

✓ MVP- Juliana Fanni

Disciplina: Sprint: Machine Learning & Analytics

Profs. Hugo Villamizar e Patrick Happ

Aluna: Juliana Cardozo Pereira Fanni

✓ 1. Definição do Problema

A instituição é uma organização que tem como objetivo projetos de pesquisas científicas na área de saúde.

Dentre das diversas iniciativas da instituição, temos uma Plataforma API (Assistência & Pesquisa Integradas) que integra a comunidade médica da Oncologia para desenvolver novas pesquisas com o instituto. Contando com uma base unificada de dados clínicos integrada aos banco de dados, para desenvolver estudos que preencham lacunas importantes do conhecimento em oncologia. A missão da Plataforma API é fomentar o avanço da pesquisa em oncologia no Brasil a partir de um modelo inovador de colaboração no qual os médicos responsáveis pelo cuidado também assumem o papel de pesquisadores.

Diante do acima exposto, o objetivo deste trabalho é rastrear e identificar tipos de sinais suspeitos de cancer de pele do paciente e assim classificar como um Cancer contribuindo para o diagnóstico do médico. Esta iniciativa vem se mostrando relevante pela possibilidade da identificação precoce de doenças que apresentam maior probabilidade de cura quando tratadas em estágios iniciais

O dataset usado neste projeto será do https://huggingface.co/datasets/marmal88/skin_cancer, por haver um grande numero de imagens dos diversos tipo de manchas na pele e servir de experimento para o dataset futuro do banco de dados da instituição.

```
import os
import numpy as np
import pandas as pd
import tensorflow as tf
import sklearn.metrics as skm
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from IPython.display import Image, display
from sklearn.linear_model import LinearRegression
from datetime import datetime
from sklearn.metrics import confusion_matrix, classification_report
import itertools
import numpy as np
import sklearn.metrics as skm
import matplotlib.pyplot as plt
from transformers import Trainer, TrainingArguments, AutoModelForSequenceClassification, AutoTokenizer
from keras.preprocessing.image import ImageDataGenerator
from keras import models
from keras import layers
from google.colab import drive # acesso ao drive do google Colab - 'drive.mount'

from tensorflow.keras.preprocessing import image

# instalando a biblioteca gdown
!pip install gdown
import gdown

Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (5.1.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.13.4)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2024.2)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7)

# instalando dependencias
! pip install transformers[torch] datasets evaluate scikit-learn matplotlib
```

```

Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from accelerate>=0.21.0->transformers[torch])
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers[
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers[torch]) (3.7
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers[torch]
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers[torch]
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->transformers[torch]) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch->transformers[torch]) (3.3)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->transformers[torch]) (3.1.3)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch->transformers[torch])
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch->transformers[torch])
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch->transformers[torch])
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch->transformers[torch])
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch->transformers[torch])
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch->transformers[torch])
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch->transformers[torch])
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch->transformers[torch])
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
Collecting nvidia-cuspars-cu12==12.1.0.106 (from torch->transformers[torch])
  Using cached nvidia_cuspars-cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
Collecting nvidia-nccl-cu12==2.19.3 (from torch->transformers[torch])
  Using cached nvidia_nccl_cu12-2.19.3-py3-none-manylinux1_x86_64.whl (166.0 MB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch->transformers[torch])
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Requirement already satisfied: triton==2.2.0 in /usr/local/lib/python3.10/dist-packages (from torch->transformers[torch]) (2.2.0)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torch->transformers[torch])
  Using cached nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2->torch->transformers[torch]
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch->transformers[torch])
Installing collected packages: xxhash, nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cuf
Attempting uninstall: huggingface-hub
  Found existing installation: huggingface-hub 0.20.3
  Uninstalling huggingface-hub-0.20.3:
    Successfully uninstalled huggingface-hub-0.20.3
Successfully installed accelerate-0.29.3 datasets-2.19.0 dill-0.3.8 evaluate-0.4.1 huggingface-hub-0.22.2 multiprocess-0.70.16 n
WARNING: The following packages were previously imported in this runtime:
[huggingface_hub]
You must restart the runtime in order to use newly installed versions.

```

RESTART SESSION

```

from datasets import load_dataset

dataset = load_dataset("marmal88/skin_cancer")

/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Downloading readme: 100%                               3.24k/3.24k [00:00<00:00, 94.2kB/s]
Downloading data: 100%                                521M/521M [00:13<00:00, 43.1MB/s]
Downloading data: 100%                                525M/525M [00:15<00:00, 48.7MB/s]
Downloading data: 100%                                527M/527M [00:13<00:00, 37.9MB/s]
Downloading data: 100%                                528M/528M [00:13<00:00, 45.5MB/s]
Downloading data: 100%                                548M/548M [00:20<00:00, 22.5MB/s]
Downloading data: 100%                                341M/341M [00:06<00:00, 59.3MB/s]
Downloading data: 100%                                348M/348M [00:10<00:00, 31.2MB/s]
Downloading data: 100%                                355M/355M [00:11<00:00, 50.6MB/s]
Generating train split: 100%                            9577/9577 [00:24<00:00, 377.39 examples/s]
Generating validation split: 100%                      2492/2492 [00:05<00:00, 426.48 examples/s]
Generating test split: 100%                            1285/1285 [00:02<00:00, 506.13 examples/s]

```

```

#Analisando o Dataset
dataset

DatasetDict({
  train: Dataset({
    features: ['image', 'image_id', 'lesion_id', 'dx', 'dx_type', 'age', 'sex', 'localization'],
    num_rows: 9577
  })
  validation: Dataset({
    features: ['image', 'image_id', 'lesion_id', 'dx', 'dx_type', 'age', 'sex', 'localization'],
    num_rows: 2492
  })
  test: Dataset({
    features: ['image', 'image_id', 'lesion_id', 'dx', 'dx_type', 'age', 'sex', 'localization'],
    num_rows: 1285
  })
})

```

O dataset já veio separado com as bases train, test e validation. Agora, irei analisar o número de imagens que o dataset possui

```

train_dataset = len(dataset['train'])
val_dataset = len(dataset['validation'])
test_dataset = len(dataset['test'])

tam_train_dataset = len(dataset['train'])
tam_val_dataset = len(dataset['validation'])
tam_test_dataset = len(dataset['test'])

print(f"Temos um total de {tam_test_dataset} para o teste do modelo de deep learning")
print(f"Temos um total de {tam_train_dataset} para o treinamento do modelo de deep learning")
print(f"Temos um total de {tam_val_dataset} para avaliação do modelo de deep learning")

Temos um total de 1285 para o teste do modelo de deep learning
Temos um total de 9577 para o treinamento do modelo de deep learning
Temos um total de 2492 para avaliação do modelo de deep learning

print(f"Informações dos 10 primeiros de avaliação: \n {dataset['test'][:10]}")

print(f"Informações dos 10 primeiros de treinamento: \n {dataset['train'][:10]}")

print(f"Informações dos 10 primeiros de avaliação: \n {dataset['validation'][:10]}")

Informações dos 10 primeiros de avaliação:
{'image': [PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at 0x7EDAFD91DCC0], <PIL.JpegImagePlugin.JpegImageFile i
Informações dos 10 primeiros de treinamento:
{'image': [PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at 0x7EDAFD965E10], <PIL.JpegImagePlugin.JpegImageFile i
Informações dos 10 primeiros de avaliação:
{'image': [PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at 0x7EDAFD965E10], <PIL.JpegImagePlugin.JpegImageFile i

```

2. Acessando e tratando os dados que serão a entrada do modelo de deep learning

```

# Dividindo o dataset em tamanho menor, por muitas vezes dar erro no código por não conseguir executar por falta de memoria
subset_dataset_train = dataset['train'].select(range(4000))
subset_dataset_teste = dataset['test'].select(range(1000))
subset_dataset_aval = dataset['validation'].select(range(1000))

tam_test_dataset = len(subset_dataset_teste)
tam_train_dataset = len(subset_dataset_train)
tam_aval_dataset = len(subset_dataset_aval)

print(f"Temos um total de {tam_test_dataset} para o teste do modelo de deep learning")
print(f"Temos um total de {tam_train_dataset} para o treinamento do modelo de deep learning")
print(f"Temos um total de {tam_aval_dataset} para o avaliação do modelo de deep learning")

Temos um total de 1000 para o teste do modelo de deep learning
Temos um total de 4000 para o treinamento do modelo de deep learning
Temos um total de 1000 para o avaliação do modelo de deep learning

```

Preparação dos dados

```

# Convertendo para um DataFrame pandas o dataset de Treinamento
df_train = pd.DataFrame(subset_dataset_train, columns=subset_dataset_train.features)
df_train['class'] = 0
df_train.head()

```

	image	image_id	lesion_id	dx	dx_ty
0	<PIL.JpegImagePlugin.JpegImageFile image mode=...	ISIC_0024329	HAM_0002954	actinic_keratosis	hi
1	<PIL.JpegImagePlugin.JpegImageFile image mode=...	ISIC_0024372	HAM_0005389	actinic_keratosis	hi
2	<PIL.JpegImagePlugin.JpegImageFile image mode=...	ISIC_0024418	HAM_0003380	actinic_keratosis	hi

Next steps:

[Generate code with df_train](#)[View recommended plots](#)

```
# Convertendo para um DataFrame pandas o dataset de Teste
df_teste = pd.DataFrame(subset_dataset_teste, columns=subset_dataset_teste.features)
df_teste['class'] = 0
df_teste.head()
```

	image	image_id	lesion_id	dx	dx_ty
0	<PIL.JpegImagePlugin.JpegImageFile image mode=...	ISIC_0024329	HAM_0002954	actinic_keratosis	hi
1	<PIL.JpegImagePlugin.JpegImageFile image mode=...	ISIC_0024654	HAM_0005705	actinic_keratosis	hi
2	<PIL.JpegImagePlugin.JpegImageFile image mode=...	ISIC_0024707	HAM_0005448	actinic_keratosis	hi

Next steps:

[Generate code with df_teste](#)[View recommended plots](#)

```
# Convertendo para um DataFrame pandas o dataset de aval
df_aval = pd.DataFrame(subset_dataset_aval, columns=subset_dataset_teste.features)
df_aval['class'] = 0
df_aval.head()
```

	image	image_id	lesion_id	dx	dx_ty
0	<PIL.JpegImagePlugin.JpegImageFile image mode=...	ISIC_0024450	HAM_0005505	actinic_keratosis	hi
1	<PIL.JpegImagePlugin.JpegImageFile image mode=...	ISIC_0024470	HAM_0006220	actinic_keratosis	hi
2	<PIL.JpegImagePlugin.JpegImageFile image mode=...	ISIC_0024511	HAM_0001505	actinic_keratosis	hi

Next steps:

[Generate code with df_aval](#)[View recommended plots](#)

```
def calcula_classe_cancer(cancer):
    vcancer = cancer
    if cancer == 'melanoma' or cancer == 'basal_cell_carcinoma' or cancer == 'melanocytic_Nevi' :
        return "Cancer Positivo"
    else :
        return "Cancer Negativo"
```

```
# Função que atribuiu a variável "class" com valores categóricos
```

```
def define_classe_cancer(cancer):
    vcancer = cancer
    if cancer == 'melanoma' or cancer == 'basal_cell_carcinoma' or cancer == 'melanocytic_Nevi' :
        return 1
    else :
        return 0
```

```
# Atribuiu a variável "class" com valores categóricos no dataset
```

```
for i in range(tam_test_dataset):
    vclasse = define_classe_cancer(df_teste['dx'][i])
    df_teste.loc[i,'class'] = (vclasse)
```

```
for i in range(tam_train_dataset):
    vclasse = define_classe_cancer(df_train['dx'][i])
    df_train.loc[i,'class'] = (vclasse)
```

```
for i in range(tam_aval_dataset):
    vclasse = define_classe_cancer(df_aval['dx'][i])
    df_aval.loc[i,'class'] = (vclasse)
```

```
# Definição do tamanho do batch e as dimensões das imagens
```

```
batch_size = 32
img_height = 224
img_width = 224
num_classes = 5
```

```
#Visualização do número de imagens por classe
```

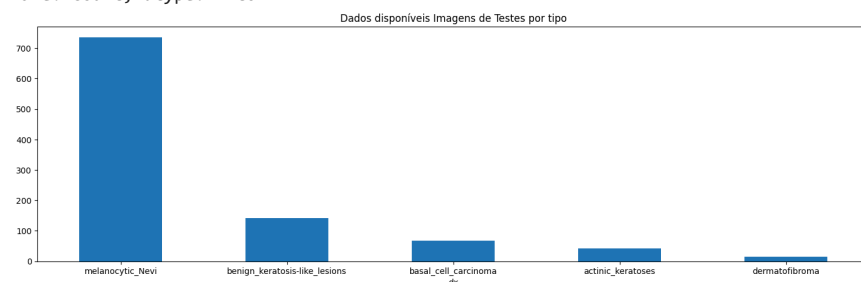
```
print((df_teste['dx']).value_counts())
```

```
plt.figure(figsize=(18, 5))
fire_count = df_teste['dx'].value_counts().plot.bar(title='Dados disponíveis Imagens de Testes por tipo')
plt.subplots_adjust(top=0.9)
plt.xticks(rotation=360)
plt.show()
```

```

dx
melanocytic_Nevi          735
benign_keratosis-like_lesions 142
basal_cell_carcinoma      67
actinic_keratoses         42
dermatofibroma            14
Name: count, dtype: int64

```



```

#Visualização do número de imagens por classe no dataset de teste
print((df_teste['class']).value_counts())

```

```

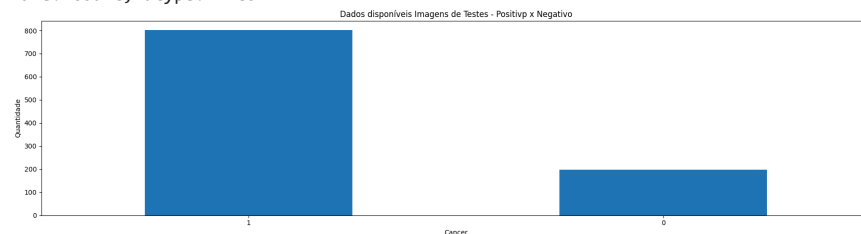
plt.figure(figsize=(18, 5))
fire_count = df_teste['class'].value_counts().plot.bar(title='Dados disponíveis Imagens de Testes - Positiv x Negativo')
plt.subplots_adjust(top=0.9)
plt.xticks(rotation=360)
plt.tight_layout()
plt.ylabel('Quantidade')
plt.xlabel('Cancer')
plt.show()

```

```

class
1      802
0      198
Name: count, dtype: int64

```



```

#Visualização do número de imagens por classe no dataset de treinamento
print((df_train['dx']).value_counts())

```

```

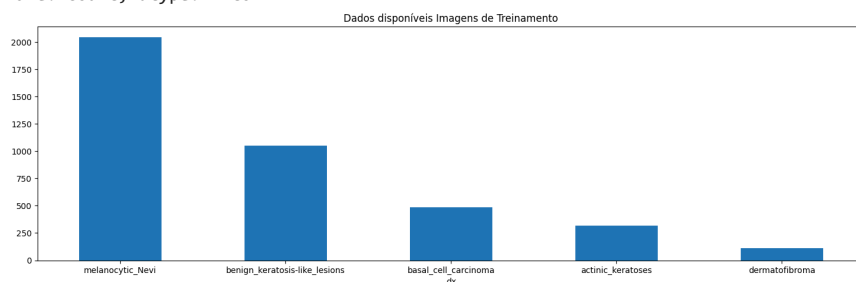
plt.figure(figsize=(18, 5))
fire_count = df_train['dx'].value_counts().plot.bar(title='Dados disponíveis Imagens de Treinamento')
plt.subplots_adjust(top=0.9)
plt.xticks(rotation=360)
plt.show()

```

```

dx
melanocytic_Nevi          2040
benign_keratosis-like_lesions  1048
basal_cell_carcinoma       487
actinic_keratosis         315
dermatofibroma            110
Name: count, dtype: int64

```



```

#Visualização do número de imagens por classe no dataset de teste
print((df_train['class']).value_counts())

```

```

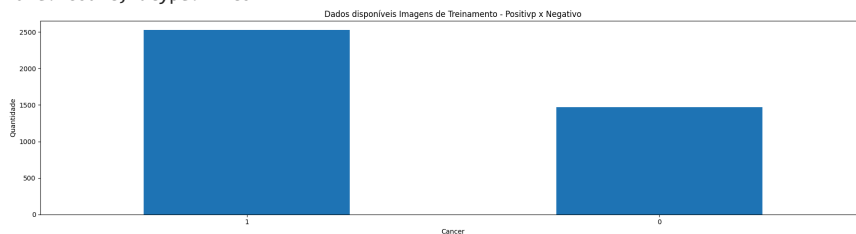
plt.figure(figsize=(18, 5))
fire_count = df_train['class'].value_counts().plot.bar(title='Dados disponíveis Imagens de Treinamento - Positivp x Negativo')
plt.subplots_adjust(top=0.9)
plt.xticks(rotation=360)
plt.tight_layout()
plt.ylabel('Quantidade')
plt.xlabel('Cancer')
plt.show()

```

```

class
1    2527
0    1473
Name: count, dtype: int64

```



```

images = df_teste['image']
print(images)

```

```

0    <PIL.JpegImagePlugin.JpegImageFile image mode=...
1    <PIL.JpegImagePlugin.JpegImageFile image mode=...
2    <PIL.JpegImagePlugin.JpegImageFile image mode=...
3    <PIL.JpegImagePlugin.JpegImageFile image mode=...
4    <PIL.JpegImagePlugin.JpegImageFile image mode=...
...
995  <PIL.JpegImagePlugin.JpegImageFile image mode=...
996  <PIL.JpegImagePlugin.JpegImageFile image mode=...
997  <PIL.JpegImagePlugin.JpegImageFile image mode=...
998  <PIL.JpegImagePlugin.JpegImageFile image mode=...
999  <PIL.JpegImagePlugin.JpegImageFile image mode=...
Name: image, Length: 1000, dtype: object

```

```

df_teste['class'] = df_teste['class'].astype('str')
df_teste['image'] = df_teste['image'].astype('str')

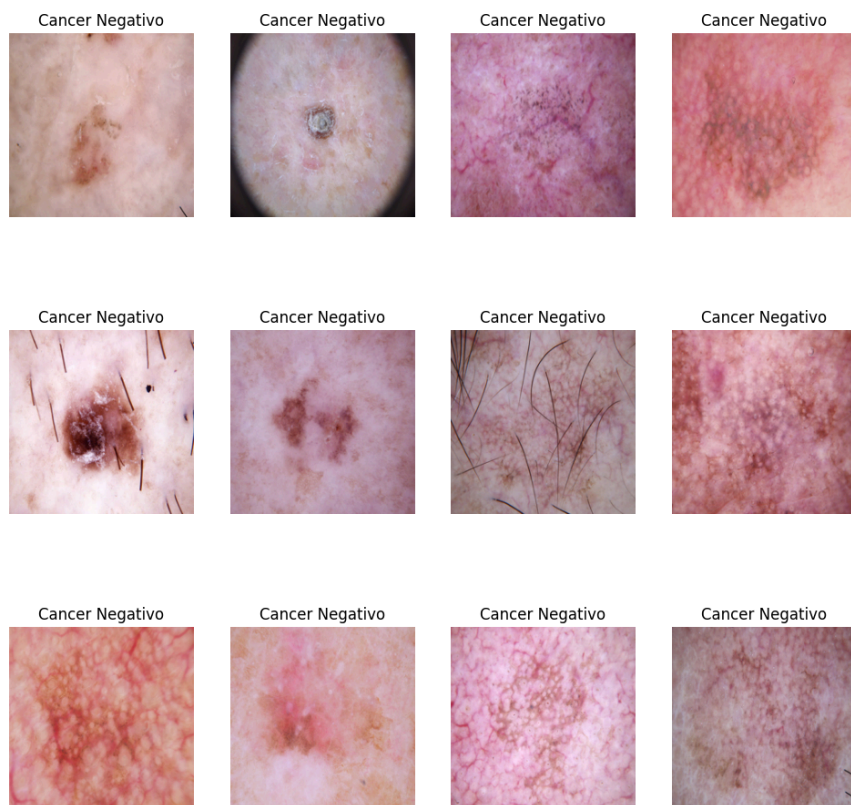
fig = plt.gcf()
fig.set_size_inches(3*4, 3*4)

# Visualizando apenas 12 imagens Teste
for i in range(12):
    sp = plt.subplot(3, 4, i + 1)
    sp.axis('Off')
    img_train = dataset["test"][i]['image'].resize((img_height, img_width))
    verifica_positivo=calcula_classe_cancer(dataset["test"][i]['dx'])
    plt.title(verifica_positivo)
    plt.imshow(img_train)

print(f"Exemplos de 12 imagens de Teste: \n")
plt.axis('off') # Para não exibir os eixos
plt.show()

```

Exemplos de 12 imagens de Teste:




```

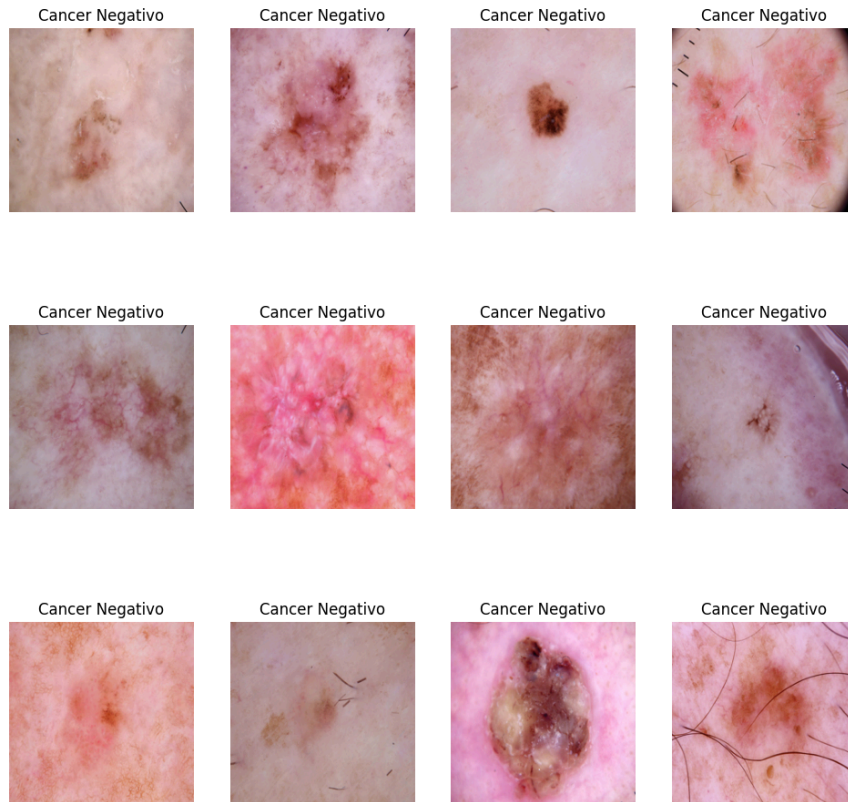
fig = plt.gcf()
fig.set_size_inches(3*4, 3*4)

# Visualizando apenas 12 imagens Treinamento
for i in range(12):
    sp = plt.subplot(3, 4, i + 1)
    sp.axis('Off')
    img_train = dataset["train"][i]['image'].resize((img_height, img_width))
    verifica_positivo=calcula_classe_cancer(dataset["test"][i]['dx'])
    plt.title(verifica_positivo)
    plt.imshow(img_train)

print(f"Exemplos de 12 imagens de Treinamento: \n")
plt.axis('off') # Para não exibir os eixos
plt.show()

```

Exemplos de 12 imagens de Treinamento:



✓ 4. Treinamento do modelo de deep learning

```

path = 'datasets/train'

train_dir = 'datasets/train'
test_dir = 'datasets/test'
val_dir = 'datasets/aval'

train_datagen=ImageDataGenerator(validation_split=0.2, #include validation split
                                rescale = 1.0/255,
                                #rotation_range=40,
                                shear_range=0.2,
                                zoom_range=0.2,
                                horizontal_flip=True,
                                fill_mode='nearest')

test_datagen=ImageDataGenerator(rescale=1./255)


train_generator=train_datagen.flow_from_dataframe(
    df_teste ,train_dir,
    target_size=(150,150),
    batch_size=100, #50
    class_mode='binary',
    x_col='image',
    y_col='class',
    subset='training' #training set
)

val_generator=train_datagen.flow_from_dataframe(
    df_teste ,val_dir,
    target_size=(150,150),
    batch_size=100, #50
    class_mode='binary',
    x_col='image',
    y_col='class',
    subset='validation' #validation set
)

test_generator=test_datagen.flow_from_dataframe(
    df_teste ,test_dir,
    target_size=(150,150),
    batch_size=100, #50
    class_mode='binary',
    x_col='image',
    y_col='class'
)

Found 0 validated image filenames belonging to 0 classes.
Found 0 validated image filenames belonging to 0 classes.
Found 0 validated image filenames belonging to 0 classes.
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 1000 invalid image filename(s) in
warnings.warn(
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 1000 invalid image filename(s) in
warnings.warn(
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 1000 invalid image filename(s) in
warnings.warn(

```



```

model=models.Sequential()
model.add(layers.Conv2D(32 ,(3,3) , activation='relu' , input_shape=(150,150,3) ))
model.add(layers.MaxPooling2D( (2,2)))

model.add(layers.Conv2D(64 , (3,3) , activation='relu'))
model.add(layers.MaxPooling2D( (2,2)))

model.add(layers.Conv2D(128 , (3,3) , activation='relu'))
model.add(layers.MaxPooling2D( (2,2)))

model.add(layers.Conv2D(128 , (3,3) , activation='relu'))
model.add(layers.MaxPooling2D( (2,2)))

model.add(layers.Flatten() )
model.add(layers.Dropout(0.25))
model.add(layers.Dense( 512 ,activation='relu' ))
model.add(layers.Dropout(0.25))
model.add(layers.Dense( 1 , activation='sigmoid' ) )

model.summary()

```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 148, 148, 32)	896