens digitais: princípios, algoritmos e aplicações*. Cengage Learning.

[Ribeiro et al. 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.

[Shukla et al. 2021] Shukla, S., Maheshwari, A., and Johri, P. (2021). Comparative analysis of ml algorithms & stream lit web application. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 175–180. IEEE.

[Souza et al. 2019] Souza, J. C., Diniz, J. O. B., Ferreira, J. L., da Silva, G. L. F., Silva, A. C., and de Paiva, A. C. (2019). An automatic method for lung segmentation and reconstruction in chest x-ray using deep neural networks. *Computer methods and programs in biomedicine*, 177:285–296.

[Zhang et al. 2020] Zhang, J., Xie, Y., Li, Y., Shen, C., and Xia, Y. (2020). Covid-19 screening on chest x-ray images using deep learning based anomaly detection. *arXiv preprint arXiv:2003.12338*, 27.