Lung Cancer Classification using Computerized Tomography (CT) Data

Índice

Introdução

[voltar ao índice]

O cancro do pulmão representa um dos tipos de cancro mais comuns e mortais em todo o mundo. A sua elevada mortalidade deve-se ao facto da deteção deste tipo de cancro ser um desafio para os profissionais da área. Para a sua deteção, são usualmente analisadas imagens de tomografia computadorizada, que possibilitam a identificação de nódulos pulmonares. Contudo, a análise deste tipo de dados exige um elevado grau de especialização e experiência, o que pode levar a diagnósticos incorretos. A solução pode passar pela utilização de modelos de Inteligência Artificial, que possam auxiliar os profissionais na deteção e classificação de nódulos pulmonares.

O projeto proposto consiste, então, na classificação de nódulos pulmunares, através da análise detalhada de dados e implementação de modelos de IA, utilizando imagens de tomografia computadorizada (CT). Os dados utilizados são provenientes do dataset LIDC-IDRI, disponível em https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI. Ao longo deste projeto serão propostas soluções para a sua análise e processamento, tendo por base estudos e projetos já realizados na área, que se encontram referenciados ao longo

classificação dos nódulos pulmonares será realizada através de diferentes

técnicas e algoritmos, que serão comparados e avaliados, de forma a determinar qual o modelo que melhor se adequa ao problema proposto.

Para realizar este trabalho foram necessárias utilizar algumas bibliotecas como:

In [2]:
```
import pandas as pd
import pylidc as pl
import numpy as np
import cv2
import matplotlib.pyplot as plt
from radiomics import featureextractor
import SimpleITK as sitk
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from skimage import measure
import seaborn as sns
```

In [3]:
```
# Feature Importance and Selection
# t-test
from scipy.stats import ttest_ind
# Principal Component Analysis
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import scipy.stats as stats
from statsmodels.stats.multicomp import pairwise_tukeyhsd
```

In [4]:
```
# Model Training
from sklearn.model_selection import KFold
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from pyswarm import pso
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
```

Análise de Datasets

[voltar ao índice]

Nesta parte do projeto vamos começar por analisar os diferentes datasets fornecidos, de modo a compreender se existem dados que possam ser descartados, para além disso, é de extrema relevância compreender se todos os datasets devem ou não ser utilizados.

Nodule Counts By Patient

[voltar a Análise de Datasets]

Este dataset contem dados que nos indicam qual o número de nódulos por paciente, estando estes subdivididos em duas categorias:

- Number of Nodules >=3mm
- Number of Nodules <3mm

Começamos por fazer a leitura do ficheiro 'NoduleCountsByPatient.xlsx' que se encontra disponivel no site do dataset fornecido no enunciado do projeto.

In [4]:
```
df1_Nodules = pd.read_excel('NoduleCountsByPatient.xlsx')
```

Confirmamos agora se a leitura do ficheiro foi efetuada corretamente. Para isso começamos por ver se a leitura do nome das colunas está bem.

In [5]:
```
print(df1_Nodules.columns)
```

```
Index(['TCIA Patent ID', 'Total Number of Nodules* ',
       'Number of Nodules >=3mm**', 'Number of Nodules <3mm***', 'Unnamed: 4',
       'Unnamed: 5'],
      dtype='object')
```

Concluimos que existem duas colunas sem nomes, vamos portanto ver quais são os dados que estas possuem. Para isso podemos utilizar head().

In [6]:
```
df1_Nodules.head()
```

Out[6]:

	TCIA Patent ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Unnamed: 4	Unnamed: 5
0	LIDC-IDRI-0001	4	1	3	NaN	NaN
1	LIDC-IDRI-0002	12	1	11	NaN	*total number of lesions that received either ...
2	LIDC-IDRI-0003	4	4	0	NaN	**total number of lesions that received a "nod...
3	LIDC-IDRI-0004	4	1	3	NaN	***total number of lesions that received a "no...
4	LIDC-IDRI-0005	9	3	6	NaN	NaN

Como podemos concluir existem colunas com vários valores NaN. Vamos ver quantas vezes este valor aparece em cada coluna.

In [7]:
```
print(df1_Nodules.isna().sum())
```

```
TCIA Patent ID                1
Total Number of Nodules*      0
Number of Nodules >=3mm**     0
Number of Nodules <3mm***     0
Unnamed: 4                 1019
Unnamed: 5                 1016
dtype: int64
```

Eliminação de colunas

Como percebemos existem vários valores NaN nas colunas 4 e 5. Sendo que na coluna 5, sabemos que na realidade existem mais 3 valores NaN, que estão a ser ocupados por textos descritivos, como se pode ver em head(). Como estas colunas não contribuem para informação relevante no dataset, serão eliminadas.

In [8]:
```
df1_Nodules = df1_Nodules.iloc[:, :-2]
```

Vamos só confirmar se a remoção das duas últimas colunas foi bem efetuada:

In [9]:
```
df1_Nodules.head()
```

Out[9]:

	TCIA Patent ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***
0	LIDC-IDRI-0001	4	1	3
1	LIDC-IDRI-0002	12	1	11
2	LIDC-IDRI-0003	4	4	0
3	LIDC-IDRI-0004	4	1	3
4	LIDC-IDRI-0005	9	3	6

Agora apenas precisamos de averiguar o porquê do NaN na primeira coluna.

In [10]:
```
df1_Nodules.tail()
```

Out[10]:

	TCIA Patent ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***
1014	LIDC-IDRI-1009	2	1	1
1015	LIDC-IDRI-1010	10	1	9
1016	LIDC-IDRI-1011	4	4	0
1017	LIDC-IDRI-1012	1	1	0
1018	NaN	7371	2669	4702

Eliminação de uma entrada

Como podemos ver e como sabemos graças ao site do dataset, a útima linha deste dataframe corresponde à soma do número de nódulos de cada coluna. Neste caso, consideramos que esses valores são irrelevantes e como tal, também podemos eliminar essa linha.

```
df1_Nodules.drop(df1_Nodules.index[-1])
```

Confirmamos agora que a última linha foi corretamente eliminada:

In [12]:
```
df1_Nodules.tail()
```

Out[12]:

	TCIA Patent ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***
1013	LIDC-IDRI-1008	7	6	1
1014	LIDC-IDRI-1009	2	1	1
1015	LIDC-IDRI-1010	10	1	9
1016	LIDC-IDRI-1011	4	4	0
1017	LIDC-IDRI-1012	1	1	0

Concluimos que sim.

Tipos de Dados

In [13]:
```
df1_Nodules.dtypes
```

Out[13]:
```
TCIA Patent ID               object
Total Number of Nodules*      int64
Number of Nodules >=3mm**     int64
Number of Nodules <3mm***     int64
dtype: object
```

Análise Profunda

Nota: Durante a análise dos dados notamos que os nomes das colunas estavam desformatados, pelo que os iremos modificar agora para que fiquem direitos.

In [14]:
```
df1_Nodules.columns = ['Patient_ID', 'Total Number of Nodules*', 'Number of Nodules
                       'Number of Nodules <3mm***']
```

Como sabemos este dataset diz respeito ao número de nódulos que cada paciente tem. Como tal vamos agora apenas confirmar quantos pacientes existem de facto, devendo esses ser 1010.

In [15]:
```
df1_Nodules['Patient_ID'].nunique()
```

Out[15]:
```
1010
```

Concluimos que existem 1010 pacientes diferentes como seria suposto, contudo, existem 1018 entradas, logo existem entradas de pacientes repetidos. Vamos descorbrir quais

In [16]:
```
df1_Nodules[df1_Nodules['Patient_ID'].duplicated(keep=False)]
```

Out[16]:

	Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***
131	LIDC-IDRI-0132	14	6	8
132	LIDC-IDRI-0132	12	8	4
151	LIDC-IDRI-0151	3	1	2
152	LIDC-IDRI-0151	7	1	6
315	LIDC-IDRI-0315	13	7	6
316	LIDC-IDRI-0315	8	5	3
333	LIDC-IDRI-0332	6	5	1
334	LIDC-IDRI-0332	3	2	1
357	LIDC-IDRI-0355	4	1	3
358	LIDC-IDRI-0355	3	2	1
368	LIDC-IDRI-0365	8	1	7
369	LIDC-IDRI-0365	5	1	4
446	LIDC-IDRI-0442	4	3	1
447	LIDC-IDRI-0442	3	3	0
489	LIDC-IDRI-0484	33	2	31
490	LIDC-IDRI-0484	4	2	2

Como seria de esperar alguns pacientes apresentam mais que uma entrada, mas como é que sabemos qual é a correta? Talvez com a ajuda de outros datasets seja possivel obter uma resposta a este problema.

Será que há pacientes sem nódulos?

In [17]:
```
df1_Nodules[df1_Nodules['Total Number of Nodules*'] == 0]
```

Out[17]:

	Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***
306	LIDC-IDRI-0306	0	0	0
307	LIDC-IDRI-0307	0	0	0
323	LIDC-IDRI-0322	0	0	0
367	LIDC-IDRI-0364	0	0	0
445	LIDC-IDRI-0441	0	0	0
512	LIDC-IDRI-0506	0	0	0
546	LIDC-IDRI-0540	0	0	0
570	LIDC-IDRI-0564	0	0	0
579	LIDC-IDRI-0573	0	0	0
617	LIDC-IDRI-0612	0	0	0
673	LIDC-IDRI-0668	0	0	0
694	LIDC-IDRI-0689	0	0	0
721	LIDC-IDRI-0716	0	0	0
736	LIDC-IDRI-0731	0	0	0
765	LIDC-IDRI-0760	0	0	0
892	LIDC-IDRI-0887	0	0	0
905	LIDC-IDRI-0900	0	0	0
942	LIDC-IDRI-0937	0	0	0
980	LIDC-IDRI-0975	0	0	0
1000	LIDC-IDRI-0995	0	0	0

Pelo que pudemos observar existem vários pacientes que não apresentam qualquer tipo de nódulos o que não nos é muito útil para este trabalho, uma vez que o objetivo é classificar nódulos. Como tal estas entradas deverão ser removidas, juntamente com todos os pacientes que também não apresentarem nódulos na coluna "Number of Nodules

>=3mm**", pois como sabemos, apenas temos anotações para nódulos classificados como >= 3mm de diametro e menores que 30 mm.

No entanto, do mesmo modo que sabemos que este dataset apresenta incongruencias, pois tem pacientes repetidos, pode haver mais que datasets em que tal também aconteça. Assim sendo, temos duas opções, eliminar essas entradas ou descobrir a correta, de modo a resolver este problema, puderemos utilizar outros datasets para nos ajudarem a encontrar uma solução.

Patient Diagnoses

[voltar a Análise de Datasets]

Este dateset fornece informações valiosas sobre os pacientes e os seus nódulos, como o facto deste apresentarem ou não tumores malignos.

Começamos por fazer a leitura do ficheiro 'PatientDiagnoses.xlsx' que se encontra disponivel no site do dataset fornecido no enunciado do projeto.

In [18]:
```
df2_Diagnoses = pd.read_excel('PatientDiagnoses.xls')
```

Confirmamos agora se a leitura do ficheiro foi efetuada corretamente. Para isso começamos por ver se a leitura do nome das colunas foi executada corretamente.

In [19]:
```
df2_Diagnoses.columns
```

Out[19]:
```
Index(['TCIA Patient ID',
       'Diagnosis at the Patient Level\n0=Unknown\n1=benign or non-malignant disea
se\n2= malignant, primary lung cancer\n3 = malignant metastatic\n',
       'Diagnosis Method\n0 = unknown\n1 = review of radiological images to show 2
years of stable nodule\n2 = biopsy\n3 = surgical resection\n4 = progression or res
ponse',
       'Primary tumor site for metastatic disease',
       'Nodule 1\nDiagnosis at the Nodule Level \n0=Unknown\n1=benign or non-malig
nant disease\n2= malignant, primary lung cancer\n3 = malignant metastatic)\n',
       'Nodule 1\nDiagnosis Method at the Nodule Level\n0 = unknown\n1 = review of
radiological images to show 2 years of stable nodule\n2 = biopsy\n3 = surgical res
ection\n4 = progression or response\n',
       'Nodule 2\nDiagnosis at the Nodule Level \n0=Unknown\n1=benign or non-malig
nant disease\n2= malignant, primary lung cancer\n3 = malignant metastatic)\n',
       'Nodule 2\nDiagnosis Method at the Nodule Level\n0 = unknown\n1 = review of
radiological images to show 2 years of stable nodule\n2 = biopsy\n3 = surgical res
ection\n4 = progression or response\n',
       'Nodule 3\nDiagnosis at the Nodule Level \n0=Unknown\n1=benign or non-malig
nant disease\n2= malignant, primary lung cancer\n3 = malignant metastatic)\n',
       'Nodule 3\nDiagnosis Method at the Nodule Level\n0 = unknown\n1 = review of
radiological images to show 2 years of stable nodule\n2 = biopsy\n3 = surgical res
ection\n4 = progression or response\n',
       'Nodule 4\nDiagnosis at the Nodule Level \n0=Unknown\n1=benign or non-malig
nant disease\n2= malignant, primary lung cancer\n3 = malignant metastatic)\n',
       'Nodule 4\nDiagnosis Method at the Nodule Level\n0 = unknown\n1 = review of
radiological images to show 2 years of stable nodule\n2 = biopsy\n3 = surgical res
ection\n4 = progression or response\n',
       'Nodule 5\nDiagnosis at the Nodule Level \n0=Unknown\n1=benign or non-malig
nant disease\n2= malignant, primary lung cancer\n3 = malignant metastatic)\n',
       'Nodule 5\nDiagnosis Method at the Nodule Level\n0 = unknown\n1 = review of
radiological images to show 2 years of stable nodule\n2 = biopsy\n3 = surgical res
ection\n4 = progression or response\n'],
      dtype='object')
```

Mudar o nome das colunas

Como podemos perceber, ao fazer a leitura do ficheiro, os nomes das colunas ficaram complexos. Vamos portanto simplificá-los:

In [20]:
```
df2_Diagnoses.columns = ['Patient_ID', 'Diagnosis at the Patient Level', 'Diagnosis
                         'Primary tumor site for metastatic disease',
                         'Nodule 1 Diagnosis at the Nodule Level', 'Nodule 1 Diagnc
                         'Nodule 2 Diagnosis at the Nodule Level', 'Nodule 2 Diagnc
                         'Nodule 3 Diagnosis at the Nodule Level', 'Nodule 3 Diagnc
                         'Nodule 4 Diagnosis at the Nodule Level', 'Nodule 4 Diagnc
                         'Nodule 5 Diagnosis at the Nodule Level', 'Nodule 5 Diagnc
```

Agora vamos confirmar se as alterações foram efetuadas corretamente:

In [21]:
```
df2_Diagnoses.columns
```

Out[21]:
```
Index(['Patient_ID', 'Diagnosis at the Patient Level',
       'Diagnosis Method at the Nodule',
       'Primary tumor site for metastatic disease',
       'Nodule 1 Diagnosis at the Nodule Level',
       'Nodule 1 Diagnosis Method at the Nodule Level',
       'Nodule 2 Diagnosis at the Nodule Level',
       'Nodule 2 Diagnosis Method at the Nodule Level',
       'Nodule 3 Diagnosis at the Nodule Level',
       'Nodule 3 Diagnosis Method at the Nodule Level',
       'Nodule 4 Diagnosis at the Nodule Level',
       'Nodule 4 Diagnosis Method at the Nodule Level',
       'Nodule 5 Diagnosis at the Nodule Level',
       'Nodule 5 Diagnosis Method at the Nodule Level'],
      dtype='object')
```

Concluimos que sim.

Mas será que há a necessidade de haver todas estas colunas?

Vamos ver se existem valores NaN

In [22]:
```
print(df2_Diagnoses.isna().sum())
```

```
Patient_ID                                          0
Diagnosis at the Patient Level                      0
Diagnosis Method at the Nodule                      0
Primary tumor site for metastatic disease           0
Nodule 1 Diagnosis at the Nodule Level             20
Nodule 1 Diagnosis Method at the Nodule Level      20
Nodule 2 Diagnosis at the Nodule Level            137
Nodule 2 Diagnosis Method at the Nodule Level     137
Nodule 3 Diagnosis at the Nodule Level            156
Nodule 3 Diagnosis Method at the Nodule Level     156
Nodule 4 Diagnosis at the Nodule Level            157
Nodule 4 Diagnosis Method at the Nodule Level     157
Nodule 5 Diagnosis at the Nodule Level            157
Nodule 5 Diagnosis Method at the Nodule Level     157
dtype: int64
```

Sabemos que o dataset tem 157 entradas, logo as últimas quatro colunas não possuem quaisquer dados.

Mas será que as colunas que dizem respeito ao Nódulo 3 tem de facto uma entrada com dados, ou será algum erro de leitura?

In [23]:
```
print(df2_Diagnoses['Nodule 3 Diagnosis at the Nodule Level'].value_counts())
```

```
Nodule 3 Diagnosis at the Nodule Level
     1
Name: count, dtype: int64
```

In [24]:
```
df2_Diagnoses[df2_Diagnoses['Nodule 3 Diagnosis at the Nodule Level'] == ' ']
```

Out[24]:

	Patient_ID	Diagnosis at the Patient Level	Diagnosis Method at the Nodule	Primary tumor site for metastatic disease	Nodule 1 Diagnosis at the Nodule Level	Nodule 1 Diagnosis Method at the Nodule Level	Nodule 2 Diagnosis at the Nodule Level	Nodule 2 Diagnosis Method at the Nodule Level	No Dia N
114	LIDC-IDRI-0285	3	3	melanoma	3.0	3.0	0.0	0.0	

In [25]:
```
print(df2_Diagnoses['Nodule 3 Diagnosis Method at the Nodule Level'].value_counts()
```

```
Nodule 3 Diagnosis Method at the Nodule Level
     1
Name: count, dtype: int64
```

In [26]:
```
df2_Diagnoses[df2_Diagnoses['Nodule 3 Diagnosis Method at the Nodule Level'] == ' '
```

Out[26]:

	Patient_ID	Diagnosis at the Patient Level	Diagnosis Method at the Nodule	Primary tumor site for metastatic disease	Nodule 1 Diagnosis at the Nodule Level	Nodule 1 Diagnosis Method at the Nodule Level	Nodule 2 Diagnosis at the Nodule Level	Nodule 2 Diagnosis Method at the Nodule Level	No Dia N
114	LIDC-IDRI-0285	3	3	melanoma	3.0	3.0	0.0	0.0	

Eliminar Colunas

Concluimos que é um erro de leitura e que também não tem dados. Por este motivo vamos eliminar as colunas que dizem respeito aos nódulos 3, 4 e 5.

In [27]:
```
df2_Diagnoses = df2_Diagnoses.iloc[:, :-6]
```

Vamos confirmar se as colunas foram removidas corretamente:

In [28]:
```
df2_Diagnoses.head()
```

Out[28]:

	Patient_ID	Diagnosis at the Patient Level	Diagnosis Method at the Nodule	Primary tumor site for metastatic disease	Nodule 1 Diagnosis at the Nodule Level	Nodule 1 Diagnosis Method at the Nodule Level	Nodule 2 Diagnosis at the Nodule Level	Nodule 2 Diagnosis Method at the Nodule Level
0	LIDC-IDRI-0068	3	4	Head & Neck Cancer	3.0	4.0	NaN	NaN
1	LIDC-IDRI-0071	3	1	Head & Neck	1.0	1.0	NaN	NaN
2	LIDC-IDRI-0072	2	4	Lung Cancer	1.0	4.0	NaN	NaN
3	LIDC-IDRI-0088	3	0	Uterine Cancer	0.0	0.0	NaN	NaN
4	LIDC-IDRI-0090	2	3	NSCLC	2.0	3.0	NaN	NaN

Concluimos que sim.

Tipo de Dados

In [29]:
```
df2_Diagnoses.dtypes
```

Out[29]:
```
Patient_ID                                        object
Diagnosis at the Patient Level                     int64
Diagnosis Method at the Nodule                     int64
Primary tumor site for metastatic disease         object
Nodule 1 Diagnosis at the Nodule Level           float64
Nodule 1 Diagnosis Method at the Nodule Level    float64
Nodule 2 Diagnosis at the Nodule Level           float64
Nodule 2 Diagnosis Method at the Nodule Level    float64
dtype: object
```

Os valores dos nódulos 1 e 2 deveriam ser inteiros e não floats. Para convertermos uma coluna de floats em inteiros é preciso que não hajam valores NaN. Para isso iremos primeiro converter os valores NaN em -1 e depois passamos as colunas para inteiros.

In [30]:
```
colunas_para_converter = [
    'Nodule 1 Diagnosis at the Nodule Level',
    'Nodule 1 Diagnosis Method at the Nodule Level',
    'Nodule 2 Diagnosis at the Nodule Level',
    'Nodule 2 Diagnosis Method at the Nodule Level'
]

# Passo 1: Preencher NaN com -1
df2_Diagnoses[colunas_para_converter] = df2_Diagnoses[colunas_para_converter].fillr

# Passo 2: Converter as colunas para int
df2_Diagnoses[colunas_para_converter] = df2_Diagnoses[colunas_para_converter].astyp

# Exibe o DataFrame atualizado
df2_Diagnoses.head()
```

Out[30]:

	Patient_ID	Diagnosis at the Patient Level	Diagnosis Method at the Nodule	Primary tumor site for metastatic disease	Nodule 1 Diagnosis at the Nodule Level	Nodule 1 Diagnosis Method at the Nodule Level	Nodule 2 Diagnosis at the Nodule Level	Nodule 2 Diagnosis Method at the Nodule Level
0	LIDC-IDRI-0068	3	4	Head & Neck Cancer	3	4	-1	-1
1	LIDC-IDRI-0071	3	1	Head & Neck	1	1	-1	-1
2	LIDC-IDRI-0072	2	4	Lung Cancer	1	4	-1	-1
3	LIDC-IDRI-0088	3	0	Uterine Cancer	0	0	-1	-1
4	LIDC-IDRI-0090	2	3	NSCLC	2	3	-1	-1

Como podemos ver a conversão foi realizada com sucesso. Poderiamos voltar a passar os valores '-1' para NaN, mas pensamos que tal não seja necessário, pois sabemos que neste dataset todos os valores que são -1, correspondem a valores NaN.

Análise Profunda

Anteriormente, quando vimos os valores NaN de cada coluna, havia entradas com esse valor nas colunas que dizem respeito ao Nódulo 1:

In [31]:
```
df2_Diagnoses[df2_Diagnoses['Nodule 1 Diagnosis Method at the Nodule Level'] == -1]
```

Out[31]:

	Patient_ID	Diagnosis at the Patient Level	Diagnosis Method at the Nodule	Primary tumor site for metastatic disease	Nodule 1 Diagnosis at the Nodule Level	Nodule 1 Diagnosis Method at the Nodule Level	Nodule 2 Diagnosis at the Nodule Level	Nodul Diagn Meth at Nod Le
120	LIDC-IDRI-0314	2	2	adenocarcinoma	-1	-1	-1	
121	LIDC-IDRI-0325	2	2	squamous cell carcinoma/non-small cell	-1	-1	-1	
126	LIDC-IDRI-0405	1	2	reactive mesothelial cells	-1	-1	-1	
127	LIDC-IDRI-0454	2	2	non-small cell carcinoma	-1	-1	-1	
130	LIDC-IDRI-0510	1	2	focal fibrosis	-1	-1	-1	
131	LIDC-IDRI-0522	2	2	Adenocarcinoma	-1	-1	-1	
132	LIDC-IDRI-0543	2	2	small cell carcinoma	-1	-1	-1	
134	LIDC-IDRI-0562	2	3	adenocarcinoma	-1	-1	-1	
136	LIDC-IDRI-0576	0	2	reactive mesothelial cells	-1	-1	-1	
139	LIDC-IDRI-0624	1	3	Histoplasmosis with necrotizing granulomas	-1	-1	-1	
140	LIDC-IDRI-0766	2	2	non-small cell carcinoma	-1	-1	-1	
141	LIDC-IDRI-0771	2	2	squamous cell carcinoma	-1	-1	-1	
143	LIDC-IDRI-0811	2	3	Adenocarcinoma	-1	-1	-1	
145	LIDC-IDRI-0875	2	3	Adenocarcinoma	-1	-1	-1	
146	LIDC-IDRI-0893	3	3	recurrent/metastatic	-1	-1	-1	
147	LIDC-IDRI-0905	2	3	RLL NSCLC stage 1A	-1	-1	-1	
148	LIDC-IDRI-0921	2	2	small cell carcinoma	-1	-1	-1	
149	LIDC-IDRI-0924	2	3	poorly differentiated adenocarcinoma	-1	-1	-1	
151	LIDC-IDRI-0965	2	3	Adenocarcinoma	-1	-1	-1	
	LIDC-IDRI-	2	2	non-small cell carcinoma	-1	-1	-1	

Ou seja, existem entradas que não apresentam qualquer tipo de dados relativamente aos seus nós.

Será que há pacientes em duplicado?

In [32]:
```
df2_Diagnoses[df2_Diagnoses['Patient_ID'].duplicated(keep=False)]
```

Out[32]:

	Patient_ID	Diagnosis at the Patient Level	Diagnosis Method at the Nodule	Primary tumor site for metastatic disease	Nodule 1 Diagnosis at the Nodule Level	Nodule 1 Diagnosis Method at the Nodule Level	Nodule 2 Diagnosis at the Nodule Level	Nodule 2 Diagnosis Method at the Nodule Level

Não há pacientes em duplicado.

Quantos pacientes há?

In [33]:
```
df2_Diagnoses['Patient_ID'].nunique()
```

Out[33]:
```
157
```

Tendo em conta que deveriam ser 1010 pacientes, percebemos que este dataset encontra-se incompleto.

Mas de entre estes pacientes, será que há alguns que na realidade não tenham dados relevantes?

In [33]:
```
df2_Diagnoses.tail()
```

Out[33]:

	Patient_ID	Diagnosis at the Patient Level	Diagnosis Method at the Nodule	Primary tumor site for metastatic disease	Nodule 1 Diagnosis at the Nodule Level	Nodule 1 Diagnosis Method at the Nodule Level	Nodule 2 Diagnosis at the Nodule Level	Nodule 2 Diagnosis Method at the Nodule Level
152	LIDC-IDRI-0994	2	3	LUL Large cell CA	2	3	-1	-1
153	LIDC-IDRI-1002	2	2	non-small cell carcinoma	-1	-1	-1	-1
154	LIDC-IDRI-1004	2	3	LUL NSCLC	2	3	-1	-1
155	LIDC-IDRI-1010	0	0	lymphoma	0	0	-1	-1
156	LIDC-IDRI-1011	3	2	small-cell carcinoma of the tongue	3	2	3	4

Ao analisarmos a parte final do dataset, por exemplo, percebemos que existem entradas dos apresentam qualquer classificação. Para além disso, encontramos também

casos de entradas que tem todas as colunas com valor 0, o que não faz sentido manter, porque isso significa que não sabem qualquer tipo de informações sobre esses dados.

Metadata

[voltar a Análise de Datasets]

Este dataset também foi obtido através do site do fornecido no enunciado do projeto e contem algumas informações relativas às imagens extraidas, como o número de imagens que cada paciente deve ter.

In [34]:
```
df3_metadata = pd.read_csv('MetadataDigest.csv')
```

Vamos ver o que é que este dataset tem:

In [35]:
```
df3_metadata.head()
```

Out[35]:

	Subject ID	Study UID	Study Description	Study Date	
0	LIDC-IDRI-1001	1.3.6.1.4.1.14519.5.2.1.6279.6001.281499745765...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519.5.2
1	LIDC-IDRI-0778	1.3.6.1.4.1.14519.5.2.1.6279.6001.174809695196...	CHEST	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519.5.2
2	LIDC-IDRI-0813	1.3.6.1.4.1.14519.5.2.1.6279.6001.139110171863...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519.5.2
3	LIDC-IDRI-0710	1.3.6.1.4.1.14519.5.2.1.6279.6001.116146223752...	CHEST	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519.5.2
4	LIDC-IDRI-0410	1.3.6.1.4.1.14519.5.2.1.6279.6001.818775038273...	CT LUNG SCREEN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519.5.2

Temos dados de quantos pacientes?

In [36]:
```
df3_metadata['Subject ID'].value_counts()
```

Out[36]:
```
Subject ID
LIDC-IDRI-0132    3
LIDC-IDRI-0151    3
LIDC-IDRI-0034    2
LIDC-IDRI-0258    2
LIDC-IDRI-0020    2
                 ..
LIDC-IDRI-0478    1
LIDC-IDRI-0921    1
LIDC-IDRI-0523    1
LIDC-IDRI-0432    1
LIDC-IDRI-0493    1
Name: count, Length: 1010, dtype: int64
```

Existem 1010 entradas, o que seria de esperar porque temos 1010 pacientes. No entanto, alguns pacientes tem mais que uma entrada, a que se deve esse facto?

Vamos ver o caso de 'LIDC-IDRI-0132' e de 'LIDC-IDRI-0151' que apresentam ambos 3 entradas.

In [37]:
```
df3_metadata[df3_metadata['Subject ID'] == 'LIDC-IDRI-0132']
```

Out[37]:

	Subject ID	Study UID	Study Description	Study Date	
233	LIDC-IDRI-0132	1.3.6.1.4.1.14519.5.2.1.6279.6001.218658642102...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
683	LIDC-IDRI-0132	1.3.6.1.4.1.14519.5.2.1.6279.6001.314138616411...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
1109	LIDC-IDRI-0132	1.3.6.1.4.1.14519.5.2.1.6279.6001.300027087262...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519

In [38]:
```
df3_metadata[df3_metadata['Subject ID'] == 'LIDC-IDRI-0151']
```

Out[38]:

	Subject ID	Study UID	Study Description	Study Date	
1076	LIDC-IDRI-0151	1.3.6.1.4.1.14519.5.2.1.6279.6001.222444895275...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
1110	LIDC-IDRI-0151	1.3.6.1.4.1.14519.5.2.1.6279.6001.280420083608...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
1210	LIDC-IDRI-0151	1.3.6.1.4.1.14519.5.2.1.6279.6001.780121538270...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519

Concluimos que quando se trata do mesmo sujeito, apesar de este puder ter mais que uma [illegible]tradas podem ser bastante diferentes entre si. Variando dados como o Study UID, a Modality, o Manufacturer, entre outros.

Vamos observar em maior detalhe o Study UID:

In [39]:
```
df3_metadata['Study UID'].value_counts()
```

Out[39]:
```
Study UID
1.3.6.1.4.1.14519.5.2.1.6279.6001.281499745765120562304307889347    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.127299727722058129007613721740    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.190036064555105762519342089670    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.120580422725898456426920474122    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.238406803970631866767433328949    1
                                                                    ..
1.3.6.1.4.1.14519.5.2.1.6279.6001.327077273262887321600634232119    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.147733809306552412615241520287    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.165928214371593461789742284562    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.305863253247137744276642948253    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.333362756208643390458434024958    1
Name: count, Length: 1308, dtype: int64
```

In [40]:
```
df3_metadata['Study UID'].nunique()
```

Out[40]:
```
1308
```

Sabemos que existem 1308 entradas neste dataset e que na coluna 'Study UID' todos os seus valores são diferentes.

Será que o mesmo acontece para o Series ID?

In [41]:
```
df3_metadata['Series ID'].value_counts()
```

Out[41]:
```
Series ID
1.3.6.1.4.1.14519.5.2.1.6279.6001.100225287222365663678666836860    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.300392272203629213913702120739    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.305000574633573438189782624063    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.304700823314998198591652152637    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.304676828064484590312919543151    1
                                                                    ..
1.3.6.1.4.1.14519.5.2.1.6279.6001.199916291608127150948127902598    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.199670099218798685977406484591    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.199282854229880908602362094937    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.199261544234308780356714831537    1
1.3.6.1.4.1.14519.5.2.1.6279.6001.997611074084993415992563148335    1
Name: count, Length: 1308, dtype: int64
```

In [42]:
```
df3_metadata['Series ID'].nunique()
```

Out[42]:
```
1308
```

Concluimos que também não apresenta dados repetidos. O que não nos oferece grandes informações, uma vez que não é possivel fazer agrupamento de dados.

Mas será que há outras colunas que nos permitam agrupar os dados?

In [43]:
```
df3_metadata['Collection Name'].nunique()
```

Out[43]:
```
1
```

Ao que parece a coleção de onde os dados foram extraidos também não nos permite fazer

los.

In [44]:
```
df3_metadata['Modality'].nunique()
```

Out[44]:
```
3
```

In [45]:
```
df3_metadata['Modality'].value_counts()
```

Out[45]:
```
Modality
CT    1018
DX     237
CR      53
Name: count, dtype: int64
```

Mas a Modality já permite. Como podemos ver aqui temos informações sobre o modo utilizado para obter as imagens. Uma vez que o objetivo deste trabalho é utilizar CT scans, todos os dados obtidos atraves de outros meio não são relevantes, como tal podem ser eliminados.

In [46]:
```
df3_metadata = df3_metadata[df3_metadata['Modality'] == 'CT']
```

Agora que apenas temos dados obtidos através de CT scans, deparamo-nos com outro problema: há pacientes que tem mais que uma entrada, pois há 1018 entradas com CT scans. Será que os pacientes que estão aqui repetidos são os mesmos que estavam em NoduleCountsByPatient?

In [47]:
```
df3_metadata[df3_metadata['Subject ID'].duplicated(keep=False)]
```

Out[47]:

	Subject ID	Study UID	Study Description	Study Date	
136	LIDC-IDRI-0484	1.3.6.1.4.1.14519.5.2.1.6279.6001.202012743926...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
206	LIDC-IDRI-0484	1.3.6.1.4.1.14519.5.2.1.6279.6001.292226263060...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
218	LIDC-IDRI-0355	1.3.6.1.4.1.14519.5.2.1.6279.6001.246541809203...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
233	LIDC-IDRI-0132	1.3.6.1.4.1.14519.5.2.1.6279.6001.218658642102...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
266	LIDC-IDRI-0332	1.3.6.1.4.1.14519.5.2.1.6279.6001.299122687641...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
306	LIDC-IDRI-0355	1.3.6.1.4.1.14519.5.2.1.6279.6001.221138977753...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
478	LIDC-IDRI-0365	1.3.6.1.4.1.14519.5.2.1.6279.6001.212341120080...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
557	LIDC-IDRI-0442	1.3.6.1.4.1.14519.5.2.1.6279.6001.193438875802...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
683	LIDC-IDRI-0132	1.3.6.1.4.1.14519.5.2.1.6279.6001.314138616411...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
934	LIDC-IDRI-0332	1.3.6.1.4.1.14519.5.2.1.6279.6001.124508094927...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
947	LIDC-IDRI-0315	1.3.6.1.4.1.14519.5.2.1.6279.6001.709924281509...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
1001	LIDC-IDRI-0442	1.3.6.1.4.1.14519.5.2.1.6279.6001.265657827498...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
1055	LIDC-IDRI-0315	1.3.6.1.4.1.14519.5.2.1.6279.6001.190722769298...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
1076	LIDC-IDRI-0151	1.3.6.1.4.1.14519.5.2.1.6279.6001.222444895275...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
1210	LIDC-IDRI-0151	1.3.6.1.4.1.14519.5.2.1.6279.6001.780121538270...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519
1227	LIDC-IDRI-0365	1.3.6.1.4.1.14519.5.2.1.6279.6001.216207548522...	NaN	2000-01-01 00:00:00.0	1.3.6.1.4.1.14519

Após analisar os pacientes repetidos em NoduleCountsByPatient, concluimos que os pacientes que se encontram repetidos aqui, são os mesmo que se encontram repetidos no outro dataset. A diferença mais significativa que se encontra aqui presente é o facto de nas entradas repetidas haver números de imagens distintos. O que nos coloca a questão de qual é a entrada correta.

Agora que já temos uma ideia do dataset resta-nos questionar se há valores NaN?

In [48]:
```
print(df3_metadata.isna().sum())
```

```
Subject ID              0
Study UID               0
Study Description     549
Study Date              0
Series ID               0
Series Description    706
Number of images        0
File Size (Bytes)       0
Collection Name         0
Modality                0
Manufacturer            0
dtype: int64
```

Concluimos que sim, mas em colunas redundantes, isto é, os valores em falta não são de interesse.

Análise de Dados Extraidos do Pylidc

[voltar ao Indice]

Este dataset foi gerado por nós e consiste na extração dos dados que achamos relevantes das anotações disponibilizadas dos diferentes radiologistas e alguns outros dados que achamos relevantes.

In [49]:
```
df4_pylidc = pd.read_csv('DadosPylidc.csv')
```

Aqui podemos ver quais os dados que extraimos das Annotations.

In [50]:
```
df4_pylidc.head()
```

Out[50]:

	Patient_ID	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobulation	Sphericity	Margin	Subtlet
0	LIDC-IDRI-0078	4	1.0	4	2.0	1.0	4.0	4.0	4.
1	LIDC-IDRI-0078	4	1.0	4	1.0	2.0	4.0	2.0	4.
2	LIDC-IDRI-0078	4	1.0	4	4.0	4.0	4.0	4.0	5.
3	LIDC-IDRI-0078	4	1.0	4	2.0	2.0	4.0	3.0	5.
4	LIDC-IDRI-0078	4	2.0	4	1.0	1.0	3.0	4.0	5.

Vejamos então se existem 1010 pacientes:

In [51]:
```
df4_pylidc['Patient_ID'].nunique()
```

Out[51]:
```
1010
```

Existem de facto 1010 pacientes, tal como seria suposto. Sabemos também que existem pacientes com mais que uma entrada pois extraimos os dados de cada nó e de cada anotação. Mas será que há pacientes que não tenham nódulos? De acordo com o dataset analisado anteriormente ("Nodules Counts By Patient"), há pacientes que não tem nódulos analisados.

In [52]:
```
df4_pylidc[df4_pylidc['Número de Nódulos'] == 0]
```

Out[52]:

	Patient_ID	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobulation	Sphericity	Margin	Sub
313	LIDC-IDRI-0028	0	NaN	0	NaN	NaN	NaN	NaN	
333	LIDC-IDRI-0032	0	NaN	0	NaN	NaN	NaN	NaN	
652	LIDC-IDRI-0062	0	NaN	0	NaN	NaN	NaN	NaN	
727	LIDC-IDRI-0071	0	NaN	0	NaN	NaN	NaN	NaN	
895	LIDC-IDRI-0100	0	NaN	0	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	
6762	LIDC-IDRI-0668	0	NaN	0	NaN	NaN	NaN	NaN	
6763	LIDC-IDRI-0667	0	NaN	0	NaN	NaN	NaN	NaN	
6770	LIDC-IDRI-0665	0	NaN	0	NaN	NaN	NaN	NaN	
6863	LIDC-IDRI-0653	0	NaN	0	NaN	NaN	NaN	NaN	
6927	LIDC-IDRI-0646	0	NaN	0	NaN	NaN	NaN	NaN	

135 rows × 18 columns

Como percebemos há de facto pacientes que não tem nódulos analisados e como tal para este trabalho, eles não são relevantes.

Será que houve algum agrupamento com erro? Isto é, que apresentava mais de 4 anotações, o que não vai de acordo com o que deveria acontecer e casos em que o paciente não apresenta nódulos, pois só há anotações para aqueles pacientes que tinham nódulos classificados com diametro >= 3mm.

In [53]:
```
df4_pylidc[df4_pylidc['Agrupamento com Erro'] == 'Sim']
```

Out[53]:

	Patient_ID	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobulation	Sphericity	Margin	Sub
313	LIDC-IDRI-0028	0	NaN	0	NaN	NaN	NaN	NaN	
333	LIDC-IDRI-0032	0	NaN	0	NaN	NaN	NaN	NaN	
573	LIDC-IDRI-0055	7	1.0	6	1.0	1.0	4.0	5.0	
574	LIDC-IDRI-0055	7	1.0	6	2.0	2.0	3.0	3.0	
575	LIDC-IDRI-0055	7	1.0	6	4.0	3.0	3.0	3.0	
...	...	...	...	...	...	...	...	...	
6762	LIDC-IDRI-0668	0	NaN	0	NaN	NaN	NaN	NaN	
6763	LIDC-IDRI-0667	0	NaN	0	NaN	NaN	NaN	NaN	
6770	LIDC-IDRI-0665	0	NaN	0	NaN	NaN	NaN	NaN	
6863	LIDC-IDRI-0653	0	NaN	0	NaN	NaN	NaN	NaN	
6927	LIDC-IDRI-0646	0	NaN	0	NaN	NaN	NaN	NaN	

216 rows × 18 columns

Concluimos que de facto existem casos assim.

Será que existem pacientes com Study UID distintos como acontecia na metadata?

In [54]:
```
study_count = df4_pylidc.groupby('Patient_ID')['Study Instance UID'].nunique()

# Filtra pacientes que têm mais de um Study Instance UID
study_count[study_count > 1]
```

Out[54]:
```
Patient_ID
LIDC-IDRI-0132    2
LIDC-IDRI-0151    2
LIDC-IDRI-0315    2
LIDC-IDRI-0332    2
LIDC-IDRI-0355    2
LIDC-IDRI-0365    2
LIDC-IDRI-0442    2
LIDC-IDRI-0484    2
Name: Study Instance UID, dtype: int64
```

Estes são exatamente os mesmo pacientes que tinham erro na metadata.

Será que há pacientes com números de nódulos distintos para o mesmo paciente, isto é, ue uma vez no dataset?

In [55]:
```
study_count = df4_pylidc.groupby('Patient_ID')['Número de Nódulos'].nunique()

# Filtra pacientes que têm mais de um Número de Nódulos
study_count[study_count > 1]
```

Out[55]:
```
Patient_ID
LIDC-IDRI-0132    2
LIDC-IDRI-0315    2
LIDC-IDRI-0332    2
LIDC-IDRI-0355    2
Name: Número de Nódulos, dtype: int64
```

Vamos terminar vendo os tipos de dados que temos:

In [56]:
```
df4_pylidc.dtypes
```

Out[56]:
```
Patient_ID                  object
Número de Nódulos            int64
Nódulo ID                  float64
Quantidade de Anotações      int64
Spiculation                float64
Lobulation                 float64
Sphericity                 float64
Margin                     float64
Subtlety                   float64
Texture                    float64
Calcification              float64
Malignancy                 float64
Internal Structure         float64
Agrupamento com Erro        object
Study Instance UID          object
Series Instance UID         object
Espessura das Fatias (mm)  float64
Resolução do Pixel         float64
dtype: object
```

Merge Datasets

[voltar a Indice]

Após analisarmos todos os datasets percebemos que há alguns com mais importância que outros, portanto quais é que vamos usar?

- **NoduleCountsByPatient:** Este dataset **vai ser usado** apesar de que ainda precisam de ser feitos alguns ajustes nos seus dados para corrigir alguns erros.
- **PatientDiagnoses:** Após analisarmos este dataset concluimos que **não vai ser utilizado**. Apesar de conter dados que podem ser considerados relevantes para classificar nódulos e a condição dos pacientes, há muitos dados em falta, sendo que apenas tem 157 pacientes e que alguns deles nem seriam utilizados por estarem classificados com 0 (unknown). Deste modo, iremos encontrar outra forma de classificar os nódulos.
- **Metadata:** Este dataset também **será utilizado** pois irá ajudar-nos a encontrar as ...ncias corretas entre os diferentes datasets.

- **DadosPylidc:** Este dataset é essencial, pois permite-nos encontrar erros nos restantes datasets, como tal, este dataset **será utilizado**.

O **objetivo** deste merge é sermos capazes de compreender quais são os pacientes que podem ser descartadados, seja por não terem nódulos para analisar ou por qualquer outro motivo que achemos relevantes e para garantir que os dados estão em concordância uns com os outros.

Primeiro Passo: Dar merge a NoduleCountsByPatient e a DadosPylidc.

In [57]:
```
merged_data = pd.merge(df1_Nodules, df4_pylidc, on='Patient_ID', how='inner')
```

In [58]:
```
merged_data.head()
```

Out[58]:

	Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobula
0	LIDC-IDRI-0001	4	1	3	1	1.0	4	4.0	
1	LIDC-IDRI-0001	4	1	3	1	1.0	4	5.0	
2	LIDC-IDRI-0001	4	1	3	1	1.0	4	3.0	
3	LIDC-IDRI-0001	4	1	3	1	1.0	4	5.0	
4	LIDC-IDRI-0002	12	1	11	1	1.0	2	1.0	

5 rows × 21 columns

Vamos apenas confirmar que temos todos os pacientes:

In [59]:
```
merged_data['Patient_ID'].nunique()
```

Out[59]:
```
1010
```

Agora que demos merge aos datasets vamos começar a tentar resolver os problemas que existiam:

- **Remover pacientes que tenham 0 nódulos em ambos os datasets:**

In [60]:
```
merged_data = merged_data[(merged_data['Número de Nódulos'] != 0) & (merged_data['N
```

Confirmar se há algum caso em que houvessem pacientes com zero numa das colunas e na outra não:

In [61]:
```
merged_data[merged_data['Número de Nódulos'] == 0]
```

Out[61]:

Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobulati

0 rows × 21 columns

In [62]:
```
merged_data[merged_data['Number of Nodules >=3mm**'] == 0]
```

Out[62]:

Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobulati

0 rows × 21 columns

In [63]:
```
merged_data['Patient_ID'].nunique()
```

Out[63]: 875

- **Remover todas as entradas que deram erro ao fazer o agrupamento:**

In [64]:
```
merged_data = merged_data[merged_data['Agrupamento com Erro'] != 'Sim']
```

Acabamos de remover todos aquele nódulos que tinham mais de 4 anotações, pois já removemos aqueles que tinham 0 nódulos. Deste modo garantimos que só há sempre até 4 anotações o que deveria ser o correto, uma vez que só há 4 radiologistas.

In [65]:
```
merged_data[merged_data['Agrupamento com Erro'] == 'Sim']
```

Out[65]:

Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobulati

0 rows × 21 columns

In [66]:
```
merged_data['Patient_ID'].nunique()
```

Out[66]: 870

- **Remover os pacientes que tem números de nódulos >= 3mm distintos nos datasets**

In [67]:
```
merged_data = merged_data[merged_data['Número de Nódulos'] == merged_data['Number c
```

Neste momento todas as entradas deste dataset estão de acordo uma com as outras no que diz respeito ao número de nódulos >= 3mm.

In [68]:
```
merged_data[merged_data['Número de Nódulos'] != merged_data['Number of Nodules >=3m
```

Out[68]:

Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobulati

0 rows × 21 columns

In [69]:
```
merged_data['Patient_ID'].nunique()
```

Out[69]:
```
855
```

- **Confirmar se há entradas com pacientes repetidos e com 'Total Number of Nodules' distintos**

In [70]:
```
study_count = merged_data.groupby('Patient_ID')['Total Number of Nodules*'].nunique

# Filtra pacientes que têm mais de um Study Instance UID
study_count[study_count > 1]
```

Out[70]:
```
Patient_ID
LIDC-IDRI-0132    2
LIDC-IDRI-0151    2
LIDC-IDRI-0315    2
LIDC-IDRI-0355    2
LIDC-IDRI-0365    2
LIDC-IDRI-0442    2
LIDC-IDRI-0484    2
Name: Total Number of Nodules*, dtype: int64
```

Como podemos ver os pacientes que tinham entradas duplicadas com valores distintos em NoduleCountsByPatient continuam aqui presentes, exceto o paciente 'LIDC-IDRI-0332'.

In [71]:
```
merged_data[merged_data['Patient_ID'] == 'LIDC-IDRI-0332']
```

Out[71]:

	Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lol
2507	LIDC-IDRI-0332	3	2	1	2	1.0	4	1.0	
2508	LIDC-IDRI-0332	3	2	1	2	1.0	4	2.0	
2509	LIDC-IDRI-0332	3	2	1	2	1.0	4	2.0	
2510	LIDC-IDRI-0332	3	2	1	2	1.0	4	4.0	
2511	LIDC-IDRI-0332	3	2	1	2	2.0	4	4.0	
2512	LIDC-IDRI-0332	3	2	1	2	2.0	4	4.0	
2513	LIDC-IDRI-0332	3	2	1	2	2.0	4	4.0	
2514	LIDC-IDRI-0332	3	2	1	2	2.0	4	2.0	

8 rows × 21 columns

In [72]:
```
df1_Nodules[df1_Nodules['Patient_ID'] == 'LIDC-IDRI-0332']
```

Out[72]:

	Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***
333	LIDC-IDRI-0332	6	5	1
334	LIDC-IDRI-0332	3	2	1

Como podemos ver uma das suas entradas foi eliminada quando se confirmou se o número de nódulos com diametro >= 3mm eram iguais em ambos os datasets.

O problema com o qual nos deparamos neste momento é que apesar de os dados que se encontram com 'Total Number of Nodules*' distintos tem o mesmo valor para nódulos >= 3mm, como não temos como confirmar qual das entradas é a correta, de modo a não termos problemas, iremos eliminar estes 7 pacientes.

In [73]:
```
# Lista de pacientes a serem removidos
pacientes_a_remover = ['LIDC-IDRI-0132', 'LIDC-IDRI-0151', 'LIDC-IDRI-0315',
                       'LIDC-IDRI-0355', 'LIDC-IDRI-0365', 'LIDC-IDRI-0442',
                       'LIDC-IDRI-0484']

# Remove os pacientes da lista
merged_data = merged_data[~merged_data['Patient_ID'].isin(pacientes_a_remover)]
```

Loading [MathJax]/extensions/Safe.js

até ao momento já só temos pacientes que tem nódulos >= 3mm de diametro e que os dados de NoduleCountsByPatient estão de acordo com os dados do

pylidc.

In [74]:
```
merged_data['Patient_ID'].nunique()
```

Out[74]:
848

Segundo Passo: Juntar a merged_data a metadata.

Para dar merge precisamos apenas de mudar o nome da coluna com os ID's dos pacientes em metadata.

In [75]:
```
df3_metadata = df3_metadata.rename(columns={'Subject ID': 'Patient_ID'})
```

Agora que ambos os datasets tem uma coluna com os mesmo dados e o nome em comum, podemos dar merge.

In [76]:
```
data = pd.merge(merged_data, df3_metadata, on='Patient_ID', how='inner')
```

In [77]:
```
data.head()
```

Out[77]:

	Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobula
0	LIDC-IDRI-0001	4	1	3	1	1.0	4	4.0	
1	LIDC-IDRI-0001	4	1	3	1	1.0	4	5.0	
2	LIDC-IDRI-0001	4	1	3	1	1.0	4	3.0	
3	LIDC-IDRI-0001	4	1	3	1	1.0	4	5.0	
4	LIDC-IDRI-0002	12	1	11	1	1.0	2	1.0	

5 rows × 31 columns

Vamos confirmar se continua a haver o mesmo número de pacientes:

In [78]:
```
data['Patient_ID'].nunique()
```

Out[78]:
848

- **Será que há dados com Study UID diferentes?**


```
y Instance UID'] != data['Study UID']]
```

Out[79]:

	Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lol
2226	LIDC-IDRI-0332	3	2	1	2	1.0	4	1.0	
2228	LIDC-IDRI-0332	3	2	1	2	1.0	4	2.0	
2230	LIDC-IDRI-0332	3	2	1	2	1.0	4	2.0	
2232	LIDC-IDRI-0332	3	2	1	2	1.0	4	4.0	
2234	LIDC-IDRI-0332	3	2	1	2	2.0	4	4.0	
2236	LIDC-IDRI-0332	3	2	1	2	2.0	4	4.0	
2238	LIDC-IDRI-0332	3	2	1	2	2.0	4	4.0	
2240	LIDC-IDRI-0332	3	2	1	2	2.0	4	2.0	

8 rows × 31 columns

Como conseguimos perceber o paciente que está a dar erro era aquele que aparecia em duplicado em NoduleCountsByPatient que na primeira parte desta fase do trabalho não deu erro, pois o número de nódulos que tinham >= 3mm eram iguais. No entanto, agora conseguimos perceber porque é que haviam várias entradas com dados distintos para o mesmo paciente, estava relacionado com o facto de pertencerem a estudos diferentes.

Vamos portanto eliminar os casos em que este erro acontece:

In [80]:
```
data = data[data['Study Instance UID'] == data['Study UID']]
```

In [81]:
```
data[data['Patient_ID'] == 'LIDC-IDRI-0332']
```

Out[81]:

	Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lol
2227	LIDC-IDRI-0332	3	2	1	2	1.0	4	1.0	
2229	LIDC-IDRI-0332	3	2	1	2	1.0	4	2.0	
2231	LIDC-IDRI-0332	3	2	1	2	1.0	4	2.0	
2233	LIDC-IDRI-0332	3	2	1	2	1.0	4	4.0	
2235	LIDC-IDRI-0332	3	2	1	2	2.0	4	4.0	
2237	LIDC-IDRI-0332	3	2	1	2	2.0	4	4.0	
2239	LIDC-IDRI-0332	3	2	1	2	2.0	4	4.0	
2241	LIDC-IDRI-0332	3	2	1	2	2.0	4	2.0	

8 rows × 31 columns

Apesar de se terem eliminado as entradas em que o Study UID era diferente, percebemos que o paciente 'LIDC-IDRI-0332' continua a ter dados válidos neste dataset.

- **Será que há dados com Series ID diferentes?**

In [82]:
```
data[data['Series Instance UID'] != data['Series ID']]
```

Out[82]:

	Patient_ID	Total Number of Nodules*	Number of Nodules >=3mm**	Number of Nodules <3mm***	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobulati

0 rows × 31 columns

Para as Series não há qualquer tipo de incompatibilidade.

- **Será que há pacientes com números de imagens diferentes?**

In [83]:
```
study_count = data.groupby('Patient_ID')['Number of images'].nunique()

# Filtra pacientes que têm mais de um Número de Imagens
study_count[study_count > 1]
```

Out[83]:
```
Series([], Name: Number of images, dtype: int64)
```

Concluimos que os dados estão corretos e limpos.

Terceiro Passo: Eliminar colunas repetidas ou desnecessárias.

Neste momento temos várias colunas que apresentam extamente a mesma informação e que não necessitam de estar repetidas, sendo essas:

- Number of Nodules >=3mm** == Número de Nódulos
- Study Instance UID == Study UID
- Series Instance UID == Series ID

E colunas que neste momento apresentam dados que já não são necessários, pois para o que se segue adiante não nos dão qualquer tipo de informação valiosa, sendo essas colunas:

- Total Number of Nodules*
- Number of Nodules <3mm***
- Agrupamento com Erro
- Study Description
- Study Date
- Series Description
- Number of images
- File Size (Bytes)
- Collection Name
- Modality
- Manufacturer

Assim sendo vamos remover todas as colunas sem interesse e deixar das repetidas apenas uma delas.

In [84]:
```
data = data.drop(['Total Number of Nodules*', 'Number of Nodules >=3mm**',
       'Number of Nodules <3mm***','Agrupamento com Erro', 'Study UID', 'Study Desc
       'Series ID', 'Series Description', 'Number of images',
       'File Size (Bytes)', 'Collection Name', 'Modality', 'Manufacturer'], axis =
```

Confirmar que as colunas foram removidas com sucesso:

In [85]:
```
data.head()
```

Out[85]:

	Patient_ID	Número de Nódulos	Nódulo ID	Quantidade de Anotações	Spiculation	Lobulation	Sphericity	Margin	Subtlet
0	LIDC-IDRI-0001	1	1.0	4	4.0	3.0	3.0	3.0	5.
1	LIDC-IDRI-0001	1	1.0	4	5.0	5.0	4.0	4.0	5.
2	LIDC-IDRI-0001	1	1.0	4	3.0	3.0	3.0	2.0	5.
3	LIDC-IDRI-0001	1	1.0	4	5.0	1.0	5.0	4.0	5.
4	LIDC-IDRI-0002	1	1.0	2	1.0	1.0	5.0	1.0	2.

Vamos apenas alterar o nome de duas colunas para que fiquem iguais aos dados originais:

In [86]:
```
data.rename(columns={
    'Study Instance UID': 'Study UID',
    'Series Instance UID': 'Series ID'
}, inplace=True)
```

Quarto Passo: Passar de floats para inteiros algumas colunas do pylidc

In [87]:
```
cols_to_convert = ['Nódulo ID', 'Spiculation', 'Lobulation', 'Sphericity', 'Margin'
data[cols_to_convert] = data[cols_to_convert].astype(int)
```

Ficamos então com o dataset pronto a ser utilizado. Iremos portanto convertê-o para um csv:

In [88]:
```
data.to_csv("merged_data.csv", encoding='utf-8', index=False)
```

A partir deste momento passamos a ter um dataset que nos garante que todos os pacientes e os seus nódulos aqui presentes estão dentro das condições que pertendemos analisar.

Junção das Anotações Semanticas e Label's

[voltar a Indice]

Até ao momento, não existem informações que nos permitam saber se um nódulo é cancerisno ou não, logo, como é que os podemos classificar como tal?

A base de dados LIDC-IDRI, na qual o pylidc se baseia, foi criada para fornecer imagens de tomografia computadorizada (TC) de pulmão e anotações feitas por radiologistas, mas ela não contém um diagnóstico final de malignidade confirmado por biópsia ou tratamento clínico. Em vez disso, a base de dados oferece informações sobre a probabilidade de malignidade estimada por radiologistas, sendo que a cada nódulo foi atruibuida uma pontuação de malignidade, sendo essa:

1: Altamente improvável de ser maligno.

2: Improvável de ser maligno.

3: Indeterminado.

4: Provável de ser maligno.

5: Altamente provável de ser maligno.

Estas pontuações são subjetivas e baseadas na experiência dos radiologistas, sendo úteis para avaliar a probabilidade de um nódulo ser maligno, apesar de não garantirem um diagnóstico definitivo. Por este motivo, de modo a termos uma classificação de nódulos o mais próxima do real possivel, decidimos fazer tal como é feito em vários estudos como [LLZ18]. Neste caso, o que estudo diz é que os dados serão divididos em 3 categorias:

- Malignancy > 3
- Malignancy == 3
- Malignancy < 3

De acordo com esta divisão, devemos usar apenas aqueles casos nos quais há pelo menos 3 radiologistas em concordância, isto é, tem de haver pelo menos 3 dos 4 radiologistas a escolher valores de malignidade dentro da mesma categoria, caso contrário o nódulo deve ser descartado.

In [89]:
```
df_merged = pd.read_csv("merged_data.csv")
```

In [90]:
```
contagem_valores = df_merged['Malignancy'].value_counts()
contagem_valores
```

Out[90]:
```
Malignancy
3    2481
2    1529
1     957
4     894
5     638
Name: count, dtype: int64
```

De modo a ir de acordo com o método que escolhemos para classificar um nódulo como maligno ou benigno, vamos começar por transformar os valores da coluna 'Malignancy', de modo a que os valores menores que 3, tenham valor 0 e os valores maiores que 3 tenham valor 1, isto é:

[4,5] = 1

[1,2] = 0

In [91]:
```
binary_values = {4: 1, 5: 1, 1: 0, 2: 0}
df_merged['Malignancy'] = df_merged['Malignancy'].map(binary_values)
```

Neste momento já temos os valores da Malignancy prontos para que possa ser utilizada a moda, no entanto, se queremos agrupar as anotações de modo a ter apenas uma classificação por nódulo, precisamos de fazer o mesmo às restantes colunas, para isso

iremos primeiro criar um novo dataset, com apenas aqueles dados que nos interessam, usando a moda para classificar cada nódulo:

In [92]:
```
# Definir as colunas de identificação e as semânticas
identification_columns = ['Número de Nódulos', 'Quantidade de Anotações']
semantic_columns = ['Spiculation', 'Lobulation', 'Sphericity', 'Margin', 'Subtlety'
                    'Texture', 'Calcification', 'Internal Structure']

# Agrupar por 'Patient_ID' e 'Nódulo ID' e pegar a primeira entrada dos dados de id
df_identification = df_merged.groupby(['Patient_ID', 'Nódulo ID'])[identification_c

# Calcular a moda para as colunas semânticas agrupando por 'Patient_ID' e 'Nódulo I
df_semantic_mode = df_merged.groupby(['Patient_ID', 'Nódulo ID'])[semantic_columns]

# Juntar os dados de identificação com os dados das colunas semânticas
df_final = df_identification.join(df_semantic_mode)
```

Ficamos então com um dataset pronto para ser classificado de acordo com o critério defenido previamente:

In [93]:
```
# Função para aplicar o critério de malignidade
def malignancy_criteria(group):
    count_ones = (group == 1).sum()
    count_zeros = (group == 0).sum()
    count_threes = (group == 3).sum()

    if count_ones >= 3:
        return 1
    elif count_zeros >= 3:
        return 0
    else:
        return 3  # Manter os indeterminados no critério
```

In [94]:
```
# Aplicar o critério de malignidade por 'Patient_ID' e 'Nódulo ID'
df_malignancy = df_merged.groupby(['Patient_ID', 'Nódulo ID'])['Malignancy'].agg(ma
```

In [95]:
```
# Juntar a coluna 'Malignancy' no dataframe final
df_final = df_final.join(df_malignancy, on=['Patient_ID', 'Nódulo ID'])
```

Ficamos, portanto, com apenas uma classificação por nódulo, contudo, levanta-se uma questão: devemos usar aqueles nódulos que foram classificados como indeterminados ou não?

Usando AUC, accuracy, sensitivity e specificity, o estudo XZX+18, mostrou que é mais vantajoso eliminar todos aqueles nódulos que são classificados como indeterminados (3), uma vez que o dataset que exclui completamente esses nódulos foi o que obteve melhores resuldados. Desta forma, decidimos eliminar completamente os nódulos que tenham tido essa classificação.

Removemos, portanto, todas as entradas cujos nódulos que tenham sido classificados como indeterminados(3):

In [96]:
```
# Remover os casos indeterminados (3)
df_final_cleaned = df_final[df_final['Malignancy'] != 3]
```

O dataset está quase pronto a ser utilizado. Resta apenas descobrir se ao fazer a junção das anotações, alguma das colunas ficou com valores únicos, pois caso isso tenha acontecido a feature deixa de ser relevante, pois em nada contribui para a melhor classificação dos nódulos.

In [97]:
```
df_final_cleaned.nunique()
```

Out[97]:
```
Número de Nódulos          16
Quantidade de Anotações     2
Spiculation                 5
Lobulation                  5
Sphericity                  5
Margin                      5
Subtlety                    5
Texture                     5
Calcification               5
Internal Structure          1
Malignancy                  2
dtype: int64
```

Como podemos ver a coluna 'Internal Structure' apresenta um único valor e como tal deixa de ser relevante.

In [98]:
```
df_final_cleaned = df_final_cleaned.drop(columns=['Internal Structure'])
```

Vamos ver quantos nódulos temos classificados como malignos e benignos:

In [99]:
```
contagem_valores = df_final_cleaned['Malignancy'].value_counts()
contagem_valores
```

Out[99]:
```
Malignancy
1    205
0    204
Name: count, dtype: int64
```

Conseguimos verificar que as classes estão bem balenciadas.

Agora sim podemos guardar o dataset.

In [100…
```
# Guardar o novo dataset
df_final_cleaned.to_csv("data.csv", encoding='utf-8', index=True)
```

Pré-Processamento Imagens

[voltar ao índice]

Neste momento, já fizemos uma análise e tratamento dos dados que dizem respeito a quais pacientes devem ser utilizados neste projeto. Tendo isto em conta, devemos agora preparar os dados desses pacientes para que possamos extrair de modo mais eficaz as features usando o radiomics.

O pré-processamento de imagens é uma etapa essencial em qualquer projeto de visão computacional, especialmente quando se trata de imagens médicas como as de tomografia (CT), devido à complexidade e variabilidade desses dados.

Apesar de colocarmos aqui as funções que iremos utilizar para o pré-processamento das imagens decidimos que apenas as iremos utilizar aquando da extração das features com o pyradiomics.

Hounsfield Units (HU)

A Hounsfield Unit é uma escala adimensional utilizada em tomografia computarizada para expressar valores de forma padronizada. Os valores são obtidos a partir de transformações lineares a partir dos coeficientes de atenuação medidos (quantificam a perda de intensidade dos raios X). Pode ser usada para fornecer informações importantes sobre densidade e composição dos nódulos.

In [101…
```python
# Função para converter valores de pixel para Hounsfield Units (HU)
def convert_to_hu(image):
    # Aceder a Rescale Intercept e Rescale Slope
    intercept = -1024
    slope = 1

    # Aplicar a fórmula de conversão
    hu_image = image * slope + intercept
    hu_image = np.clip(hu_image, -1000, 400)  # Limitar a faixa de HU
    return hu_image
```

Normalização

A normalização permite ajustar a escala dos valores (como os Hounsfield Units), colocando-os num intervalo uniforme, o que é essencial para processá-las em lotes uniformes e reduzir a complexidade computacional.

In [102…
```python
# Função de pré-processamento: normalização
def norm(hu_image):
    # Normalização dos valores HU para o intervalo [0, 1]
    normalized_image = (hu_image - np.min(hu_image)) / (np.max(hu_image) - np.min(h
    return normalized_image
```

Segmentação - 2d

A segmentação de imagens é o processo de dividir uma imagem em partes ou regiões significativas, com o objetivo de identificar e isolar objetos ou estruturas específicas dentro da imagem. Em termos simples, segmentar uma imagem significa separar ou destacar áreas de interesse para análise, ignorando as demais áreas que não são relevantes para o estudo. Deste modo, a segmentação facilita a análise e interpretação das imagens, tal como a extração de features. Para a segmentação, utilizamos única e exclusivamente um slice, sendo este obtido a partir do centroide do nódulo, tal como é feito em alguns estudos como [CBMLN].

In [103…
```python
# Esta função devolve a imagem e a máscara a serem usadas num determinado nó.
def segmentation_2d(scan, annotation):

    volume = scan.to_volume()

    # Obter o centro do nódulo (i, j, k) - k é a slice correspondente ao centro
    nnotation.centroid
    slice_idx = int(k)  # Convertendo k para índice da slice
```

```
    # Gerar a máscara binária para a slice correspondente
    mascara_binaria = np.zeros(volume[:, :, slice_idx].shape, dtype=np.uint8)

    for contour in annotation.contours:
        if contour.image_k_position == slice_idx:
            # Obter as coordenadas do contorno e convertê-las em 2D
            ii, jj = contour.to_matrix(include_k=False).T
            mascara_binaria[ii.astype(int), jj.astype(int)] = 1

    # Aplicar a máscara na imagem original da slice
    imagem_slice = volume[:, :, slice_idx]

    return imagem_slice, mascara_binaria
```

Segmentação - 3d

Tal como a segmentação em 2d, a segmentação em 3d permite identificar objetos ou regiões de interesse, mas em vez de ser numa imagem, esta aplica-se a imagens tridimensionais (3D), que geralmente são obtidas a partir de várias fatias de CT scans, por exemplo. Essas imagens possuem informações em três dimensões (x, y e z), resultando em uma representação volumétrica dos objetos.

In [104…

```
# Esta função devolve o volume 3d e a máscara 3d do nódulo.
def segmentation_3d(scan, annotation):

    # Obter o volume completo do scan em HU
    volume = scan.to_volume()

    # Obter os limites de cada fatia do contorno
    mascara_binaria_3d = np.zeros(volume.shape, dtype=np.uint8)  # Máscara 3D para

    # Iterar sobre os contornos e preencher a máscara 3D
    for contour in annotation.contours:
        slice_idx = np.argmin(np.abs(scan.slice_zvals - contour.image_z_position))

        # Obter as coordenadas do contorno e convertê-las em 2D
        ii, jj = contour.to_matrix(include_k=False).T  # Coordenadas do contorno em
        mascara_binaria_3d[ii.astype(int), jj.astype(int), slice_idx] = 1  # Preenc

    # O volume HU já está pronto para ser retornado
    return volume, mascara_binaria_3d
```

Análise Exploratória dos Dados

[voltar ao índice]

Tal como mencionamos previamente, decidimos que o pré-processamento das imagens será feito imediatamente antes da extração das features com auxilio ao pyradiomics. Contudo, de modo a compreender o que cada uma daquelas funções irá realmente fazer, vamos fazer uma pequena análise:

Começamos por escolher um paciente específico:

```
# Exemplo de teste com o scan do pylidc
scans = pl.query(pl.Scan).filter(pl.Scan.patient_id == 'LIDC-IDRI-0001').all()
```

```
scan = pl.query(pl.Scan).filter(pl.Scan.patient_id == 'LIDC-IDRI-0001').first()
print(scan)
```

```
Scan(id=12,patient_id=LIDC-IDRI-0001)
```

In [106…

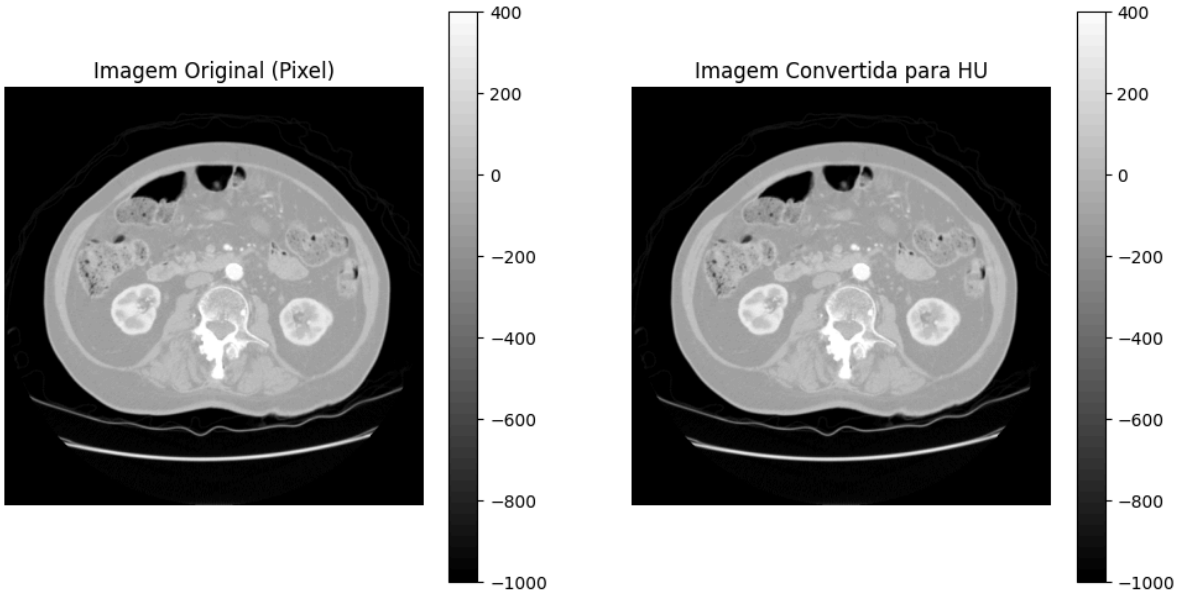
```
# Carregar a primeira fatia DICOM (image)
image = scan.load_all_dicom_images()[0]
```

```
Loading dicom files ... This may take a moment.
```

Conversão para HU

In [107…

```
# Obter a imagem original (valores de pixel)
pixel_image = image.pixel_array

# Testar a função de conversão para HU
hu_image = convert_to_hu(image.pixel_array)

# Visualizar as duas imagens lado a lado
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

# Imagem original (pixels)
axes[0].imshow(pixel_image, cmap='gray')
axes[0].set_title('Imagem Original (Pixel)')

# Imagem convertida para HU
axes[1].imshow(hu_image, cmap='gray')
axes[1].set_title('Imagem Convertida para HU')

# Mostrar as barras de cor
for ax in axes:
    ax.axis('off')
    plt.colorbar(ax.imshow(hu_image, cmap='gray'), ax=ax)

plt.show()
```



Na maioria dos casos, não há uma diferença visual muito evidente entre a imagem original (valores de pixel brutos) e a imagem convertida para Unidades Hounsfield (HU), quando ambas são exibidas em escala de cinza. Contudo, a diferença mais importante está no array de pixels em si, que agora tem valores que representam as densidades de diferentes materiais em termos de HU, que é a escala padrão usada na prática médica, como podemos

In [108…
```
# Imagem original
print(pixel_image)
```

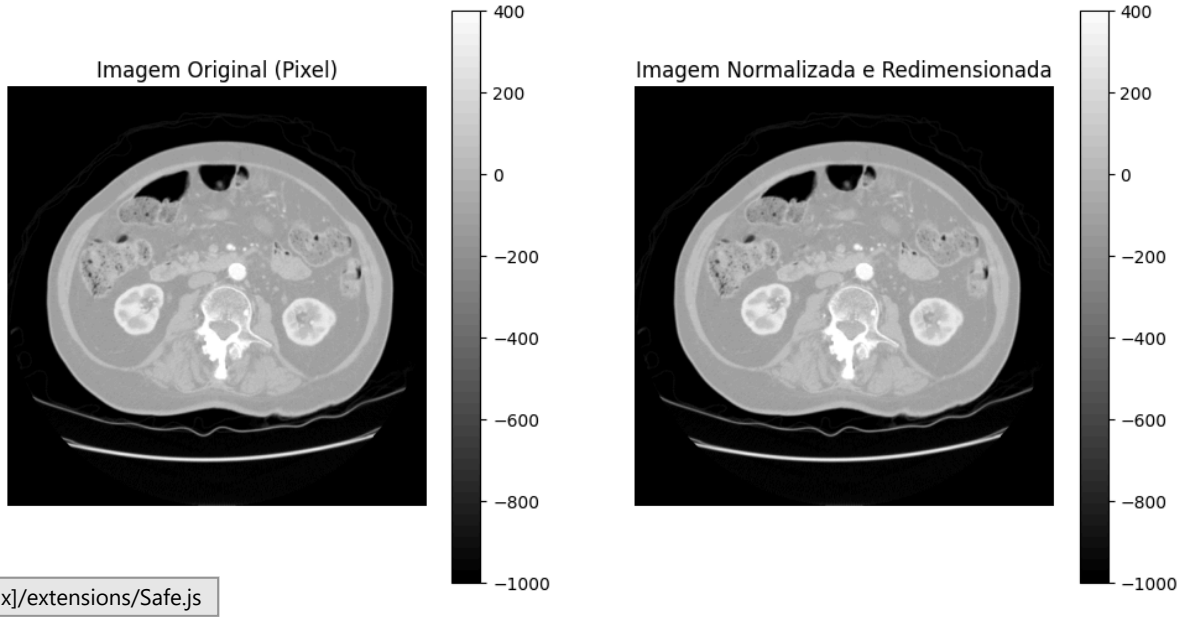
```
[[-1024 -1024 -1024 ... -1024 -1024 -1024]
 [-1024 -1024 -1024 ... -1024 -1024 -1024]
 [-1024 -1024 -1024 ... -1024 -1024 -1024]
 ...
 [-1024 -1024 -1024 ... -1024 -1024 -1024]
 [-1024 -1024 -1024 ... -1024 -1024 -1024]
 [-1024 -1024 -1024 ... -1024 -1024 -1024]]
```

In [109…
```
# Imagem convertida
print(hu_image)
```

```
[[-1000 -1000 -1000 ... -1000 -1000 -1000]
 [-1000 -1000 -1000 ... -1000 -1000 -1000]
 [-1000 -1000 -1000 ... -1000 -1000 -1000]
 ...
 [-1000 -1000 -1000 ... -1000 -1000 -1000]
 [-1000 -1000 -1000 ... -1000 -1000 -1000]
 [-1000 -1000 -1000 ... -1000 -1000 -1000]]
```

Normalizar

In [110…
```
normalized_image = norm(hu_image)

# Visualizar as duas imagens lado a lado
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

# Imagem original (pixels)
axes[0].imshow(pixel_image, cmap='gray')
axes[0].set_title('Imagem Original (Pixel)')

# Imagem convertida para HU
axes[1].imshow(normalized_image, cmap='gray')
axes[1].set_title('Imagem Normalizada e Redimensionada')

# Mostrar as barras de cor
for ax in axes:
    ax.axis('off')
    plt.colorbar(ax.imshow(hu_image, cmap='gray'), ax=ax)

plt.show()
```



Observadas assim lado a lado, também não vemos grandes diferenças, mas isso tem a haver com o modo como o matplotlib mostra as imagens, pois se analisarmos as caracteristicas das imagens é possivel ver as diferenças:

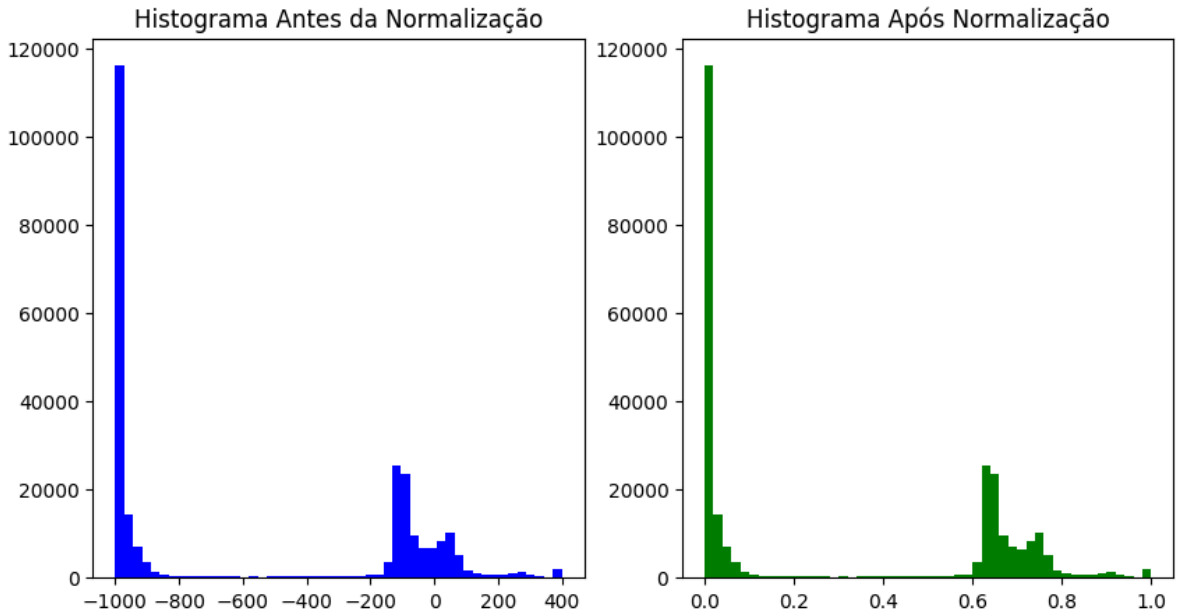
- PIXEL_ARRAY

In [111…
```
print(normalized_image)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

Como não é possivel ver o array completo, podemos fazer um histograma que mostre as intensidades dos pixel's antes e depois da normalização:

In [112…
```
# Plotar o histograma dos valores de pixel antes e depois
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.hist(hu_image.flatten(), bins=50, color='blue')
plt.title('Histograma Antes da Normalização')

plt.subplot(1, 2, 2)
plt.hist(normalized_image.flatten(), bins=50, color='green')
plt.title('Histograma Após Normalização')

plt.show()
```



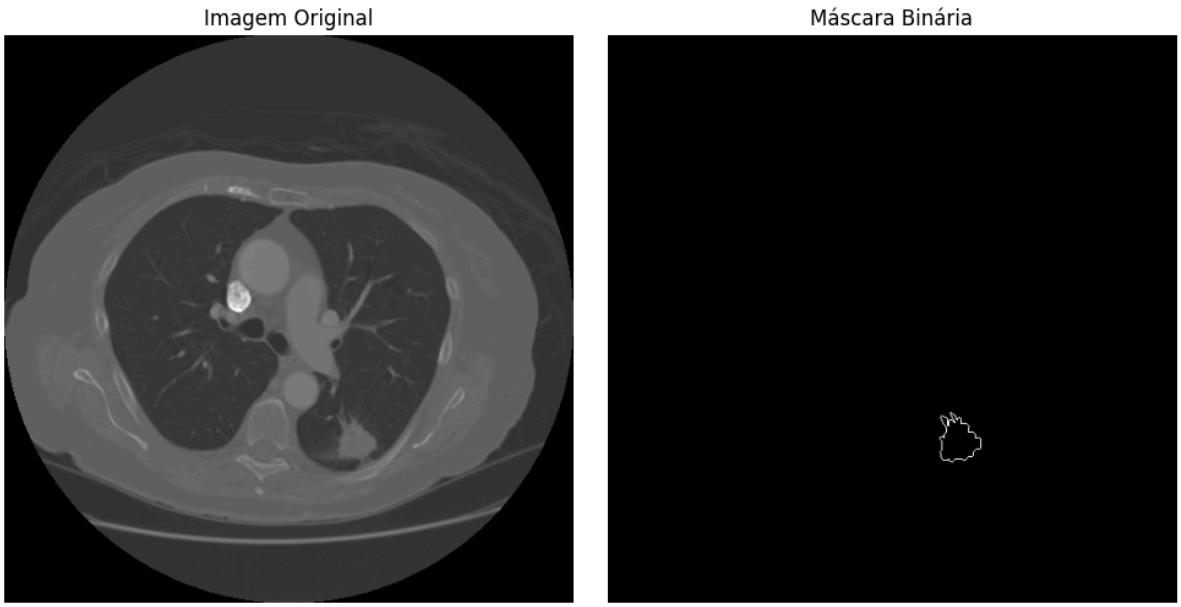
Como podemos ver, de facto, existem diferenças nas escalas.

Segmentação 2D

In [113…
```
scan = pl.query(pl.Scan).filter(pl.Scan.patient_id == 'LIDC-IDRI-0001').all()[0]
                                                an.annotations[0]
# Chamar a função para processar cada nódulo
```

```
imagem, mascara = segmentation_2d(scan, annotation)

# Criar um subplot para exibir as duas imagens lado a lado
fig, axes = plt.subplots(1, 2, figsize=(10, 5))

# Mostrar a imagem original no primeiro eixo
axes[0].imshow(imagem, cmap='gray')
axes[0].set_title('Imagem Original')
axes[0].axis('off')  # Esconder os eixos

# Mostrar a máscara binária no segundo eixo
axes[1].imshow(mascara, cmap='gray')
axes[1].set_title('Máscara Binária')
axes[1].axis('off')  # Esconder os eixos

# Mostrar a figura com as duas imagens
plt.tight_layout()
plt.show()
```

```
Loading dicom files ... This may take a moment.
```



Como podemos ver o segmentation em 2d está a funcionar corretamente. Neste caso o que fizemos foi dar à função as anotações relativas a um nódulo e deixamos que a função escolha o slice que passa pelo centroide, devolvendo-nos a imagem e a respetiva máscara binária, como podemos observar.

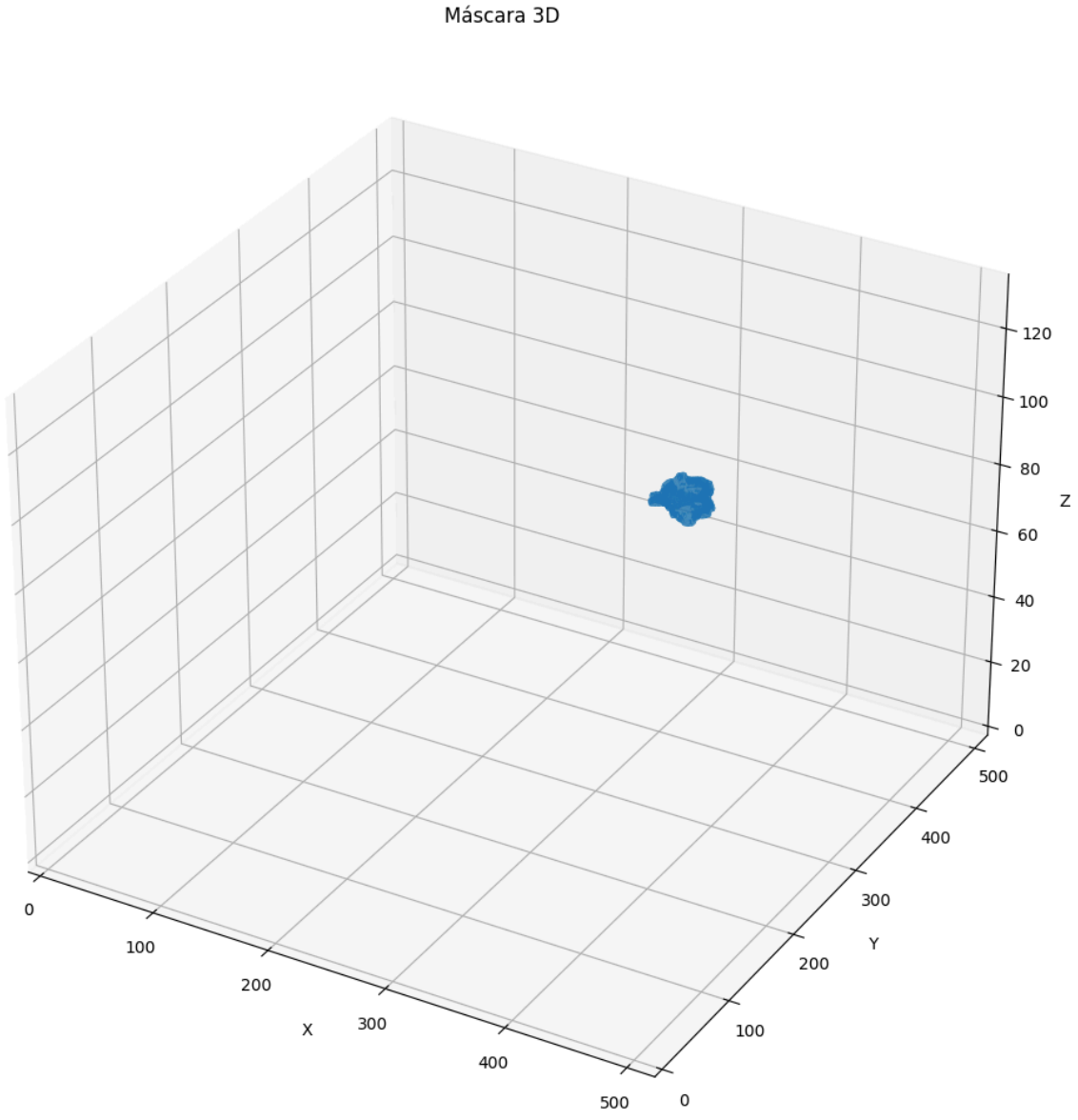
Segmentação 3D

In [114…

```
scans = pl.query(pl.Scan).filter(pl.Scan.patient_id == 'LIDC-IDRI-0001').all()[0]
annotations = scan.annotations
_, mascara_3d = segmentation_3d(scan, annotation)


# Encontrar a superfície da máscara binária usando marching cubes
verts, faces, _, _ = measure.marching_cubes(mascara_3d, level=0)

# Criar a figura e o eixo 3D
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')

# Renderizar a superfície 3D da máscara
llection(verts[faces], alpha=0.7)
ax.add_collection3d(mesh)
```

```
# Definir os limites do gráfico
ax.set_xlim(0, mascara_3d.shape[0])
ax.set_ylim(0, mascara_3d.shape[1])
ax.set_zlim(0, mascara_3d.shape[2])

# Ajustar a visualização
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_title("Máscara 3D")

plt.tight_layout()
plt.show()
```

```
Loading dicom files ... This may take a moment.
```



Como podemos ver a função de segmentação em 3d também está a funcionar corretamente, tal como é possivel concluir através da máscara binária criada.

Extração de Features - Pyradiomics

IR BUSCAR OS PACIENTES E OS NÓDULOS A ANALISAR

Agora podemos começar a extração das features através do pyradiomics, contudo, como já percebemos anteriormente, nem todos os nódulos eram de interesse para este estudo. Uma vez que já sabemos quais são os nódulos de interesse, iremos apenas realizar a extração das features desses nódulos. Deste modo, não iremos perder tempo com nódulos irrelevantes.

In [115…
```
df= pd.read_csv("data.csv")
df['Nódulo ID'] = df['Nódulo ID'].astype(int)
```

In [116…
```
patient_dict = df.groupby('Patient_ID')['Nódulo ID'].apply(lambda x: list(set(x))).
```

Extração de Features -> 2D

Vamos configurar as features que queremos extrair das imagens 2D, para evitar que descarreguemos features descenessárias:

In [117…
```
extractor_2d = featureextractor.RadiomicsFeatureExtractor()

extractor_2d.disableAllFeatures()

extractor_2d.enableFeatureClassByName('firstorder')
extractor_2d.enableFeatureClassByName('glcm')
extractor_2d.enableFeatureClassByName('glrlm')
extractor_2d.enableFeatureClassByName('glszm')
extractor_2d.enableFeatureClassByName('gldm')
extractor_2d.enableFeatureClassByName('ngtdm')
extractor_2d.enableFeatureClassByName('shape2D')

feature_list = []
```

Agora podemos começar a extração dos dados que serão guardados num CSV.

In [118…
```
def processar_pacientes(patient_dict, extractor):
    feature_list = []

    for patient_id in patient_dict:
        print(patient_id)
        # Obter os scans por paciente
        scans = pl.query(pl.Scan).filter(pl.Scan.patient_id == patient_id).all()

        for _, scan in enumerate(scans):
            # Obter todas as anotações (nódulos) para esse scan
            annotations = scan.annotations

            # Iterar dentro dos nódulos
            for nodule_idx, annotation in enumerate(annotations):
                if nodule_idx + 1 in patient_dict[patient_id]:
                    # Chamar a função para processar cada nódulo
                    imagem, mascara = segmentation_2d(scan, annotation)

                    # Converter a imagem para unidades HU
                    imagem_hu = convert_to_hu(imagem)

                    # Converter a imagem e a máscara para SimpleITK
                    imagem_sitk = sitk.GetImageFromArray(imagem_hu)
                    mascara_sitk = sitk.GetImageFromArray(mascara.astype(np.uint8))

                    # Extrair as features radiômicas
```

```
                    features = extractor.execute(imagem_sitk, mascara_sitk)

                    feature_dict = dict(features)

                    # Adicionar o ID do paciente e o ID do nódulo ao dicionário de
                    feature_dict['Patient_ID'] = patient_id
                    feature_dict['Nodule_ID'] = nodule_idx + 1

                    # Adicionar o dicionário à lista de features
                    feature_list.append(feature_dict)

    # Converter a lista de dicionários em um DataFrame do pandas
    df_features = pd.DataFrame(feature_list)

    # Salvar o DataFrame em um arquivo CSV com codificação UTF-8
    df_features.to_csv('radiomics_features.csv', index=False, encoding='utf-8')

    return df_features

# Chamar a função principal para processar os pacientes e salvar o resultado no CSV
result_features = processar_pacientes(patient_dict, extractor_2d)
```

```
LIDC-IDRI-0001
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0003
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0004
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0007
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0011
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0013
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0015
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0016
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0018
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0021
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0023
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0031
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0034
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0037
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0039
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0043
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0044
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0045
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0046
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0047
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0049
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0051
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0052
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0053
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0054
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0057
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0058
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0059
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0061
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0066
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0067
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0068
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0072
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0073
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0075
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0077
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0078
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0080
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0081
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0082
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0085
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0087
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0088
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0089
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0094
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0095
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0106
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0108
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0111
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0116
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0118
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0119
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0120
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0121
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0124
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0126
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0130
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0142
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0144
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0148
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0156
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0160
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0163
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0168
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0169
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0170
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```



```
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0172
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0183
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0185
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0186
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0187
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0188
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0190
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0191
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0193
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0194
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0195
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```



```
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0196
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0203
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0206
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0211
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0212
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0213
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0215
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0217
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0221
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0222
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0240
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0242
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0244
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0248
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0249
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0250
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0256
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0262
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0263
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0265
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0267
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0272
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0276
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0281
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0284
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0286
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0287
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0294
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0305
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0309
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0313
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0321
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0325
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0332
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0337
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0346
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0347
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0348
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0351
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0353
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0354
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0359
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0362
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0368
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0375
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0376
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0377
                    iles ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0379
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0380
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0385
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0386
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0390
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0399
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0402
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0403
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0407
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0412
Loading dicom files ... This may take a moment.
```

Loading [MathJax]/extensions/Safe.js

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0415
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0419
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0421
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0423
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0426
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0427
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0433
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0436
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0437
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0440
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0443
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0447
Loading dicom files ... This may take a moment.
ical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0450
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0456
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0461
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0462
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0463
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0464
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0466
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0470
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0473
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0481
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0485
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0486
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0488
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0489
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0492
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0496
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0497
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0507
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0515
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0523
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0526
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0529
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0530
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0537
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0541
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0542
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0546
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0559
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0562
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0565
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0568
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0569
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0572
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0576
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0578
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0587
iles ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0590
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0602
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0604
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0605
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0606
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0610
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0614
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0615
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0617
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0619
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0620
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0625
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0639
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0640
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0641
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0642
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0645
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0648
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0649
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
Loading [MathJax]/extensions/Safe.js
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0655
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0660
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0663
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0671
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0674
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0678
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0686
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0697
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0698
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0702
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0704
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0708
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
ical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0709
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0713
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0715
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0717
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0722
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0726
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0727
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0732
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0740
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0748
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0749
Loading dicom files ... This may take a moment.
ical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
calculated
```

```
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0751
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0756
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0759
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0772
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0773
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0775
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0776
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0777
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0785
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0796
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0797
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0798
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0799
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0801
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0807
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0811
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0819
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0822
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0825
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0827
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0838
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0843
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0844
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0849
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0850
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0854
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0855
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0858
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0860
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0861
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0866
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0869
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0871
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0875
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0879
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0884
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0886
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0890
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0892
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0893
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0895
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0902
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0905
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0908
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0910
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0912
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0913
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0914
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0916
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0919
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0921
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0925
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0929
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0933
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0935
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0936
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0938
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0940
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0944
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0951
Loading dicom files ... This may take a moment.
ical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0956
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0961
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0963
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0965
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0966
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0968
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0976
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0980
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0987
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0993
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0994
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0997
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0999
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1002
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1004
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1007
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1011
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1012
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

Extração de Features -> 3D

Tal como configuramos as features que queremos extrair das imagens 2D, vamos agora fazer o mesmo para as imagens 3D.

In [119…

```
extractor_3d = featureextractor.RadiomicsFeatureExtractor()

extractor_3d.disableAllFeatures()

extractor_3d.enableFeatureClassByName('firstorder')
extractor_3d.enableFeatureClassByName('glcm')
extractor_3d.enableFeatureClassByName('glrlm')
extractor_3d.enableFeatureClassByName('glszm')
extractor_3d.enableFeatureClassByName('gldm')
extractor_3d.enableFeatureClassByName('ngtdm')
extractor_3d.enableFeatureClassByName('shape3D')
```

```
Feature class shape3D is not recognized
```

Agora podemos começar a extração dos dados que serão guardados num CSV.

```
def processar_pacientes_3D(patient_dict, extractor):
    feature_list = []
```

```
    for patient_id in patient_dict:
        print(patient_id)
        # Obter os scans por paciente
        scans = pl.query(pl.Scan).filter(pl.Scan.patient_id == patient_id).all()

        for _, scan in enumerate(scans):
            print(f"Processando paciente {patient_id}")

            # Obter todas as anotações (nódulos) para esse scan
            annotations = scan.annotations

            # Iterar dentro dos nódulos
            for nodule_idx, annotation in enumerate(annotations):
                if nodule_idx + 1 in patient_dict[patient_id]:
                    print(f"  Processando nódulo {nodule_idx + 1}/{len(annotations)

                    # Chamar a função para processar cada nódulo
                    imagem, mascara = segmentation_3d(scan, annotation)

                    # Converter a imagem para unidades HU
                    imagem_hu = convert_to_hu(imagem)

                    # Converter a imagem e a máscara para SimpleITK
                    imagem_sitk = sitk.GetImageFromArray(imagem_hu)
                    mascara_sitk = sitk.GetImageFromArray(mascara.astype(np.uint8))

                    # Extrair as features radiômicas
                    features = extractor.execute(imagem_sitk, mascara_sitk)

                    feature_dict = dict(features)

                    # Adicionar o ID do paciente e o ID do nódulo ao dicionário de
                    feature_dict['Patient_ID'] = patient_id
                    feature_dict['Nodule_ID'] = nodule_idx + 1

                    # Adicionar o dicionário à lista de features
                    feature_list.append(feature_dict)

    # Converter a lista de dicionários em um DataFrame do pandas
    df_features = pd.DataFrame(feature_list)

    # Salvar o DataFrame em um arquivo CSV com codificação UTF-8
    df_features.to_csv('radiomics_features_3d.csv', index=False, encoding='utf-8')

    return df_features

# Chamar a função principal para processar os pacientes e salvar o resultado no CSV
result_features = processar_pacientes_3D(patient_dict, extractor_3d)
```

```
LIDC-IDRI-0001
Processando paciente LIDC-IDRI-0001
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
calculated
LIDC-IDRI-0003
Processando paciente LIDC-IDRI-0003
  Processando nódulo 2/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
calculated
```

```
LIDC-IDRI-0004
Processando paciente LIDC-IDRI-0004
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0007
Processando paciente LIDC-IDRI-0007
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0011
Processando paciente LIDC-IDRI-0011
  Processando nódulo 6/23
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 10/23
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0013
Processando paciente LIDC-IDRI-0013
  Processando nódulo 3/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0015
Processando paciente LIDC-IDRI-0015
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0016
Processando paciente LIDC-IDRI-0016
  Processando nódulo 1/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 6/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0018
Processando paciente LIDC-IDRI-0018
  Processando nódulo 4/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0021
Processando paciente LIDC-IDRI-0021
  Processando nódulo 3/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0023
Processando paciente LIDC-IDRI-0023
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0031
Processando paciente LIDC-IDRI-0031
  Processando nódulo 3/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0034
Processando paciente LIDC-IDRI-0034
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0037
Processando paciente LIDC-IDRI-0037
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0039
Processando paciente LIDC-IDRI-0039
  Processando nódulo 1/20
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/20
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/20
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/20
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0043
Processando paciente LIDC-IDRI-0043
  Processando nódulo 3/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0044
Processando paciente LIDC-IDRI-0044
  Processando nódulo 2/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0045
Processando paciente LIDC-IDRI-0045
  Processando nódulo 3/33
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
  Processando nódulo 6/33
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 10/33
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0046
Processando paciente LIDC-IDRI-0046
  Processando nódulo 5/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0047
Processando paciente LIDC-IDRI-0047
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0049
Processando paciente LIDC-IDRI-0049
  Processando nódulo 12/39
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 15/39
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0051
Processando paciente LIDC-IDRI-0051
  Processando nódulo 2/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0052
Processando paciente LIDC-IDRI-0052
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0053
Processando paciente LIDC-IDRI-0053
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0054
Processando paciente LIDC-IDRI-0054
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0057
Processando paciente LIDC-IDRI-0057
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0058
Processando paciente LIDC-IDRI-0058
  Processando nódulo 1/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0059
Processando paciente LIDC-IDRI-0059
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0061
Processando paciente LIDC-IDRI-0061
  Processando nódulo 1/19
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0066
Processando paciente LIDC-IDRI-0066
  Processando nódulo 2/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0067
Processando paciente LIDC-IDRI-0067
  Processando nódulo 1/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0068
Processando paciente LIDC-IDRI-0068
  Processando nódulo 6/22
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0072
Processando paciente LIDC-IDRI-0072
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0073
Processando paciente LIDC-IDRI-0073
  Processando nódulo 3/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0075
Processando paciente LIDC-IDRI-0075
  Processando nódulo 2/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0077
Processando paciente LIDC-IDRI-0077
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0078
Processando paciente LIDC-IDRI-0078
  Processando nódulo 4/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0080
Processando paciente LIDC-IDRI-0080
  Processando nódulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0081
Processando paciente LIDC-IDRI-0081
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0082
Processando paciente LIDC-IDRI-0082
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0085
Processando paciente LIDC-IDRI-0085
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0087
Processando paciente LIDC-IDRI-0087
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0088
Processando paciente LIDC-IDRI-0088
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0089
Processando paciente LIDC-IDRI-0089
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0094
Processando paciente LIDC-IDRI-0094
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0095
Processando paciente LIDC-IDRI-0095
  Processando nódulo 1/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0106
Processando paciente LIDC-IDRI-0106
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0108
Processando paciente LIDC-IDRI-0108
  Processando nódulo 1/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0111
Processando paciente LIDC-IDRI-0111
  Processando nódulo 2/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0116
Processando paciente LIDC-IDRI-0116
  Processando nódulo 2/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0118
Processando paciente LIDC-IDRI-0118
  Processando nódulo 1/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 6/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0119
Processando paciente LIDC-IDRI-0119
  Processando nódulo 1/8
Loading [MathJax]/extensions/Safe.js iles ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando módulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0120
Processando paciente LIDC-IDRI-0120
  Processando módulo 4/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0121
Processando paciente LIDC-IDRI-0121
  Processando módulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0124
Processando paciente LIDC-IDRI-0124
  Processando módulo 5/35
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0126
Processando paciente LIDC-IDRI-0126
  Processando módulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0130
Processando paciente LIDC-IDRI-0130
  Processando módulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0142
Processando paciente LIDC-IDRI-0142
  Processando módulo 1/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando módulo 2/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0144
Processando paciente LIDC-IDRI-0144
  Processando módulo 1/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando módulo 4/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando módulo 5/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0148
Processando paciente LIDC-IDRI-0148
  Processando nódulo 2/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0156
Processando paciente LIDC-IDRI-0156
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0160
Processando paciente LIDC-IDRI-0160
  Processando nódulo 1/15
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/15
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0163
Processando paciente LIDC-IDRI-0163
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0168
Processando paciente LIDC-IDRI-0168
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0169
Processando paciente LIDC-IDRI-0169
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0170
Processando paciente LIDC-IDRI-0170
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0171
Processando paciente LIDC-IDRI-0171
  Processando nódulo 1/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0172
Processando paciente LIDC-IDRI-0172
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0183
Processando paciente LIDC-IDRI-0183
  Processando nódulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0185
Processando paciente LIDC-IDRI-0185
  Processando nódulo 3/20
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 6/20
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0186
Processando paciente LIDC-IDRI-0186
  Processando nódulo 5/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0187
Processando paciente LIDC-IDRI-0187
  Processando nódulo 4/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0188
Processando paciente LIDC-IDRI-0188
  Processando nódulo 4/24
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0190
Processando paciente LIDC-IDRI-0190
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0191
Processando paciente LIDC-IDRI-0191
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0193
Processando paciente LIDC-IDRI-0193
  Processando nódulo 1/4
iles ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0194
Processando paciente LIDC-IDRI-0194
  Processando nódulo 2/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0195
Processando paciente LIDC-IDRI-0195
  Processando nódulo 2/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0196
Processando paciente LIDC-IDRI-0196
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0203
Processando paciente LIDC-IDRI-0203
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0206
Processando paciente LIDC-IDRI-0206
  Processando nódulo 1/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0211
Processando paciente LIDC-IDRI-0211
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0212
Processando paciente LIDC-IDRI-0212
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0213
Processando paciente LIDC-IDRI-0213
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0215
Processando paciente LIDC-IDRI-0215
  Processando nódulo 4/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0217
Processando paciente LIDC-IDRI-0217
  Processando nódulo 1/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0221
Processando paciente LIDC-IDRI-0221
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0222
Processando paciente LIDC-IDRI-0222
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0229
Processando paciente LIDC-IDRI-0229
  Processando nódulo 2/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 7/25
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0240
Processando paciente LIDC-IDRI-0240
  Processando nódulo 5/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0242
Processando paciente LIDC-IDRI-0242
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0244
Processando paciente LIDC-IDRI-0244
  Processando nódulo 2/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0248
Processando paciente LIDC-IDRI-0248
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0249
Processando paciente LIDC-IDRI-0249
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0250
Processando paciente LIDC-IDRI-0250
  Processando nódulo 1/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0256
Processando paciente LIDC-IDRI-0256
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0262
Processando paciente LIDC-IDRI-0262
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0263
Processando paciente LIDC-IDRI-0263
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0265
Processando paciente LIDC-IDRI-0265
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0267
Processando paciente LIDC-IDRI-0267
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0272
Processando paciente LIDC-IDRI-0272
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0276
Processando paciente LIDC-IDRI-0276
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0281
Processando paciente LIDC-IDRI-0281
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0284
Processando paciente LIDC-IDRI-0284
  Processando nódulo 3/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0286
Processando paciente LIDC-IDRI-0286
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0287
Processando paciente LIDC-IDRI-0287
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0294
Processando paciente LIDC-IDRI-0294
  Processando nódulo 2/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0305
Processando paciente LIDC-IDRI-0305
  Processando nódulo 8/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0309
Processando paciente LIDC-IDRI-0309
  Processando nódulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0313
Processando paciente LIDC-IDRI-0313
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0321
Processando paciente LIDC-IDRI-0321
  Processando nódulo 2/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0325
Processando paciente LIDC-IDRI-0325
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0332
Processando paciente LIDC-IDRI-0332
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
Processando paciente LIDC-IDRI-0332
  Processando nódulo 1/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0337
Processando paciente LIDC-IDRI-0337
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0346
Processando paciente LIDC-IDRI-0346
  Processando nódulo 1/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0347
Processando paciente LIDC-IDRI-0347
  Processando nódulo 4/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0348
Processando paciente LIDC-IDRI-0348
  Processando nódulo 2/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0351
Processando paciente LIDC-IDRI-0351
  Processando nódulo 2/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0353
Processando paciente LIDC-IDRI-0353
  Processando nódulo 1/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0354
Processando paciente LIDC-IDRI-0354
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0359
Processando paciente LIDC-IDRI-0359
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0362
Processando paciente LIDC-IDRI-0362
  Processando nódulo 2/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0368
Processando paciente LIDC-IDRI-0368
  Processando nódulo 4/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0375
Processando paciente LIDC-IDRI-0375
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0376
Processando paciente LIDC-IDRI-0376
  Processando nódulo 1/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0377
Processando paciente LIDC-IDRI-0377
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0379
Processando paciente LIDC-IDRI-0379
  Processando nódulo 3/15
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0380
Processando paciente LIDC-IDRI-0380
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0385
Processando paciente LIDC-IDRI-0385
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0386
Processando paciente LIDC-IDRI-0386
  Processando nódulo 4/15
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0390
Processando paciente LIDC-IDRI-0390
  Processando nódulo 1/27
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/27
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/27
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/27
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 6/27
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 7/27
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando módulo 8/27
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0399
Processando paciente LIDC-IDRI-0399
  Processando módulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0402
Processando paciente LIDC-IDRI-0402
  Processando módulo 10/27
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0403
Processando paciente LIDC-IDRI-0403
  Processando módulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0407
Processando paciente LIDC-IDRI-0407
  Processando módulo 2/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0412
Processando paciente LIDC-IDRI-0412
  Processando módulo 3/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0415
Processando paciente LIDC-IDRI-0415
  Processando módulo 7/24
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0419
Processando paciente LIDC-IDRI-0419
  Processando módulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0421
Processando paciente LIDC-IDRI-0421
  Processando módulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0423
Processando paciente LIDC-IDRI-0423
  Processando módulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0426
Processando paciente LIDC-IDRI-0426
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0427
Processando paciente LIDC-IDRI-0427
  Processando nódulo 1/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0433
Processando paciente LIDC-IDRI-0433
  Processando nódulo 3/15
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0436
Processando paciente LIDC-IDRI-0436
  Processando nódulo 1/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0437
Processando paciente LIDC-IDRI-0437
  Processando nódulo 1/15
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/15
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/15
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0440
Processando paciente LIDC-IDRI-0440
  Processando nódulo 2/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0443
Processando paciente LIDC-IDRI-0443
  Processando nódulo 1/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0447
Processando paciente LIDC-IDRI-0447
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0450
Processando paciente LIDC-IDRI-0450
  Processando nódulo 8/29
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0456
Processando paciente LIDC-IDRI-0456
  Processando nódulo 1/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0461
Processando paciente LIDC-IDRI-0461
  Processando nódulo 6/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0462
Processando paciente LIDC-IDRI-0462
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0463
Processando paciente LIDC-IDRI-0463
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0464
Processando paciente LIDC-IDRI-0464
  Processando nódulo 3/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0466
Processando paciente LIDC-IDRI-0466
  Processando nódulo 1/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0470
Processando paciente LIDC-IDRI-0470
  Processando nódulo 2/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0473
Processando paciente LIDC-IDRI-0473
  Processando nódulo 1/13
iles ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0481
Processando paciente LIDC-IDRI-0481
  Processando nódulo 4/23
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0485
Processando paciente LIDC-IDRI-0485
  Processando nódulo 2/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0486
Processando paciente LIDC-IDRI-0486
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0488
Processando paciente LIDC-IDRI-0488
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0489
Processando paciente LIDC-IDRI-0489
  Processando nódulo 1/22
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/22
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 7/22
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0492
Processando paciente LIDC-IDRI-0492
  Processando nódulo 2/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0496
Processando paciente LIDC-IDRI-0496
  Processando nódulo 2/11
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0497
Processando paciente LIDC-IDRI-0497
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0507
Processando paciente LIDC-IDRI-0507
  Processando nódulo 1/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0515
Processando paciente LIDC-IDRI-0515
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0523
Processando paciente LIDC-IDRI-0523
  Processando nódulo 5/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0526
Processando paciente LIDC-IDRI-0526
  Processando nódulo 1/19
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 6/19
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0529
Processando paciente LIDC-IDRI-0529
  Processando nódulo 4/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0530
Processando paciente LIDC-IDRI-0530
  Processando nódulo 1/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0537
Processando paciente LIDC-IDRI-0537
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0541
Processando paciente LIDC-IDRI-0541
  Processando nódulo 2/6
Loading [MathJax]/extensions/Safe.js iles ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0542
Processando paciente LIDC-IDRI-0542
  Processando nódulo 3/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0546
Processando paciente LIDC-IDRI-0546
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0559
Processando paciente LIDC-IDRI-0559
  Processando nódulo 1/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0562
Processando paciente LIDC-IDRI-0562
  Processando nódulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0565
Processando paciente LIDC-IDRI-0565
  Processando nódulo 1/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0568
Processando paciente LIDC-IDRI-0568
  Processando nódulo 2/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0569
Processando paciente LIDC-IDRI-0569
  Processando nódulo 3/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0572
Processando paciente LIDC-IDRI-0572
  Processando nódulo 1/3
Loading [MathJax]/extensions/Safe.js iles ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0576
Processando paciente LIDC-IDRI-0576
  Processando nódulo 3/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0578
Processando paciente LIDC-IDRI-0578
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0587
Processando paciente LIDC-IDRI-0587
  Processando nódulo 5/19
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0590
Processando paciente LIDC-IDRI-0590
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0602
Processando paciente LIDC-IDRI-0602
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0604
Processando paciente LIDC-IDRI-0604
  Processando nódulo 2/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0605
Processando paciente LIDC-IDRI-0605
  Processando nódulo 2/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0606
Processando paciente LIDC-IDRI-0606
  Processando nódulo 1/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0610
Processando paciente LIDC-IDRI-0610
  Processando nódulo 2/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0614
Processando paciente LIDC-IDRI-0614
  Processando nódulo 3/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0615
Processando paciente LIDC-IDRI-0615
  Processando nódulo 1/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0617
Processando paciente LIDC-IDRI-0617
  Processando nódulo 2/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0619
Processando paciente LIDC-IDRI-0619
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0620
Processando paciente LIDC-IDRI-0620
  Processando nódulo 1/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0625
Processando paciente LIDC-IDRI-0625
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0639
Processando paciente LIDC-IDRI-0639
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0640
Processando paciente LIDC-IDRI-0640
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0641
Processando paciente LIDC-IDRI-0641
  Processando nódulo 4/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0642
Processando paciente LIDC-IDRI-0642
  Processando nódulo 4/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0645
Processando paciente LIDC-IDRI-0645
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0648
Processando paciente LIDC-IDRI-0648
  Processando nódulo 3/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0649
Processando paciente LIDC-IDRI-0649
  Processando nódulo 1/37
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/37
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/37
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/37
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 6/37
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 7/37
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
  Processando nódulo 8/37
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 10/37
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 11/37
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0655
Processando paciente LIDC-IDRI-0655
  Processando nódulo 6/26
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 8/26
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0660
Processando paciente LIDC-IDRI-0660
  Processando nódulo 5/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0663
Processando paciente LIDC-IDRI-0663
  Processando nódulo 3/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0671
Processando paciente LIDC-IDRI-0671
  Processando nódulo 3/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0674
Processando paciente LIDC-IDRI-0674
  Processando nódulo 5/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0678
Processando paciente LIDC-IDRI-0678
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0686
Processando paciente LIDC-IDRI-0686
  Processando nódulo 17/49
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0697
Processando paciente LIDC-IDRI-0697
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0698
Processando paciente LIDC-IDRI-0698
  Processando nódulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0702
Processando paciente LIDC-IDRI-0702
  Processando nódulo 5/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0704
Processando paciente LIDC-IDRI-0704
  Processando nódulo 1/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0708
Processando paciente LIDC-IDRI-0708
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0709
Processando paciente LIDC-IDRI-0709
  Processando nódulo 2/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0713
Processando paciente LIDC-IDRI-0713
  Processando nódulo 1/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 6/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 7/17
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0715
Processando paciente LIDC-IDRI-0715
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0717
Processando paciente LIDC-IDRI-0717
  Processando nódulo 1/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0722
Processando paciente LIDC-IDRI-0722
  Processando nódulo 1/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0726
Processando paciente LIDC-IDRI-0726
  Processando nódulo 2/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0727
Processando paciente LIDC-IDRI-0727
  Processando nódulo 1/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0732
Processando paciente LIDC-IDRI-0732
  Processando nódulo 2/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0740
Processando paciente LIDC-IDRI-0740
  Processando nódulo 1/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0748
Processando paciente LIDC-IDRI-0748
  Processando nódulo 3/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 6/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 7/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0749
Processando paciente LIDC-IDRI-0749
  Processando nódulo 1/21
iles ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0751
Processando paciente LIDC-IDRI-0751
  Processando nódulo 1/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 7/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0756
Processando paciente LIDC-IDRI-0756
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0759
Processando paciente LIDC-IDRI-0759
  Processando nódulo 2/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0772
Processando paciente LIDC-IDRI-0772
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0773
Processando paciente LIDC-IDRI-0773
  Processando nódulo 5/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0775
Processando paciente LIDC-IDRI-0775
  Processando nódulo 3/32
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/32
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0776
Processando paciente LIDC-IDRI-0776
  Processando nódulo 3/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0777
Processando paciente LIDC-IDRI-0777
  Processando nódulo 2/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0785
Processando paciente LIDC-IDRI-0785
  Processando nódulo 1/28
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 5/28
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0791
Processando paciente LIDC-IDRI-0791
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0796
Processando paciente LIDC-IDRI-0796
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0797
Processando paciente LIDC-IDRI-0797
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0798
Processando paciente LIDC-IDRI-0798
  Processando nódulo 3/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0799
Processando paciente LIDC-IDRI-0799
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0801
Processando paciente LIDC-IDRI-0801
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0807
Processando paciente LIDC-IDRI-0807
  Processando nódulo 4/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0811
Processando paciente LIDC-IDRI-0811
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0819
Processando paciente LIDC-IDRI-0819
  Processando nódulo 2/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0822
Processando paciente LIDC-IDRI-0822
  Processando nódulo 6/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 7/18
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0825
Processando paciente LIDC-IDRI-0825
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0827
Processando paciente LIDC-IDRI-0827
  Processando nódulo 2/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0838
Processando paciente LIDC-IDRI-0838
  Processando nódulo 5/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0843
Processando paciente LIDC-IDRI-0843
  Processando nódulo 2/22
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/22
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
  Processando nódulo 6/22
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0844
Processando paciente LIDC-IDRI-0844
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0849
Processando paciente LIDC-IDRI-0849
  Processando nódulo 5/15
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0850
Processando paciente LIDC-IDRI-0850
  Processando nódulo 1/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0854
Processando paciente LIDC-IDRI-0854
  Processando nódulo 4/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0855
Processando paciente LIDC-IDRI-0855
  Processando nódulo 15/40
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 16/40
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0858
Processando paciente LIDC-IDRI-0858
  Processando nódulo 3/32
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0860
Processando paciente LIDC-IDRI-0860
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0861
Processando paciente LIDC-IDRI-0861
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0866
Processando paciente LIDC-IDRI-0866
  Processando nódulo 2/20
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0869
Processando paciente LIDC-IDRI-0869
  Processando nódulo 2/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0871
Processando paciente LIDC-IDRI-0871
  Processando nódulo 8/24
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0875
Processando paciente LIDC-IDRI-0875
  Processando nódulo 2/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0879
Processando paciente LIDC-IDRI-0879
  Processando nódulo 1/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0884
Processando paciente LIDC-IDRI-0884
  Processando nódulo 2/5
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0886
Processando paciente LIDC-IDRI-0886
  Processando nódulo 3/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0890
Processando paciente LIDC-IDRI-0890
  Processando nódulo 3/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0892
Processando paciente LIDC-IDRI-0892
  Processando nódulo 4/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0893
Processando paciente LIDC-IDRI-0893
  Processando nódulo 2/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be
```

```
LIDC-IDRI-0895
Processando paciente LIDC-IDRI-0895
  Processando nódulo 3/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0902
Processando paciente LIDC-IDRI-0902
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0905
Processando paciente LIDC-IDRI-0905
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0908
Processando paciente LIDC-IDRI-0908
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0910
Processando paciente LIDC-IDRI-0910
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0912
Processando paciente LIDC-IDRI-0912
  Processando nódulo 1/21
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0913
Processando paciente LIDC-IDRI-0913
  Processando nódulo 3/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0914
Processando paciente LIDC-IDRI-0914
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0916
Processando paciente LIDC-IDRI-0916
  Processando nódulo 2/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 8/25
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0919
Processando paciente LIDC-IDRI-0919
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0921
Processando paciente LIDC-IDRI-0921
  Processando nódulo 5/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0925
Processando paciente LIDC-IDRI-0925
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0929
Processando paciente LIDC-IDRI-0929
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0933
Processando paciente LIDC-IDRI-0933
  Processando nódulo 5/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 6/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0935
Processando paciente LIDC-IDRI-0935
  Processando nódulo 1/10
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0936
Processando paciente LIDC-IDRI-0936
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0938
Processando paciente LIDC-IDRI-0938
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0940
Processando paciente LIDC-IDRI-0940
  Processando nódulo 1/5
iles ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0944
Processando paciente LIDC-IDRI-0944
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0951
Processando paciente LIDC-IDRI-0951
  Processando nódulo 1/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0956
Processando paciente LIDC-IDRI-0956
  Processando nódulo 2/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0961
Processando paciente LIDC-IDRI-0961
  Processando nódulo 1/20
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/20
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0963
Processando paciente LIDC-IDRI-0963
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0965
Processando paciente LIDC-IDRI-0965
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0966
Processando paciente LIDC-IDRI-0966
  Processando nódulo 1/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0968
Processando paciente LIDC-IDRI-0968
  Processando nódulo 2/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

```
LIDC-IDRI-0976
Processando paciente LIDC-IDRI-0976
  Processando nódulo 2/6
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0980
Processando paciente LIDC-IDRI-0980
  Processando nódulo 1/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/8
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0987
Processando paciente LIDC-IDRI-0987
  Processando nódulo 1/9
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0993
Processando paciente LIDC-IDRI-0993
  Processando nódulo 1/3
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0994
Processando paciente LIDC-IDRI-0994
  Processando nódulo 5/14
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0997
Processando paciente LIDC-IDRI-0997
  Processando nódulo 3/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 4/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-0999
Processando paciente LIDC-IDRI-0999
  Processando nódulo 4/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1002
Processando paciente LIDC-IDRI-1002
  Processando nódulo 3/16
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1004
Processando paciente LIDC-IDRI-1004
  Processando nódulo 1/12
Loading dicom files ... This may take a moment.
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/12
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1007
Processando paciente LIDC-IDRI-1007
  Processando nódulo 1/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 2/7
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1011
Processando paciente LIDC-IDRI-1011
  Processando nódulo 2/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
  Processando nódulo 3/13
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
LIDC-IDRI-1012
Processando paciente LIDC-IDRI-1012
  Processando nódulo 1/4
Loading dicom files ... This may take a moment.
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to be calculated
```

Tratamento de Dados do Pyradiomics

[voltar ao índice]

Uma vez recolhidos os dados do Pyradiomics é necessário juntar esses dados aos que tinham sido previamente selecionados em [Junção das Anotações Semanticas e Label's]

Features 2D

In [121…
```
df_2d = pd.read_csv('radiomics_features.csv')
```

In [122…
```
df_2d.head()
```

Out[122]:

	diagnostics_Versions_PyRadiomics	diagnostics_Versions_Numpy	diagnostics_Versions_SimpleITK	d
0	v3.0.1	1.23.5	2.4.0	
1	v3.0.1	1.23.5	2.4.0	
2	v3.0.1	1.23.5	2.4.0	
3	v3.0.1	1.23.5	2.4.0	
4	v3.0.1	1.23.5	2.4.0	

5 rows × 126 columns

Vamos começar por verificar se existem valores nulos no dataset:

In [123…
```
df_2d.isna().sum()
```

Out[123]:
```
diagnostics_Versions_PyRadiomics    0
diagnostics_Versions_Numpy          0
diagnostics_Versions_SimpleITK      0
diagnostics_Versions_PyWavelet      0
diagnostics_Versions_Python         0
                                   ..
original_ngtdm_Complexity           0
original_ngtdm_Contrast             0
original_ngtdm_Strength             0
Patient_ID                          0
Nodule_ID                           0
Length: 126, dtype: int64
```

In [124…
```
df_2d.isnull().values.any()
```

Out[124]:
```
False
```

Como podemos comprovar não existem valores nulos.

Agora vamos ver que colunas tem apenas um unico valor para as podermos eliminar, uma vez que não tem significância estatística.

In [125…
```
# Variavel que guarda as colunas com valores unicos
unique_value_counts = df_2d.nunique()
```

In [126…
```
# Remoção das colunas com valores unicos
columns_with_single_unique_value = unique_value_counts[unique_value_counts == 1].ir
print(columns_with_single_unique_value)
df_2d.drop(columns = columns_with_single_unique_value, inplace = True)
```

```
Index(['diagnostics_Versions_PyRadiomics', 'diagnostics_Versions_Numpy',
       'diagnostics_Versions_SimpleITK', 'diagnostics_Versions_PyWavelet',
       'diagnostics_Versions_Python', 'diagnostics_Configuration_Settings',
       'diagnostics_Configuration_EnabledImageTypes',
       'diagnostics_Image-original_Dimensionality',
       'diagnostics_Image-original_Spacing', 'diagnostics_Image-original_Size',
       'diagnostics_Image-original_Minimum',
       'diagnostics_Mask-original_Spacing', 'diagnostics_Mask-original_Size',
       'original_firstorder_10Percentile', 'original_firstorder_Minimum'],
      dtype='object')
```

Para além das colunas que apenas apresentam valores únicos, também podemos remover aquelas features que são irrelevantes por servirem de identificação da máscara e da imagem, uma vez que não se encontram na documentação do pyradiomics:

In [127…
```
colunas_irrelevantes = [
    'diagnostics_Image-original_Hash',
    'diagnostics_Image-original_Mean',
    'diagnostics_Mask-original_Hash',
    'diagnostics_Mask-original_BoundingBox',
    'diagnostics_Mask-original_VoxelNum',
    'diagnostics_Mask-original_VolumeNum',
    'diagnostics_Mask-original_CenterOfMassIndex',
    'diagnostics_Mask-original_CenterOfMass',
]
```

In [128…
```
# Remover as colunas irrelevantes do dataframe
df_2d = df_2d.drop(columns=colunas_irrelevantes)
```

Vamos apenas confirmar o tipo de dados que temos:

In [129…
```
tipos_presentes = df_2d.dtypes.unique()
tipos_presentes_str = [str(tipo) for tipo in tipos_presentes]
print(tipos_presentes_str)
```

```
['float64', 'object', 'int64']
```

Quais são as colunas do tipo object?

In [130…
```
colunas_object = df_2d.select_dtypes(include=['object']).columns.tolist()
print(colunas_object)
```

```
['Patient_ID']
```

Não há nenhum problema no dataset, pois esta coluna será usada para fazer o merge e depois será removida.

Por fim, nesta parte do trabalho na qual não analisamos a real relevância de cada feature, falta-nos apenas juntar as características semânticas de cada nódulo ao dataset 'radiomics_features' para termos todas as variáveis que possam ser importantes analisar. Além disso, é nesse dataset onde está classificado os nódulos como maligno ou benigno.

In [131…
```
# Carregar ambos os DataFrames
df_data = pd.read_csv('data.csv')
df_data.rename(columns = {'Nódulo ID': 'Nodule_ID'}, inplace = True)

# Fazer a junção usando as colunas corretas
df_merged_2d = pd.merge(df_data, df_2d, on = ['Patient_ID', 'Nodule_ID'], how = 'ir
```

In [132…
```
# Exibir as primeiras linhas do DataFrame final
df_merged_2d.head()
```

Out[132]:

	Patient_ID	Nodule_ID	Número de Nódulos	Quantidade de Anotações	Spiculation	Lobulation	Sphericity	Margin	Subt
0	LIDC-IDRI-0001	1	1	4	5	3	3	4	
1	LIDC-IDRI-0003	2	4	4	2	2	4	3	
2	LIDC-IDRI-0004	1	1	4	1	1	2	5	
3	LIDC-IDRI-0007	1	2	4	5	1	4	3	
4	LIDC-IDRI-0011	6	10	4	1	1	5	5	

5 rows × 113 columns

Features 3D

Vamos agora realizar exatamente o mesmo processo que foi feito para as features 2D, mas para as 3D.

In [133…
```
df_3d = pd.read_csv('radiomics_features_3d.csv')
```

In [134…
```
df_3d.head()
```

Out[134]:

	diagnostics_Versions_PyRadiomics	diagnostics_Versions_Numpy	diagnostics_Versions_SimpleITK	d
0	v3.0.1	1.23.5	2.4.0	
1	v3.0.1	1.23.5	2.4.0	
2	v3.0.1	1.23.5	2.4.0	
3	v3.0.1	1.23.5	2.4.0	
4	v3.0.1	1.23.5	2.4.0	

5 rows × 117 columns

Tal como foi feito anteriormente, vamos ver se existem valores nulos:

In [135…
```
df_3d.isna().sum()
```

Out[135]:
```
diagnostics_Versions_PyRadiomics    0
diagnostics_Versions_Numpy          0
diagnostics_Versions_SimpleITK      0
diagnostics_Versions_PyWavelet      0
diagnostics_Versions_Python         0
                                   ..
original_ngtdm_Complexity           0
original_ngtdm_Contrast             0
original_ngtdm_Strength             0
Patient_ID                          0
Nodule_ID                           0
Length: 117, dtype: int64
```

In [136…
```
df_3d.isnull().values.any()
```

Out[136]:
```
False
```

Este dataset também não apresenta valores nulos. Mas será que existem colunas com valores únicos?

In [137…
```
# Variavel que guarda as colunas com valores unicos
unique_value_counts = df_3d.nunique()
```

In [138…
```
# Remoção das colunas com valores unicos
columns_with_single_unique_value = unique_value_counts[unique_value_counts == 1].ir
print(columns_with_single_unique_value)
df_3d.drop(columns = columns_with_single_unique_value, inplace = True)
```

```
Index(['diagnostics_Versions_PyRadiomics', 'diagnostics_Versions_Numpy',
       'diagnostics_Versions_SimpleITK', 'diagnostics_Versions_PyWavelet',
       'diagnostics_Versions_Python', 'diagnostics_Configuration_Settings',
       'diagnostics_Configuration_EnabledImageTypes',
       'diagnostics_Image-original_Dimensionality',
       'diagnostics_Image-original_Spacing',
       'diagnostics_Image-original_Minimum',
       'diagnostics_Mask-original_Spacing', 'original_firstorder_10Percentile',
       'original_firstorder_Median', 'original_firstorder_Minimum'],
      dtype='object')
```

Após uma breve analise, reparamos que as colunas irrelevantes que apenas indicavam as versões utilizadas no radiomics_features.csv, são as mesmas neste dataset, logo também serão eliminadas:

In [139…
```
# Remover as colunas irrelevantes do dataframe
df_3d = df_3d.drop(columns = colunas_irrelevantes)
```

Vamos apenas confirmar o tipo de dados que temos:

In [140…
```
tipos_presentes = df_3d.dtypes.unique()
tipos_presentes_str = [str(tipo) for tipo in tipos_presentes]
print(tipos_presentes_str)
```

```
['object', 'float64', 'int64']
```

Quais são as colunas do tipo object?

In [141…
```
colunas_object = df_3d.select_dtypes(include=['object']).columns.tolist()
print(colunas_object)
```



```
mage-original_Size', 'diagnostics_Mask-original_Size', 'Patient_I
D']
```

In [142...
```
print(df_3d['diagnostics_Image-original_Size'].head())
```

```
0    (133, 512, 512)
1    (140, 512, 512)
2    (241, 512, 512)
3    (145, 512, 512)
4    (128, 512, 512)
Name: diagnostics_Image-original_Size, dtype: object
```

In [143...
```
print(df_3d['diagnostics_Mask-original_Size'].head())
```

```
0    (133, 512, 512)
1    (140, 512, 512)
2    (241, 512, 512)
3    (145, 512, 512)
4    (128, 512, 512)
Name: diagnostics_Mask-original_Size, dtype: object
```

As colunas 'diagnostics_Image-original_Size' e 'diagnostics_Mask-original_Size' estão no tipo string, contudo, de modo a melhorar a qualidade do modelos que iremos usar no futuro, vamos tranformar estas colunas em tuplos e de seguida subdividi-los em três novas colunas, de modo a que cada uma delas fique com os dados de cada posição do tuplo, convertendo tudo para numérico.

In [144...
```
# Converter strings de tupla para tupla numérica apenas se o valor for uma string:
df_3d['diagnostics_Image-original_Size'] = df_3d['diagnostics_Image-original_Size']
    lambda x: tuple(map(float, x.strip("()").split(", "))) if isinstance(x, str) el
)

# Converter strings de tupla para tupla numérica apenas se o valor for uma string:
df_3d['diagnostics_Mask-original_Size'] = df_3d['diagnostics_Mask-original_Size'].a
    lambda x: tuple(map(float, x.strip("()").split(", "))) if isinstance(x, str) el
)

df_3d[['diagnostics_Image-original_Size_x', 'diagnostics_Image-original_Size_y', 'd

df_3d[['diagnostics_Mask-original_Size_x', 'diagnostics_Mask-original_Size_y', 'dia

# Remova as colunas originais com tuplas
df_3d = df_3d.drop(['diagnostics_Image-original_Size', 'diagnostics_Mask-original_S
```

Falta apenas juntar os datasets:

In [145...
```
df_data.rename(columns = {'Nódulo ID': 'Nodule_ID'}, inplace = True)

# Fazer a junção usando as colunas corretas
df_merged_3d = pd.merge(df_data, df_3d, on = ['Patient_ID', 'Nodule_ID'], how = 'ir
```

In [146...
```
df_merged_3d
```

Out[146]:

	Patient_ID	Nodule_ID	Número de Nódulos	Quantidade de Anotações	Spiculation	Lobulation	Sphericity	Margin	Su
0	LIDC-IDRI-0001	1	1	4	5	3	3	4	
1	LIDC-IDRI-0003	2	4	4	2	2	4	3	
2	LIDC-IDRI-0004	1	1	4	1	1	2	5	
3	LIDC-IDRI-0007	1	2	4	5	1	4	3	
4	LIDC-IDRI-0011	6	10	4	1	1	5	5	
...	...	...	...	...	...	...	...	...	
405	LIDC-IDRI-1007	1	2	4	1	1	3	4	
406	LIDC-IDRI-1007	2	2	3	2	2	4	4	
407	LIDC-IDRI-1011	2	4	4	1	3	3	4	
408	LIDC-IDRI-1011	3	4	4	1	1	3	3	
409	LIDC-IDRI-1012	1	1	4	1	1	4	2	

410 rows × 109 columns

Guardar os datasets

Antes de guardar os datsets podemos apenas remover as colunas 'Patient_ID' e 'Nodule_ID' uma vez que não serão mais necessárias. O seu único propósito era facilitar a junção das diferentes features, logo, agora não tem qualquer utilidade, pois cada entrada corresponde a um nódulo. Para além dessas também podemos remover 'Número de Nódulos' e 'Quantidade de Anotações'.

In [147…
```
df_merged_2d = df_merged_2d.drop(columns=['Patient_ID', 'Nodule_ID', 'Número de Nód
df_merged_3d = df_merged_3d.drop(columns=['Patient_ID', 'Nodule_ID', 'Número de Nód
```

In [148…
```
# Salvar o DataFrame em um arquivo CSV com codificação UTF-8
df_merged_2d.to_csv('data_2d.csv', index=False, encoding='utf-8')
df_merged_3d.to_csv('data_3d.csv', index=False, encoding='utf-8')
```

Feature Importance and Selection

[voltar ao índice]

Após extrair as features semânticas a partir do pyradiomics, obtivemos um dataset com um número elevado de features. Independentemente do modelo de classificação que escolhermos utilizar, é necessário implementar métodos que permitam selecionar as colunas (features) mais relevantes para a classificação dos nódulos, excluindo outras. A seleção das features foi, então, apoiada em três métodos distintos, utilizados na bibliografia referenciada abaixo. Estes métodos permitem reduzir significativamente a dimensionalidade do problema, o que contribui para uma maior eficácia por parte dos modelos de classificação. Esses métodos são:

- Teste de Hipóteses: t-test
- Principal Component Analysis
- Random Forest

Uma vez que as funções a serem utilizadas para os modelos 2d e 3d são as mesmas, iremos reazilar ambas nesta parte do trabalho.

In [5]:
```
data_2d = pd.read_csv('data_2d.csv')
data_3d = pd.read_csv('data_3d.csv')
```

1.Teste de Hipóteses: t-test - Tor23

[Feature Importance and Selection]

Os testes de hipóteses são utilizados para apoiar a tomada de decisões. Neste caso, permitem avaliar a importância de cada feature do dataset obtido e selecionar as features relevantes à classificação. O teste utilizado pela bibliografia é o t-test. Trata-se de um teste estatístico usado para identificar que features são estatisticamente significativas para a classificação. Serão, assim, selecionadas as features cujo valor de significância (p-value) for inferior ao valor usualmente utilizado como threshold (0.05).

Começamos por preparar o dataset, retirando a coluna 'Malignancy' (target). A variável X guarda as features e a variável y guarda o target.

In [6]:
```
X_2d = data_2d.drop(columns=["Malignancy"])
y_2d = data_2d['Malignancy']  # Target (0: benign, 1: malignant)
print("Dados 2:")
print(y_2d)

print("\n ------------------------ \n")

X_3d = data_3d.drop(columns=["Malignancy"])
y_3d = data_3d['Malignancy']
print("Dados 3:")
print(y_3d)
```

```
Dados 2:
0      1
1      1
2      0
3      1
4      0
      ..
405    0
406    1
407    1
408    1
409    0
Name: Malignancy, Length: 410, dtype: int64

 ------------------------

Dados 3:
0      1
1      1
2      0
3      1
4      0
      ..
405    0
406    1
407    1
408    1
409    0
Name: Malignancy, Length: 410, dtype: int64
```

De seguida, iremos separar os dados de acordo com a sua classificação (como benigno ou maligno), obtendo 2 grupos distintos.

In [7]:
```
benign_group_2d = X_2d[y_2d == 0]
malignant_group_2d = X_2d[y_2d == 1]

benign_group_3d = X_3d[y_3d == 0]
malignant_group_3d = X_3d[y_3d == 1]
```

Podemos agora realizar um t-test para cada feature e guardar os valor de significância. As features aparecem por ordem de significância.

In [8]:
```
t_test_results_2d = {}
for feature in X_2d.columns:
    t_stat, p_value = ttest_ind(benign_group_2d[feature], malignant_group_2d[featur
    t_test_results_2d[feature] = p_value
sorted_t_test_results_2d = pd.Series(t_test_results_2d).sort_values()
print("2D Features ranked by significance (p-value):")
print(sorted_t_test_results_2d)

print("\n ------------------------ \n")

t_test_results_3d = {}
for feature in X_3d.columns:
    t_stat, p_value = ttest_ind(benign_group_3d[feature], malignant_group_3d[featur
    t_test_results_3d[feature] = p_value
sorted_t_test_results_3d = pd.Series(t_test_results_3d).sort_values()
print("3D Features ranked by significance (p-value):")
print(sorted_t_test_results_3d)
```

```
2D Features ranked by significance (p-value):
Calcification                         6.293948e-51
original_shape2D_Sphericity           8.013241e-32
original_shape2D_MajorAxisLength      1.603369e-28
original_shape2D_MaximumDiameter      4.299303e-28
original_shape2D_MinorAxisLength      7.347870e-27
                                          ...
original_ngtdm_Busyness               7.412270e-01
original_glcm_JointEntropy            8.297815e-01
original_glcm_DifferenceEntropy       8.323663e-01
original_glcm_SumEntropy              9.508391e-01
original_glcm_InverseVariance         9.829819e-01
Length: 108, dtype: float64

 ------------------------

3D Features ranked by significance (p-value):
Calcification                          6.293948e-51
Spiculation                            1.115950e-25
Lobulation                             9.117091e-25
Margin                                 7.064467e-22
Subtlety                               8.737994e-20
                                           ...
original_glcm_Imc2                     9.515291e-01
diagnostics_Image-original_Size_y               NaN
diagnostics_Image-original_Size_z               NaN
diagnostics_Mask-original_Size_y                NaN
diagnostics_Mask-original_Size_z                NaN
Length: 104, dtype: float64
/var/folders/2t/mv0q1c7j2z97cgvtlhbfrhhm0000gn/T/ipykernel_67329/4215019795.py:3:
RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic
cancellation. This occurs when the data are nearly identical. Results may be unrel
iable.
  t_stat, p_value = ttest_ind(benign_group_2d[feature], malignant_group_2d[featur
e], equal_var=False)
/var/folders/2t/mv0q1c7j2z97cgvtlhbfrhhm0000gn/T/ipykernel_67329/4215019795.py:13:
RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic
cancellation. This occurs when the data are nearly identical. Results may be unrel
iable.
  t_stat, p_value = ttest_ind(benign_group_3d[feature], malignant_group_3d[featur
e], equal_var=False)
```

As features mais significativas para a classificação são aquelas que se encontram acima do threshold definido, isto é, cujo p-value é menor que 0.05.

In [9]:
```
significant_features_2d = sorted_t_test_results_2d[sorted_t_test_results_2d < 0.05]
print ("Número de features selecionadas para 2D:", len(significant_features_2d))

print("\n ------------------------ \n")

significant_features_3d = sorted_t_test_results_3d[sorted_t_test_results_3d < 0.05]
print ("Número de features selecionadas para 3D:", len(significant_features_3d))
```

```
Número de features selecionadas para 2D: 49

 ------------------------

Número de features selecionadas para 3D: 58
```

Como podemos ver o t-test escolheu as features que considerou mais significativas para cada dataset:

In [10]:
```
print("Dataset com features significativas 2D:")
data_ttest_2d = data_2d[significant_features_2d]
pd.concat([data_ttest_2d, y_2d], axis=1)
```

```
Dataset com features significativas 2D:
```

Out[10]:

	Calcification	original_shape2D_Sphericity	original_shape2D_MajorAxisLength	original_shape2D
0	6	0.116476	57.542760	
1	6	0.237705	17.090541	
2	3	0.293095	9.244896	
3	6	0.145259	40.718270	
4	3	0.288620	10.327956	
...	...	...	...	
405	6	0.245244	13.852837	
406	6	0.112044	61.575217	
407	6	0.138573	51.164214	
408	6	0.154955	41.452617	
409	6	0.276215	12.494295	

410 rows × 50 columns

In [11]:
```
print("Dataset com features significativas 3D:")
data_ttest_3d = data_3d[significant_features_3d]
pd.concat([data_ttest_3d, y_3d], axis=1)
```

```
Dataset com features significativas 3D:
```

Out[11]:

	Calcification	Spiculation	Lobulation	Margin	Subtlety	original_glrlm_GrayLevelNonUniformit
0	6	5	3	4	5	757.88036
1	6	2	2	3	5	88.38461
2	3	1	1	5	2	34.51526
3	6	5	1	3	5	437.07015
4	3	1	1	5	3	33.84615
...	...	...	...	...	...	
405	6	1	1	4	3	41.76923
406	6	2	2	4	5	1239.76923
407	6	1	3	4	5	573.30769
408	6	1	1	3	5	444.76923
409	6	1	1	2	4	47.84615

410 rows × 59 columns

2. Principal Component Analysis - LLZ18

[Feature Importance and Selection]

A análise dos componentes principais (PCA) é um método usado em machine learning para resolver o problema da elevada dimensionalidade de dados. É um procedimento matemático que usa as dependências entre várias variáveis, no nosso caso as dependências entre as features, de maneira a reduzir o conjunto de dados, sem perder muita informação.

Começamos por preparar o dataset, retirando a coluna 'Malignancy' (target). A variável X guarda as features e a variável y guarda o target.

In [12]:
```
X_2d = data_2d.drop(columns=["Malignancy"])
y_2d = data_2d['Malignancy']  # Target (0: benign, 1: malignant)

X_3d = data_3d.drop(columns=["Malignancy"])
y_3d = data_3d['Malignancy']
```

Para obter resultados mais verosímeis, optámos por normalizar os dados, com o método 'fit_transform'. Este método é responsável por ajustar os dados de todas as colunas, de modo a que tenham média = 0 e desvio padrão = 1 (ajustando-os a uma distribuição normal).

In [13]:
```
scaler_2d = StandardScaler()
X_scaled_2d = scaler_2d.fit_transform(X_2d)

scaler_3d = StandardScaler()
X_scaled_3d = scaler_3d.fit_transform(X_3d)
```

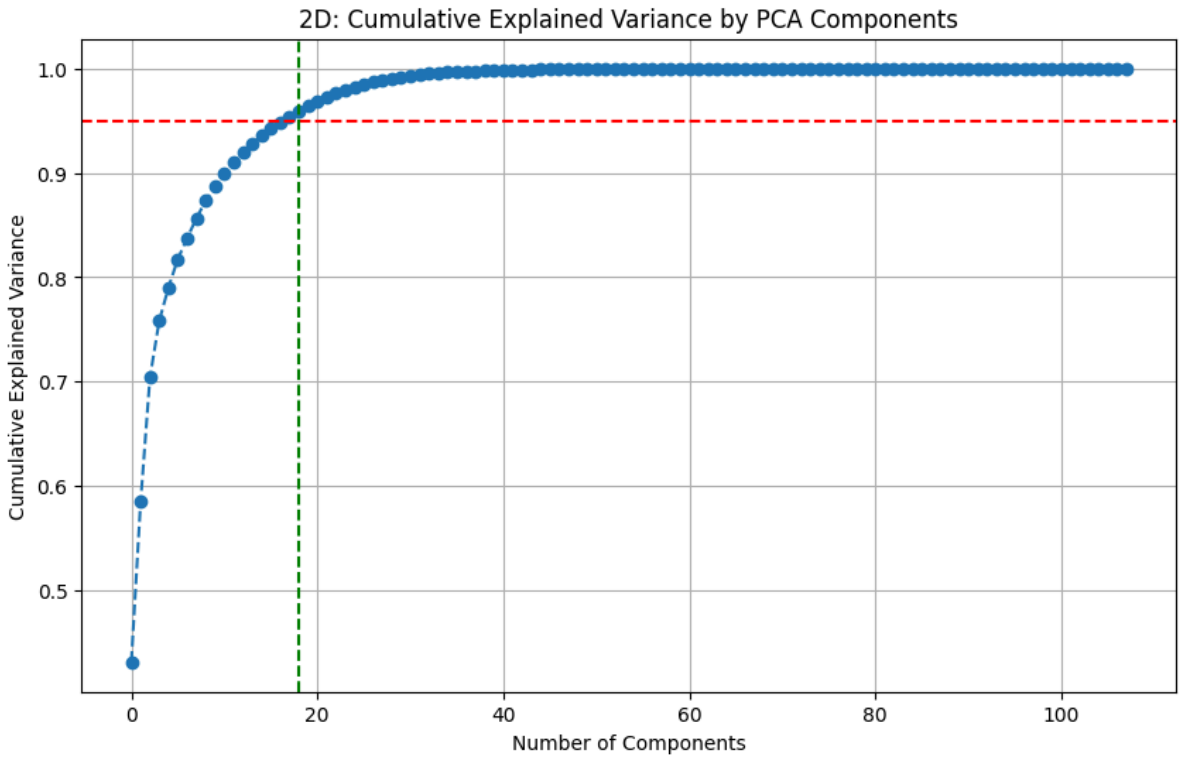
Tendo os dados normalizados, podemos aplicar o PCA. O método consiste em obter componentes (features) através da diagonalização da matriz de covariância dos dados de cada coluna, que permite indicar o nível de correlação entre os pares. A matriz de covariância é então decomposta nos seus autovalores e autovetores. Os autovetores determinam as direções dos novos eixos (componentes principais), enquanto os autovalores determinam a sua importância.

In [14]:
```
pca_2d = PCA()
X_pca_2d = pca_2d.fit_transform(X_scaled_2d)

pca_3d = PCA()
X_pca_3d = pca_3d.fit_transform(X_scaled_3d)
```

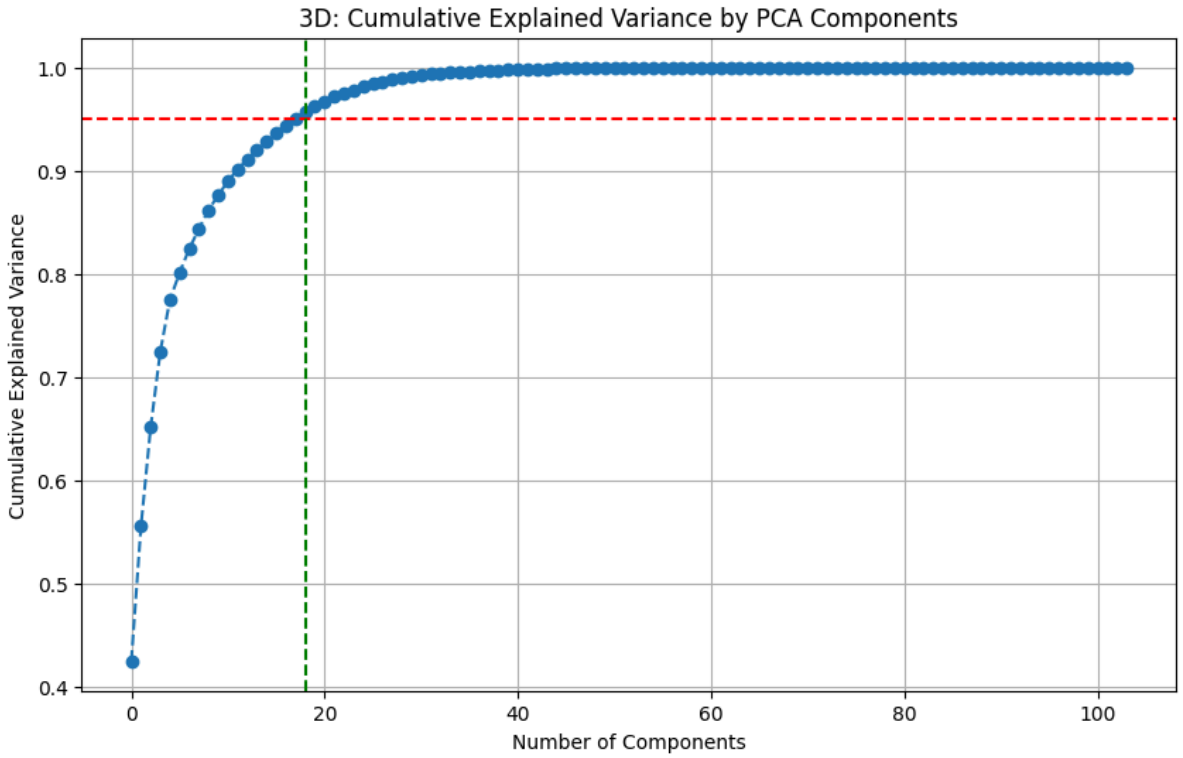
É calculada a variância para cada componente principal e em seguida a variância cumulativa.

In [15]:
```
explained_variance_2d = pca_2d.explained_variance_ratio_
cumulative_variance_2d = explained_variance_2d.cumsum()

plt.figure(figsize=(10, 6))
plt.plot(cumulative_variance_2d, marker='o', linestyle='--')
plt.title('2D: Cumulative Explained Variance by PCA Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid()
.95, color='r', linestyle='--')  # 95% threshold line
```

```
plt.axvline(x=np.argmax(cumulative_variance_2d >= 0.95) + 1, color='g', linestyle='
plt.show()
```



In [16]:
```
explained_variance_3d = pca_3d.explained_variance_ratio_
cumulative_variance_3d = explained_variance_3d.cumsum()

plt.figure(figsize=(10, 6))
plt.plot(cumulative_variance_3d, marker='o', linestyle='--')
plt.title('3D: Cumulative Explained Variance by PCA Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid()
plt.axhline(y=0.95, color='r', linestyle='--')  # 95% threshold line
plt.axvline(x=np.argmax(cumulative_variance_3d >= 0.95) + 1, color='g', linestyle='
plt.show()
```



Determinamos o número de componentes para a variância desejada.

In [17]:
```
n_components_2d = np.argmax(cumulative_variance_2d >= 0.95) + 1 # quantidade de com
pca_final_2d = PCA(n_components = n_components_2d)
reduced_features_2d = pca_final_2d.fit_transform(X_scaled_2d)
print("Número de features selecionadas 2D: ", n_components_2d)

print("\n ------------------------ \n")

n_components_3d = np.argmax(cumulative_variance_3d >= 0.95) + 1 # quantidade de com
pca_final_3d = PCA(n_components = n_components_3d)
reduced_features_3d = pca_final_3d.fit_transform(X_scaled_3d)
print("Número de features selecionadas 3D: ", n_components_3d)
```

```
Número de features selecionadas 2D:  18

 ------------------------

Número de features selecionadas 3D:  18
```

Como podemos ver o PCA determinou o número de features a agrupar de acordo com a variância encontrada, sendo essas:

In [18]:
```
print("Dataset com features agrupadas 2D:")
data_pca_2d = pd.DataFrame(data=reduced_features_2d, columns=[f'PC{i}' for i in rar
pd.concat([data_pca_2d, y_2d.reset_index(drop=True)], axis=1)
```

```
Dataset com features agrupadas 2D:
```

Out[18]:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	P
0	0.605527	14.312994	2.618862	-5.010918	-1.873619	0.486000	3.780943	4.509357	-3.0005
1	-2.267379	-1.819947	0.040090	0.846899	-1.514118	-0.992908	-0.811545	0.548039	-0.1913
2	5.058170	-2.480426	-3.601613	-4.459463	-1.977084	-3.833077	1.266602	3.487037	0.1742
3	26.065083	6.727777	-6.985184	4.449976	-1.804223	1.655932	-2.786482	-3.047505	1.8274
4	-2.158470	-3.958437	-0.673922	0.104410	1.383675	-1.097692	0.958196	-0.653246	0.2457
...	...	...	...	...	...	...	...	...	
405	-2.233398	-2.998710	-0.378282	0.497056	-0.144041	-1.423559	-0.648699	0.289676	0.1224
406	-2.923210	11.711044	5.856770	4.536946	0.801008	-0.007470	1.331167	-1.196007	0.6149
407	-2.716925	5.627624	3.129216	2.034017	-1.944248	1.328403	0.538597	-0.491240	0.2763
408	-2.511517	3.152825	2.056172	1.597862	-1.588072	0.221263	0.574015	-0.519261	0.3861
409	-2.493756	-2.801108	-0.163597	-0.006073	0.120255	1.392496	-1.301845	0.218497	-0.2664

410 rows × 19 columns

In [19]:
```
print("Dataset com features agrupadas 3D:")
data_pca_3d = pd.DataFrame(data=reduced_features_3d, columns=[f'PC{i}' for i in rar
pd.concat([data_pca_3d, y_3d.reset_index(drop=True)], axis=1)
```

```
Dataset com features agrupadas 3D:
```

Out[19]:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	P
0	-0.782122	-4.117717	1.807559	3.238631	-2.126003	0.470964	0.041700	-1.212518	-0.2280
1	-2.459356	2.218242	-1.818623	2.089689	0.356737	1.117322	-0.424158	-0.750689	-0.4551
2	13.960910	0.629641	-1.754254	-1.768815	-1.563777	-2.868636	-2.049989	-1.101771	-1.3600
3	18.196420	-6.671389	-6.882279	1.530733	2.834788	2.809031	-2.197428	2.412911	1.9639
4	-2.639728	3.325768	-2.169896	0.994315	-0.182417	-0.915270	1.220950	1.471481	0.1514
...	...	...	...	...	...	...	...	...	
405	-2.682271	3.198533	-2.295293	1.549300	0.057934	-0.301314	0.212454	-0.055443	0.9605
406	-2.269498	-5.455861	4.720836	7.225682	2.834643	-3.934610	3.135748	-0.186134	0.6010
407	-2.505729	-0.167273	0.442583	2.612708	2.009912	0.850195	-0.439097	0.373260	-1.7192
408	-2.686433	0.246159	0.531057	1.261710	1.665685	0.842926	-0.394356	0.287286	-1.0636
409	-2.196771	3.355261	-3.609834	4.514218	-0.464621	0.370936	0.322757	-0.490927	1.1526

410 rows × 19 columns

3. Random Forest - CBMLN

[Feature Importance and Selection]

O algoritmo Random Forest também é usado para selecionar as features mais relevantes ao problema de classificação. Para tal, atribui determinada importância a cada feature na fase de treino do modelo, que corresponde à contribuição de cada feature para a classificação.

Começamos por preparar o dataset, retirando a coluna 'Malignancy' (target). A variável X guarda as features e a variável y guarda o target.

In [20]:
```
X_2d = data_2d.drop(columns=["Malignancy"])
y_2d = data_2d["Malignancy"]  # Target (0: benign, 1: malignant)

X_3d = data_3d.drop(columns=["Malignancy"])
y_3d = data_3d["Malignancy"]  # Target (0: benign, 1: malignant)
```

Podemos agora inicializar o modelo Random Forest para as features.

In [21]:
```
rf_features_2d = RandomForestClassifier(n_estimators=100, random_state=42)
rf_features_2d.fit(X_2d, y_2d)
```

Out[21]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

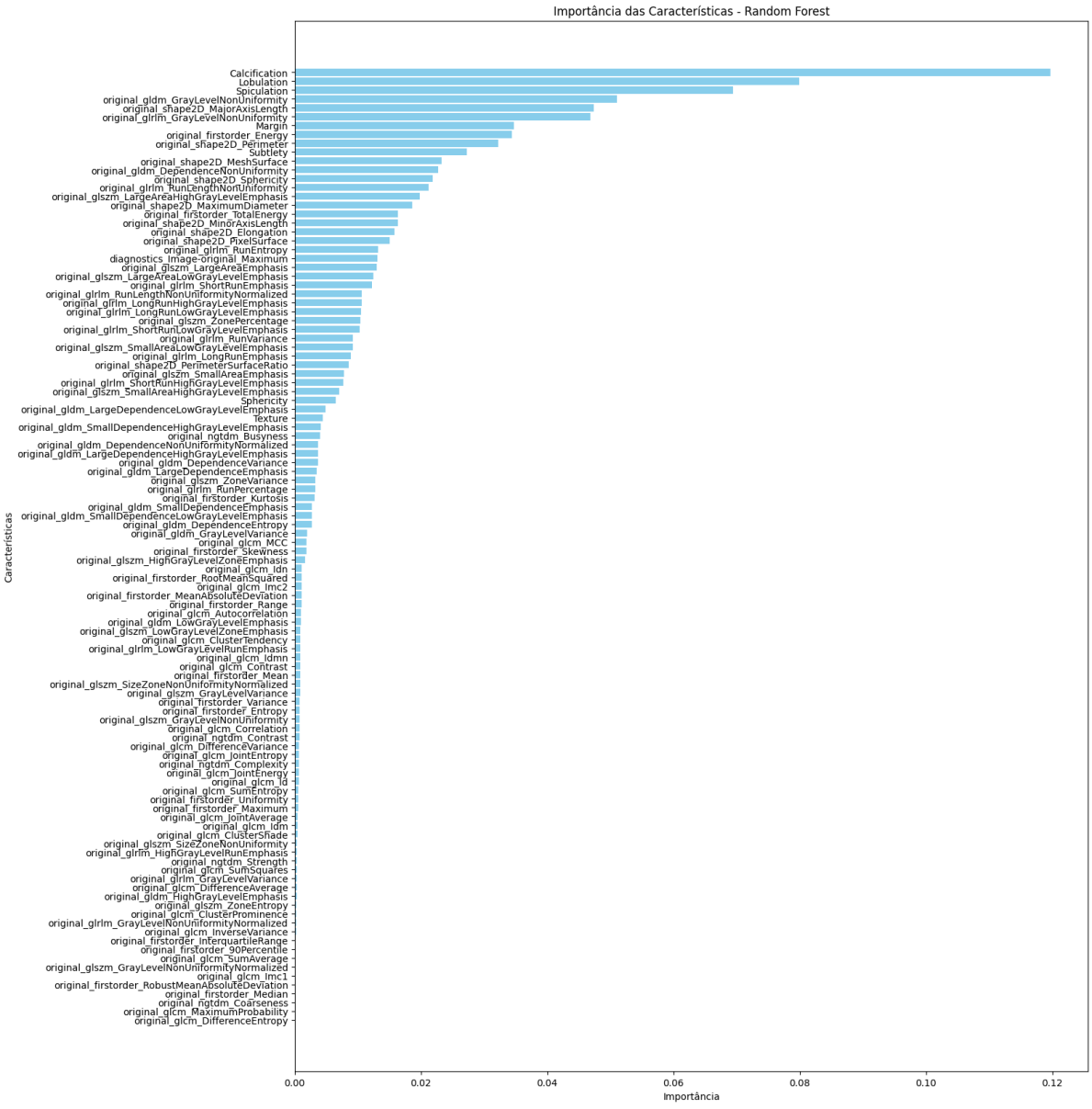
In [22]:
```
rf_features_3d = RandomForestClassifier(n_estimators=100, random_state=42)
rf_features_3d.fit(X_3d, y_3d)
```

Out[22]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Para obter o valor de importância de cada feature, usamos o atributo do classificador .feature*importances*. Tendo estes valores, criamos um dataset com os nomes das features por ordem de importância

In [23]:
```
feature_importances_2d = rf_features_2d.feature_importances_
importances_df_2d = pd.DataFrame({"Feature": X_2d.columns, "Importance": feature_im
importances_df_2d = importances_df_2d.sort_values(by="Importance", ascending=False)
print("Feature Importances 2D:")
print(importances_df_2d)

plt.figure(figsize=(15, 20))
plt.barh(importances_df_2d["Feature"], importances_df_2d["Importance"], color="skyb
plt.xlabel("Importância")
plt.ylabel("Características")
plt.title("Importância das Características - Random Forest")
plt.gca().invert_yaxis()  # Inverte o eixo para mostrar as características mais imp
plt.show()
```

```
Feature Importances 2D:
                                                Feature  Importance
6                                         Calcification    0.119684
1                                           Lobulation    0.079919
0                                           Spiculation    0.069371
93                  original_gldm_GrayLevelNonUniformity    0.051023
9                       original_shape2D_MajorAxisLength    0.047379
..                                                   ...         ...
27   original_firstorder_RobustMeanAbsoluteDeviation    0.000050
25                        original_firstorder_Median    0.000000
104                        original_ngtdm_Coarseness    0.000000
53                     original_glcm_MaximumProbability    0.000000
40                      original_glcm_DifferenceEntropy    0.000000

[108 rows x 2 columns]
```

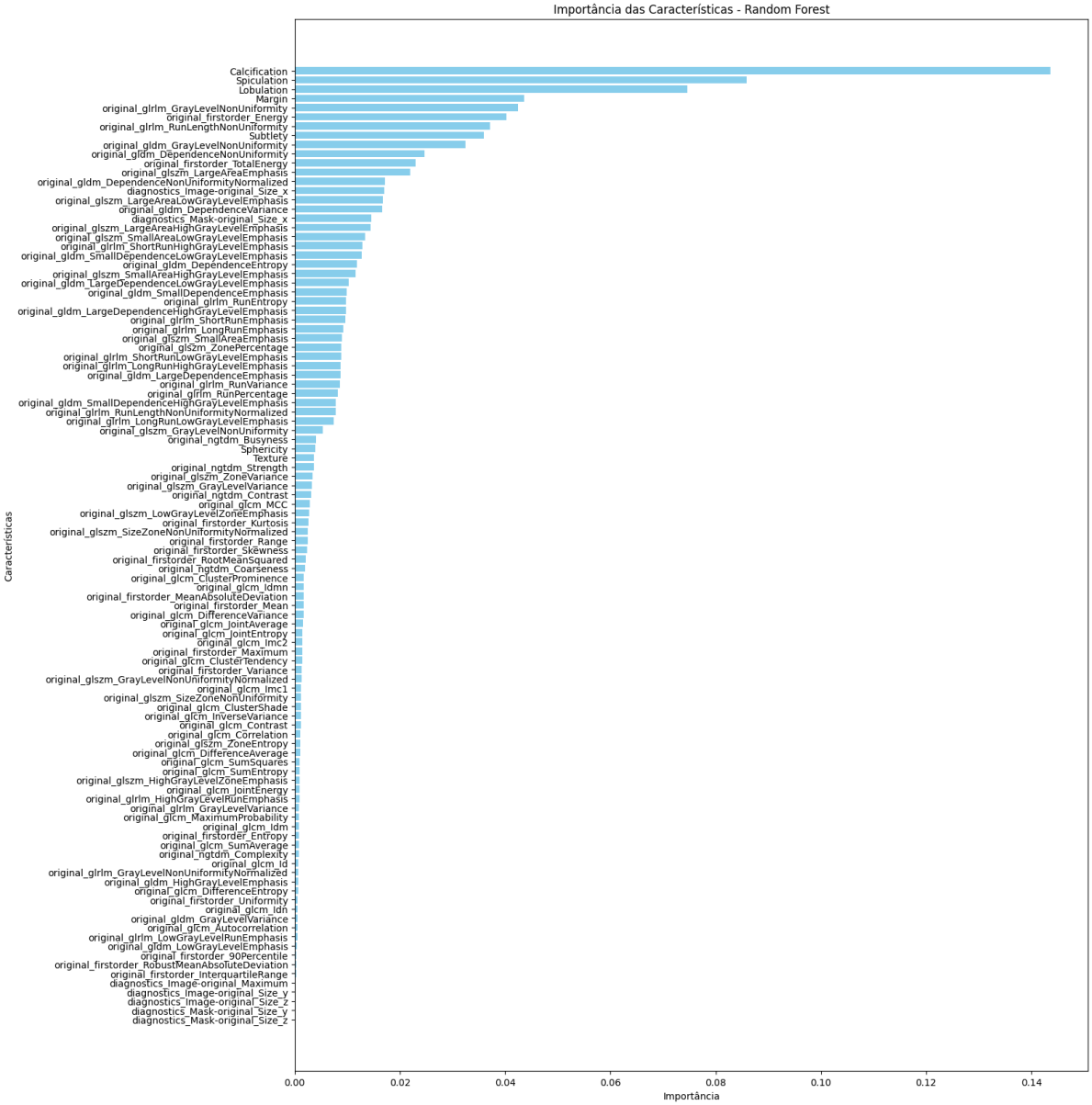


In [24]:

```
feature_importances_3d = rf_features_3d.feature_importances_
importances_df_3d = pd.DataFrame({"Feature": X_3d.columns, "Importance": feature_im
importances_df_3d = importances_df_3d.sort_values(by="Importance", ascending=False)
print("Feature Importances 3D:")
print(importances_df_3d)

plt.figure(figsize=(15, 20))
plt.barh(importances_df_3d["Feature"], importances_df_3d["Importance"], color="skyb
plt.xlabel("Importância")
plt.ylabel("Características")
plt.title("Importância das Características - Random Forest")
```

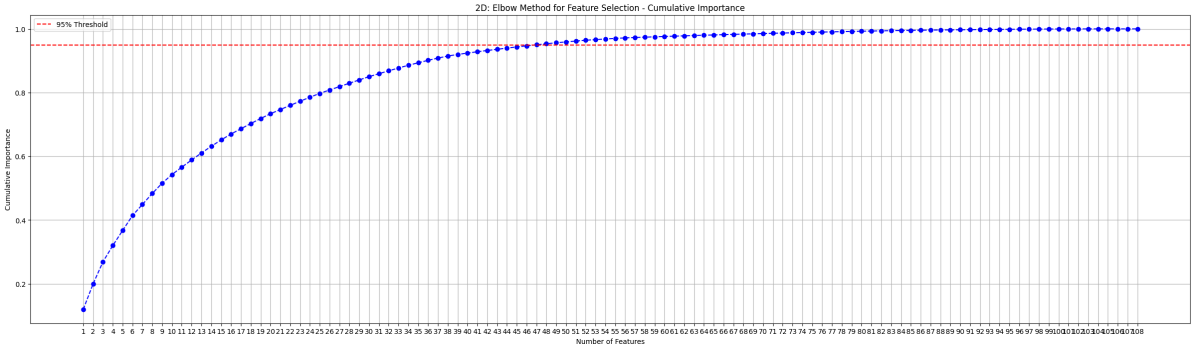
```
plt.gca().invert_yaxis()  # Inverte o eixo para mostrar as características mais imp
plt.show()
```

```
Feature Importances 3D:
                                  Feature    Importance
6                             Calcification  1.436337e-01
0                               Spiculation  8.593508e-02
1                                Lobulation  7.461222e-02
3                                    Margin  4.364640e-02
47   original_glrlm_GrayLevelNonUniformity  4.244826e-02
..                                     ...           ...
7       diagnostics_Image-original_Maximum  3.615520e-07
99       diagnostics_Image-original_Size_y  0.000000e+00
100      diagnostics_Image-original_Size_z  0.000000e+00
102       diagnostics_Mask-original_Size_y  0.000000e+00
103       diagnostics_Mask-original_Size_z  0.000000e+00

[104 rows x 2 columns]
```

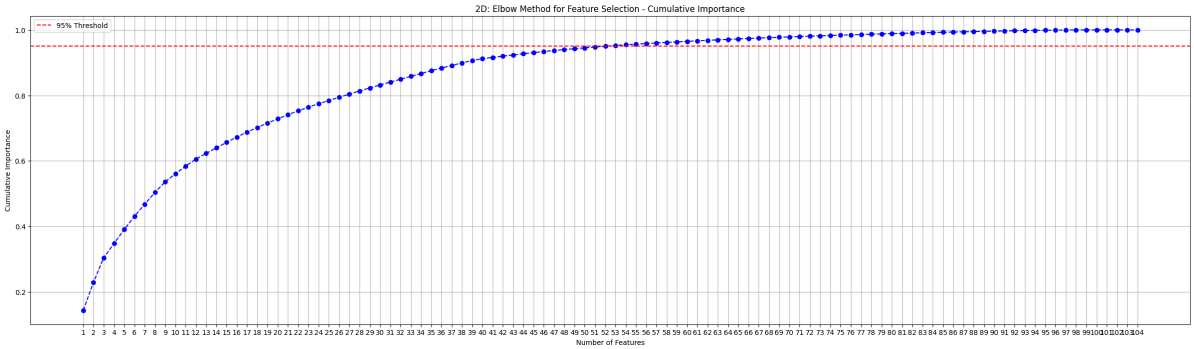


Temos apenas o valor de importância de cada feature. Para encontrarmos o número de features relevantes decidimos apoiar a decisão num método utilizado em machine learning, método **Elbow**, que encontra o threshold ideal para a seleção de features com base na sua importância, usualmente 95%. Para aplicar o método, calculamos a importância cumulativa de cada feature e selecionamos as que se encontram abaixo do valor de corte.

In [25]:
```
indices_2d = np.argsort(feature_importances_2d)[::-1]
cumulative_importance_2d = np.cumsum(feature_importances_2d[indices_2d])

num_features_2d = len(feature_importances_2d)
x = range(1, num_features_2d + 1)

plt.figure(figsize=(30, 8))
plt.plot(x, cumulative_importance_2d, marker='o', linestyle='--', color='blue')
plt.axhline(y=0.95, color='red', linestyle='--', label='95% Threshold')  # Optional
plt.xlabel("Number of Features")
plt.ylabel("Cumulative Importance")
plt.title("2D: Elbow Method for Feature Selection - Cumulative Importance")
plt.xticks(x)
plt.grid()
plt.legend()
plt.show()
```



In [26]:
```
indices_3d = np.argsort(feature_importances_3d)[::-1]
cumulative_importance_3d = np.cumsum(feature_importances_3d[indices_3d])

num_features_3d = len(feature_importances_3d)
x = range(1, num_features_3d + 1)

plt.figure(figsize=(30, 8))
plt.plot(x, cumulative_importance_3d, marker='o', linestyle='--', color='blue')
plt.axhline(y=0.95, color='red', linestyle='--', label='95% Threshold')  # Optional
plt.xlabel("Number of Features")
plt.ylabel("Cumulative Importance")
plt.title("2D: Elbow Method for Feature Selection - Cumulative Importance")
plt.xticks(x)
plt.grid()
plt.legend()
plt.show()
```



Por fim, selecionamos as features relevantes, com base no threshold=0.95

In [27]:
```
threshold = 0.95

selected_indices_2d = np.where(cumulative_importance_2d >= threshold)[0]
if selected_indices_2d.size > 0:
```

```
    selected_features_indices_2d = selected_indices_2d[0]
    selected_features_2d = importances_df_2d['Feature'].head(selected_features_indi
else:
    selected_features_2d = []
print("Número de features selecionadas 2D:", len(selected_features_2d))

print("\n ------------------------ \n")

selected_indices_3d = np.where(cumulative_importance_3d >= threshold)[0]
if selected_indices_3d.size > 0:
    selected_features_indices_3d = selected_indices_3d[0]
    selected_features_3d = importances_df_3d['Feature'].head(selected_features_indi
else:
    selected_features_3d = []
print("Número de features selecionadas 3D:", len(selected_features_3d))
```

```
Número de features selecionadas 2D: 47

 ------------------------

Número de features selecionadas 3D: 52
```

Como podemos ver o RandomForest selecionou as features que considera relevantes.

In [28]:
```
print("Features selecionadas 2D:")
data_rf_2d = data_2d[selected_features_2d]
pd.concat([data_rf_2d, y_2d], axis=1)
```

```
Features selecionadas 2D:
```

Out[28]:

	Calcification	Lobulation	Spiculation	original_gldm_GrayLevelNonUniformity	original_shape2D
0	6	3	5	160.036585	
1	6	2	2	33.000000	
2	3	1	1	12.142857	
3	6	1	5	43.079208	
4	3	1	1	18.000000	
...	...	...	...	...	
405	6	1	1	24.000000	
406	6	2	2	157.000000	
407	6	3	1	101.000000	
408	6	1	1	79.000000	
409	6	1	1	21.000000	

410 rows × 48 columns

In [29]:
```
print("Features selecionadas 3D:")
data_rf_3d = data_3d[selected_features_3d]
data_rf_3d = pd.concat([data_rf_3d, y_3d], axis=1)
data_rf_3d
```

```
Features selecionadas 3D:
```

Out[29]:

	Calcification	Spiculation	Lobulation	Margin	original_glrlm_GrayLevelNonUniformity	origina
0	6	5	3	4	757.880362	
1	6	2	2	3	88.384615	
2	3	1	1	5	34.515260	
3	6	5	1	3	437.070150	
4	3	1	1	5	33.846154	
...	...	...	...	...	...	
405	6	1	1	4	41.769231	
406	6	2	2	4	1239.769231	
407	6	1	3	4	573.307692	
408	6	1	1	3	444.769231	
409	6	1	1	2	47.846154	

410 rows × 53 columns

Training Models

[voltar ao índice]

Após preparar os datasets com a seleção das features relevantes podemos iniciar o treino dos modelos para classificação dos nódulos. Para isso decidimos utilizar diversos modelos e compará-los, sendo justificada a utilização de cada um na sua respetiva parte. Além disso realizamos testes para os diferentes modos de Features Importance and Selection utilizados.

Critérios de Avaliação usados:

-> Accuracy: Mede a proporção de previsões corretas em relação ao total de previsões. Bom para problemas de classificação binária.

-> Log Loss: Avalia a probabilidade predita pelo modelo para a classe correta. No caso da classificação de nódulos, é essencial saber se o modelo está "muito confiante" ao classificar um nódulo como benigno ou maligno. Métrica valiosa para decisões médicas, onde a decisão pode custar vidas humanas.

-> Dice Coefficient: Métrica usada para avaliar a sobreposição de duas amostras, especialmente em problemas de segmentação de imagem. Importante para capturar bem os nódulos malignos, sem ignorar os benignos.

Estes critérios são usados no estudo [DCLC]

K-Fold Cross Validation - Divisão do dataset

K-fold Cross Validation é uma técnica de divisão de dados que consiste em realizar a divisão de treino/teste k vezes. Os resultados finais são posteriormente obtidos através das médias dos resultados de cada iteração. Este método é utilizado para garantir que o modelo não está a ser treinado com um dataset específico, mas sim com o dataset completo. [CBMLN]

Vamos usar o método k_fold para dividir os dados de teste e treino e usar os mesmos folds para os diferentes modelos:

In [30]:
```
def k_fold_cross_validation(data, labels, k=5, random_state=42):
    fold = []
    kf = KFold(n_splits=k, shuffle=True, random_state=random_state)

    for train_index, test_index in kf.split(data):
        # Converte os subconjuntos de treino e teste em listas
        X_train, X_test = data.iloc[train_index].values.tolist(), data.iloc[test_ir
        y_train, y_test = labels.iloc[train_index].values.tolist(), labels.iloc[tes
        fold.append([X_train, X_test, y_train, y_test])

    return fold
```

In [31]:
```
fold_2d = k_fold_cross_validation(X_2d, y_2d)
fold_2d_pca = k_fold_cross_validation(data_pca_2d, y_2d)
fold_2d_rf = k_fold_cross_validation(data_rf_2d,y_2d)
fold_2d_ttest = k_fold_cross_validation(data_ttest_2d,y_2d)

fold_3d = k_fold_cross_validation(X_3d, y_3d)
fold_3d_pca = k_fold_cross_validation(data_pca_3d, y_3d)
fold_3d_rf = k_fold_cross_validation(data_rf_3d,y_3d)
fold_3d_ttest = k_fold_cross_validation(data_ttest_3d,y_3d)
```

Antes de iniciarmos o treino dos modelos iremos apenas colocar aqui todas as funções que serão utilizadas para a vizualização dos resultados:

[skip code]

```
rix
def heatmap(test,pred, model):
```

```
    cm = confusion_matrix(test, pred)
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=["Be
    plt.title(f"Confusion Matrix - {model}")
    plt.xlabel("Predicted Labels")
    plt.ylabel("True Labels")
    plt.figure(figsize=(5, 5))
    plt.show()
    return cm
```

In [33]:
```
# Métricas
def metrics_testing(test,pred, model):
    cm = heatmap(test,pred, model)
    dice_coef = 2 * (cm[1][1]) / (2 * (cm[1][1]) + cm[1][0] + cm[0][1])
    accuracy = accuracy_score(test, pred)
    report = classification_report(test, pred, zero_division=1)

    logloss = log_loss(test, pred)
    # print("Accuracy:", accuracy)
    print("Classification Report:\n", report)
    # print("Log Loss:", logloss)
    # print("Dice Coefficient:", dice_coef)
    return accuracy, logloss, dice_coef
```

In [34]:
```
# Gráficos das métricas - retorna uma lista com as médias das métricas
def plot_metrics(accuracies, log_losses, dice_coefs):
    fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(12, 4))  # Creat

    k = len(accuracies)  # Number of folds

    # Create an array of fold numbers (1, 2, ..., k)
    folds = np.arange(1, k + 1)

    # Plot accuracies
    ax1.bar(folds, accuracies, color='blue', alpha=0.7)
    ax1.set_xticks(folds)  # Set x-ticks to be fold numbers
    ax1.set_xlabel('Fold')
    ax1.set_ylabel('Accuracy')
    ax1.set_title('Accuracy average score: {:.2f}'.format(np.mean(accuracies)))

    # Plot log losses
    ax2.bar(folds, log_losses, color='green', alpha=0.7)
    ax2.set_xticks(folds)  # Set x-ticks to be fold numbers
    ax2.set_xlabel('Fold')
    ax2.set_ylabel('Log Loss')
    ax2.set_title('Log Loss average score: {:.2f}'.format(np.mean(log_losses)))

    # Plot Dice coefficients
    ax3.bar(folds, dice_coefs, color='purple', alpha=0.7)
    ax3.set_xticks(folds)  # Set x-ticks to be fold numbers
    ax3.set_xlabel('Fold')
    ax3.set_ylabel('Dice Coefficient')
    ax3.set_title('Dice Coefficient average score: {:.2f}'.format(np.mean(dice_coef

    # Adjust layout for clarity
    plt.tight_layout()
    plt.show()

    metrics_average = [np.mean(accuracies), np.mean(log_losses), np.mean(dice_coefs
    return metrics_average
```

In [35]:
```
# Comparação e Visualização de resultados
(average_metrics, model_name):
    average_acc, average_logloss, average_dice = zip(*average_metrics)
```

```
    x = np.arange(len(average_metrics))
    width = 0.20

    # Set the style for the plot
    plt.style.use('seaborn-darkgrid')

    fig, ax = plt.subplots(figsize=(8, 6))  # Increased figure size
    bars1 = ax.bar(x - width, average_acc, width, label='Accuracy', color='lightsky
    bars2 = ax.bar(x, average_logloss, width, label='Log Loss', color='yellowgreen'
    bars3 = ax.bar(x + width, average_dice, width, label='Dice Coefficient', color=

    ax.set_ylabel('Scores', fontsize=10)
    ax.set_title(f'Scores for {model_name}', fontsize=16, fontweight='bold')
    ax.set_xticks(x)
    ax.set_xticklabels(['Without Selection', 'Approach PCA', 'Approach T-test', 'Ap
    ax.legend(fontsize=8)

    # Add gridlines
    ax.yaxis.grid(True, linestyle='--', alpha=0.7)

    # Add data labels on bars
    for bar in bars1 + bars2 + bars3:
        yval = bar.get_height()
        ax.text(bar.get_x() + bar.get_width() / 2, yval, round(yval, 2),
                ha='center', va='bottom', fontsize=8, color='black')

    # Set limits for better spacing
    ax.set_ylim(0, max(average_acc + average_logloss + average_dice) * 1.1)

    # Adjust layout
    plt.tight_layout()
    plt.show()
```

[[end skip code]]

Support Vector Machine

[Voltar a Training Models]

O algoritmo Support Vector Machine é frequentemente utilizado na fase de classificação de dados linearmente separáveis. Contudo, é possível adaptar o algoritmo a fim de transformar dados não linearmente separáveis (o nosso caso) em linearmente separáveis. Isto é conseguido através da função de kernel utilizada e escolhida. Com base no estudo LLZ18, concluímos que obteremos uma melhor performance quando escolhemos como função de kernel a função de Radial Basis (RBF).

Estudos que usam SVM na classificação dos nódulos:

- CBMLN
- LLZ18
- TOR23

Algoritmo de Particle Swarm Optimization

Para melhorar a sua performance de classificação do Support Vector Classifier, iremos usar o

ização Particle Swarm Optimization.

O algoritmo de Particle Swarm Optimization é um algoritmo de otimização heurística que se baseia na simulação de um grupo de "partículas" e permite uma mais eficaz procura de solução global para o problema em causa, em detrimento de uma solução local.

Quando associado ao Support Vector Classifier, este algoritmo pode, então, ser utilizado para otimizar hiperparâmetros usados pelo modelo, nomeadamente os parâmetros C (o parâmetro de regularização) e gamma (parâmetro que define a influência de um único ponto de dados). Ao encontrar combinações ideais destes parâmetros, o algoritmo PSO melhora o desempenho da Support Vector Machine, obtendo maior precisão na classificação.

refs.

- LLZ18
- PSO

Código que iremos usar para treinar os modelos:

In [36]:
```
def svm_k_fold(fold, k=5, random_state=42):

    accuracies = []
    log_losses = []
    dice_coefs = []

    # Definir os limites dos parâmetros C e gamma
    lower_bounds = [0.1, 0.001]
    upper_bounds = [100, 1]

    # Função objetivo para otimização
    def objective_function(params):
        C, gamma = params
        fold_accuracies = []


        for X_train, X_test, y_train, y_test in fold:
            model = SVC(C=C, gamma=gamma, kernel='rbf', random_state=random_state)
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            fold_accuracies.append(accuracy_score(y_test, y_pred))

        return -np.mean(fold_accuracies)

    # Algoritmo PSO para otimização
    best_params, _ = pso(objective_function, lower_bounds, upper_bounds)
    best_C, best_gamma = best_params
    print(f"Melhores parâmetros a partir do PSO: C={best_C}, gamma={best_gamma}")

    # K-fold cross-validation com os melhores parâmetros
    n_fold = 1

    for (X_train, X_test, y_train, y_test) in fold:
        print(f"Numero do Fold: {n_fold}")
        model = SVC(C=best_C, gamma=best_gamma, kernel='rbf', random_state=random_s
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)

        acc, logloss, dice = metrics_testing(y_test, predictions, "SVM")

        accuracies.append(acc)
```

```
        log_losses.append(logloss)
        dice_coefs.append(dice)

        n_fold += 1

    return accuracies, log_losses, dice_coefs
```

Lista que vai guardar os resultados dos diferentes testes:

In [37]:
```
results_svm_2d = []
results_svm_3d = []

th_svm_2d = []
th_svm_3d = []
```

Support Vector Machine sem seleção de features

[Voltar a Support Vector Machine]

-> 2D

In [38]:
```
data_2d
```

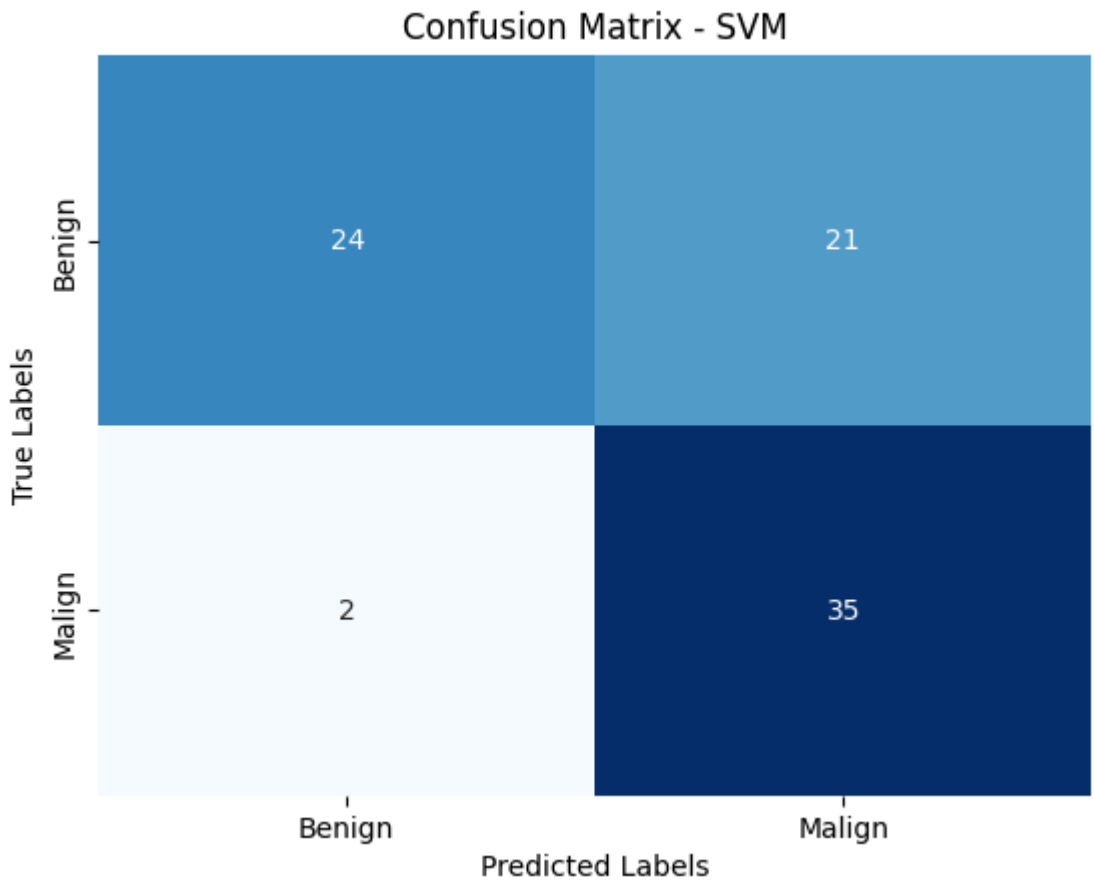
Out[38]:

	Spiculation	Lobulation	Sphericity	Margin	Subtlety	Texture	Calcification	Malignancy	dia ori
0	5	3	3	4	5	5	6	1	
1	2	2	4	3	5	4	6	1	
2	1	1	2	5	2	5	3	0	
3	5	1	4	3	5	5	6	1	
4	1	1	5	5	3	5	3	0	
...	...	...	...	...	...	...	...	...	
405	1	1	3	4	3	5	6	0	
406	2	2	4	4	5	5	6	1	
407	1	3	3	4	5	5	6	1	
408	1	1	3	3	5	5	6	1	
409	1	1	4	2	4	5	6	0	

410 rows × 109 columns

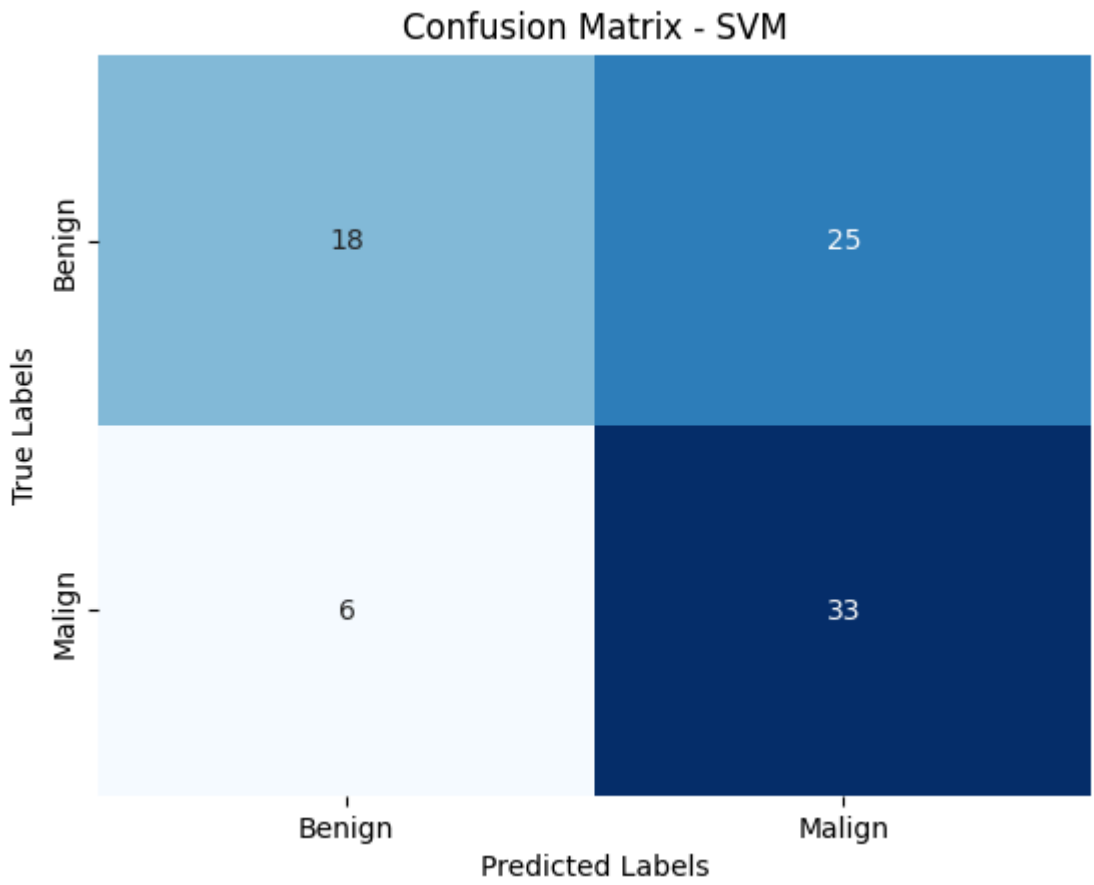
In [39]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = svm_k_fold(fold_2d)
th_svm_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
print(average_metrics_2d)
results_svm_2d.append(average_metrics_2d)
```

```
Stopping search: maximum iterations reached --> 100
Melhores parâmetros a partir do PSO: C=65.85424048297568, gamma=0.0012262975150346
763
Numero do Fold: 1
```

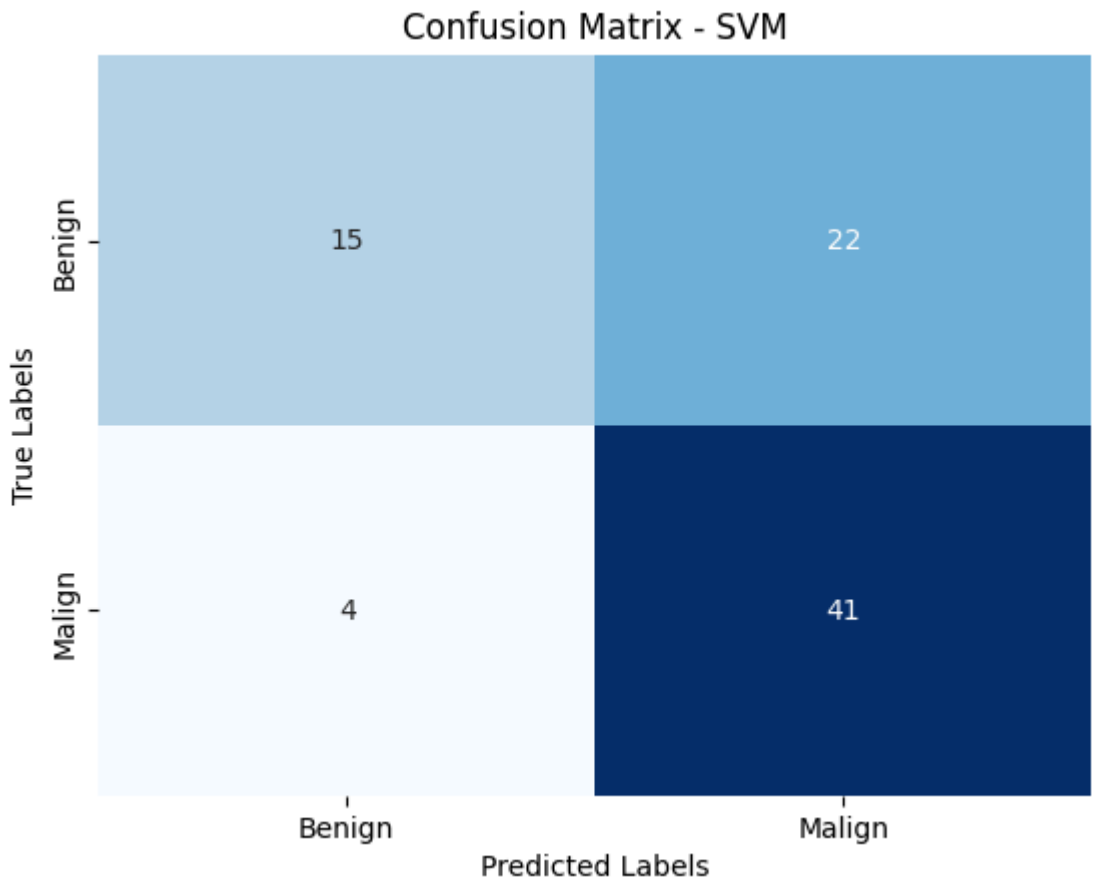



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.53      0.68        45
           1       0.62      0.95      0.75        37

    accuracy                           0.72        82
   macro avg       0.77      0.74      0.71        82
weighted avg       0.79      0.72      0.71        82

Numero do Fold: 2
```

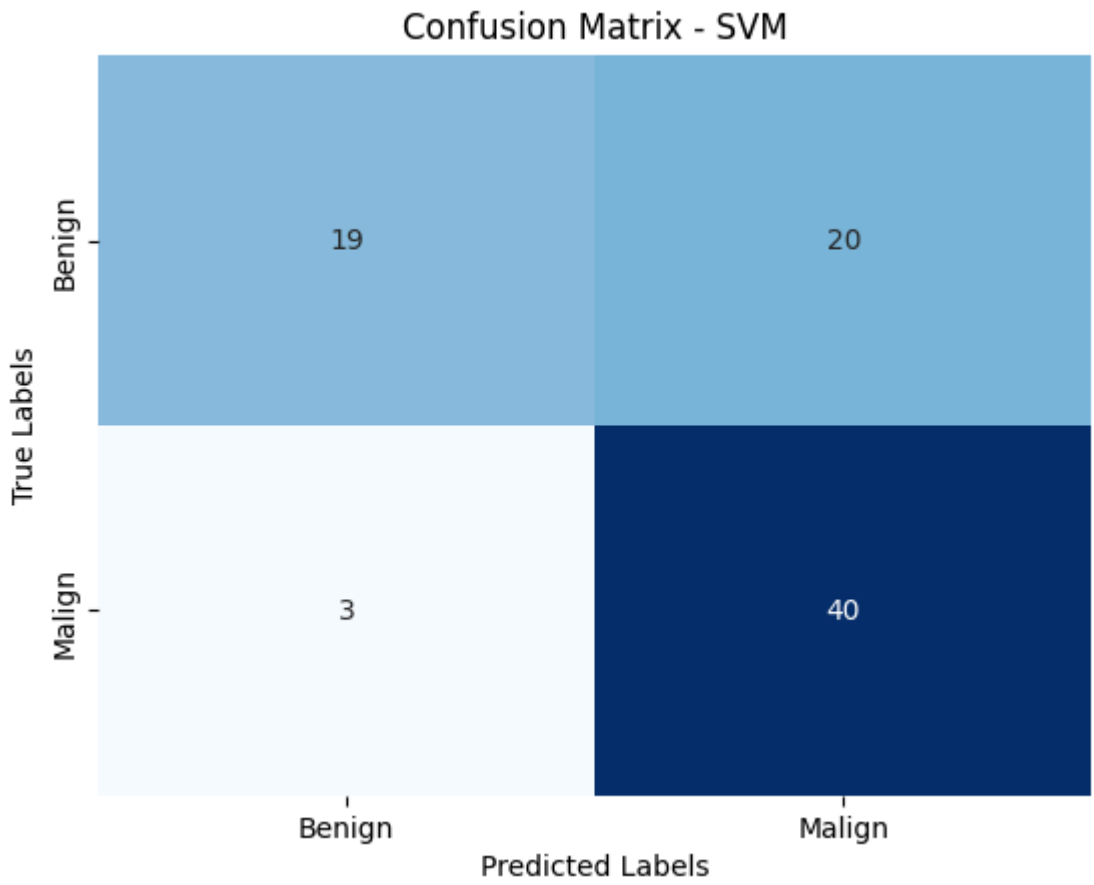
Loading [MathJax]/extensions/Safe.js



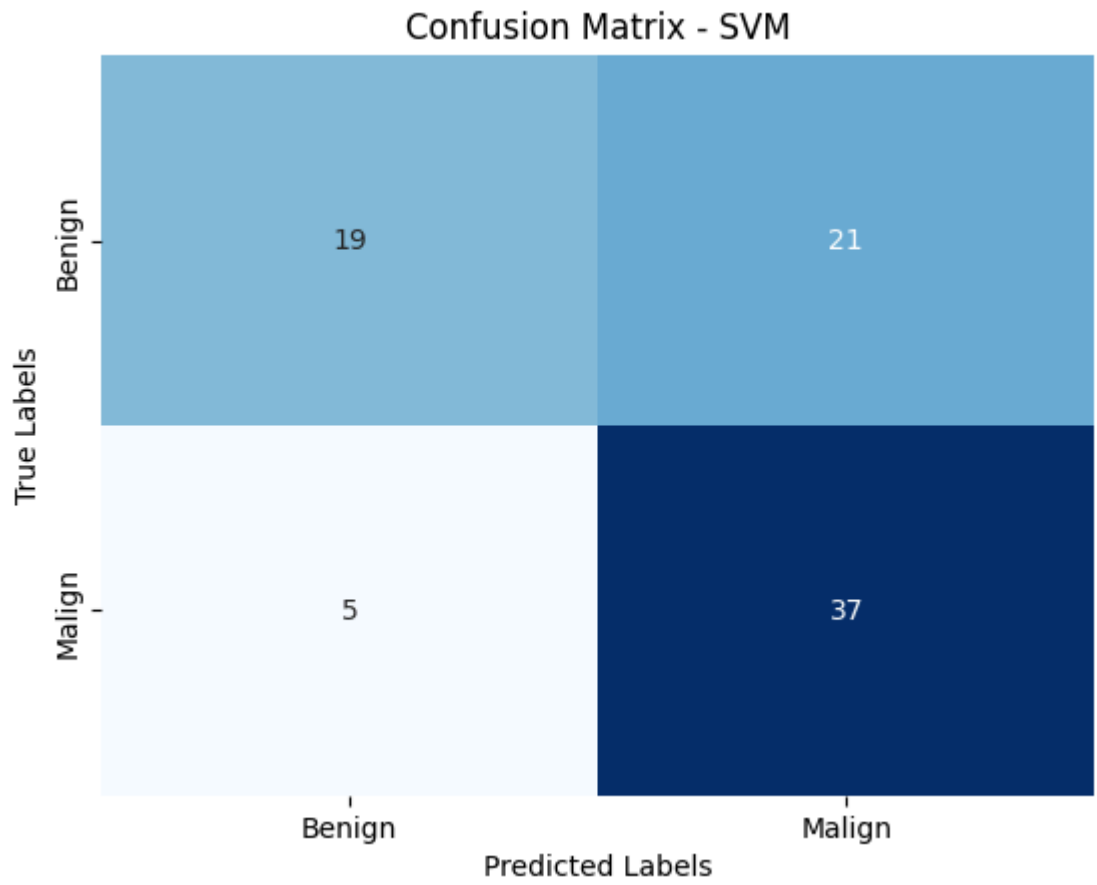

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.42      0.54        43
           1       0.57      0.85      0.68        39

    accuracy                           0.62        82
   macro avg       0.66      0.63      0.61        82
weighted avg       0.66      0.62      0.61        82

Numero do Fold: 3
```



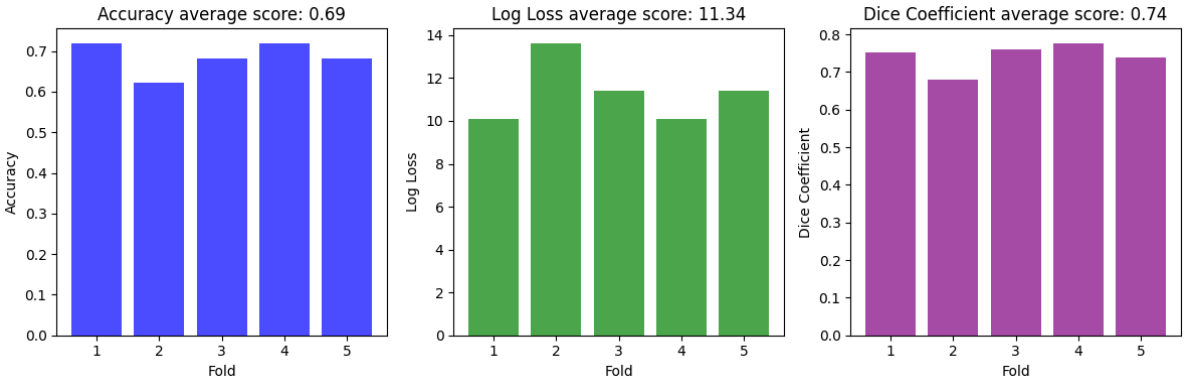

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.41      0.54        37
           1       0.65      0.91      0.76        45

    accuracy                           0.68        82
   macro avg       0.72      0.66      0.65        82
weighted avg       0.71      0.68      0.66        82

Numero do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.49      0.62        39
           1       0.67      0.93      0.78        43

    accuracy                           0.72        82
   macro avg       0.77      0.71      0.70        82
weighted avg       0.76      0.72      0.70        82

Numero do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.47      0.59        40
           1       0.64      0.88      0.74        42

    accuracy                           0.68        82
   macro avg       0.71      0.68      0.67        82
weighted avg       0.71      0.68      0.67        82
```




```
[0.6853658536585366, 11.34056411511247, 0.7418117663125008]
```

-> 3D

In [40]:
```
data_3d
```

Loading [MathJax]/extensions/Safe.js

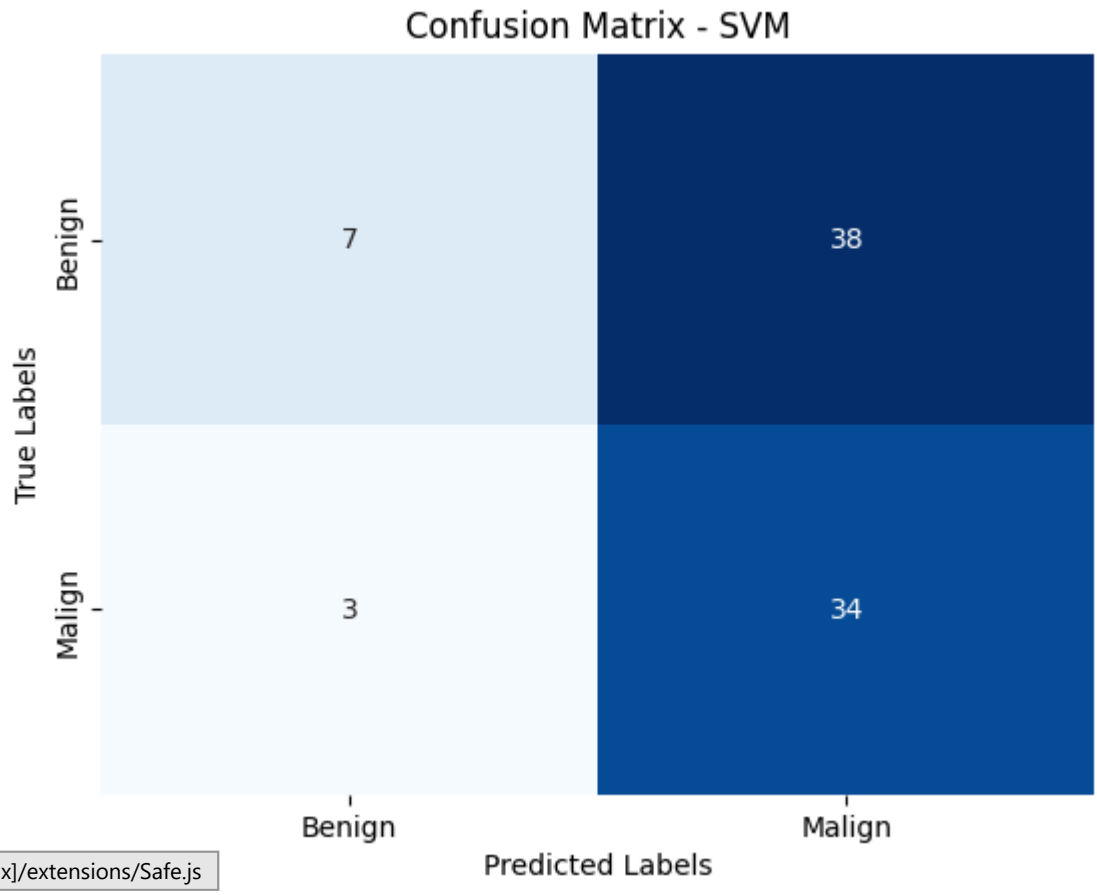
Out[40]:

	Spiculation	Lobulation	Sphericity	Margin	Subtlety	Texture	Calcification	Malignancy	dia ori
0	5	3	3	4	5	5	6	1	
1	2	2	4	3	5	4	6	1	
2	1	1	2	5	2	5	3	0	
3	5	1	4	3	5	5	6	1	
4	1	1	5	5	3	5	3	0	
...	...	...	...	...	...	...	...	...	
405	1	1	3	4	3	5	6	0	
406	2	2	4	4	5	5	6	1	
407	1	3	3	4	5	5	6	1	
408	1	1	3	3	5	5	6	1	
409	1	1	4	2	4	5	6	0	

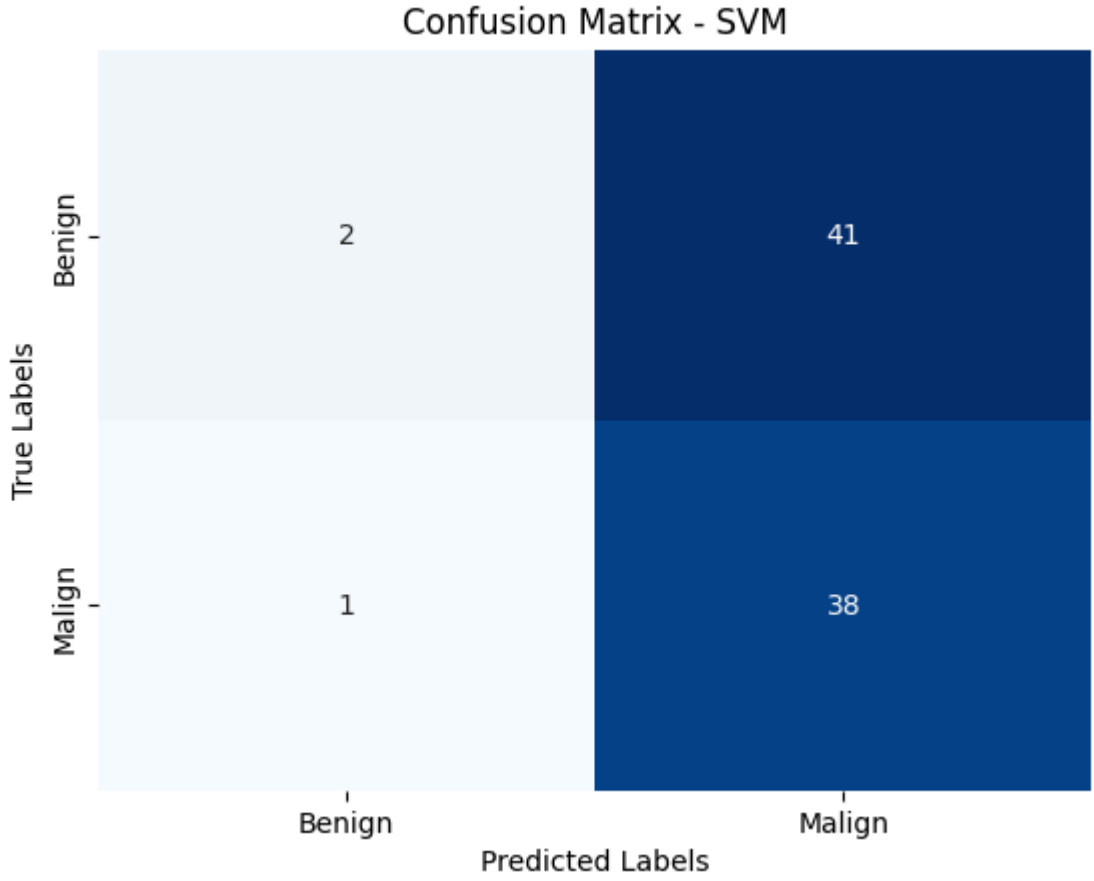
410 rows × 105 columns

In [41]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = svm_k_fold(fold_3d)
th_svm_3d.append([accuracies_3d,log_losses_3d,dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
print(average_metrics_3d)
results_svm_3d.append(average_metrics_3d)
```

```
Stopping search: maximum iterations reached --> 100
Melhores parâmetros a partir do PSO: C=67.01990688250554, gamma=0.001
Numero do Fold: 1
```

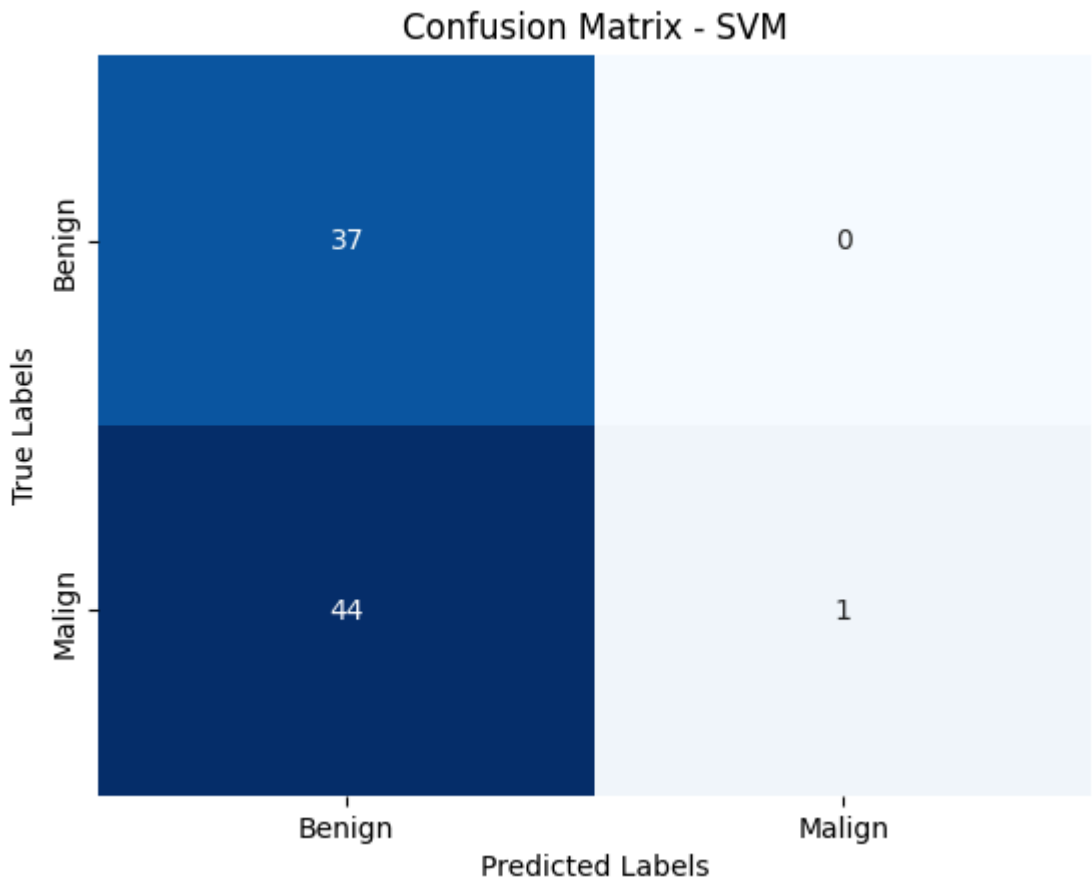


```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.16      0.25        45
           1       0.47      0.92      0.62        37

    accuracy                           0.50        82
   macro avg       0.59      0.54      0.44        82
weighted avg       0.60      0.50      0.42        82

Numero do Fold: 2
```



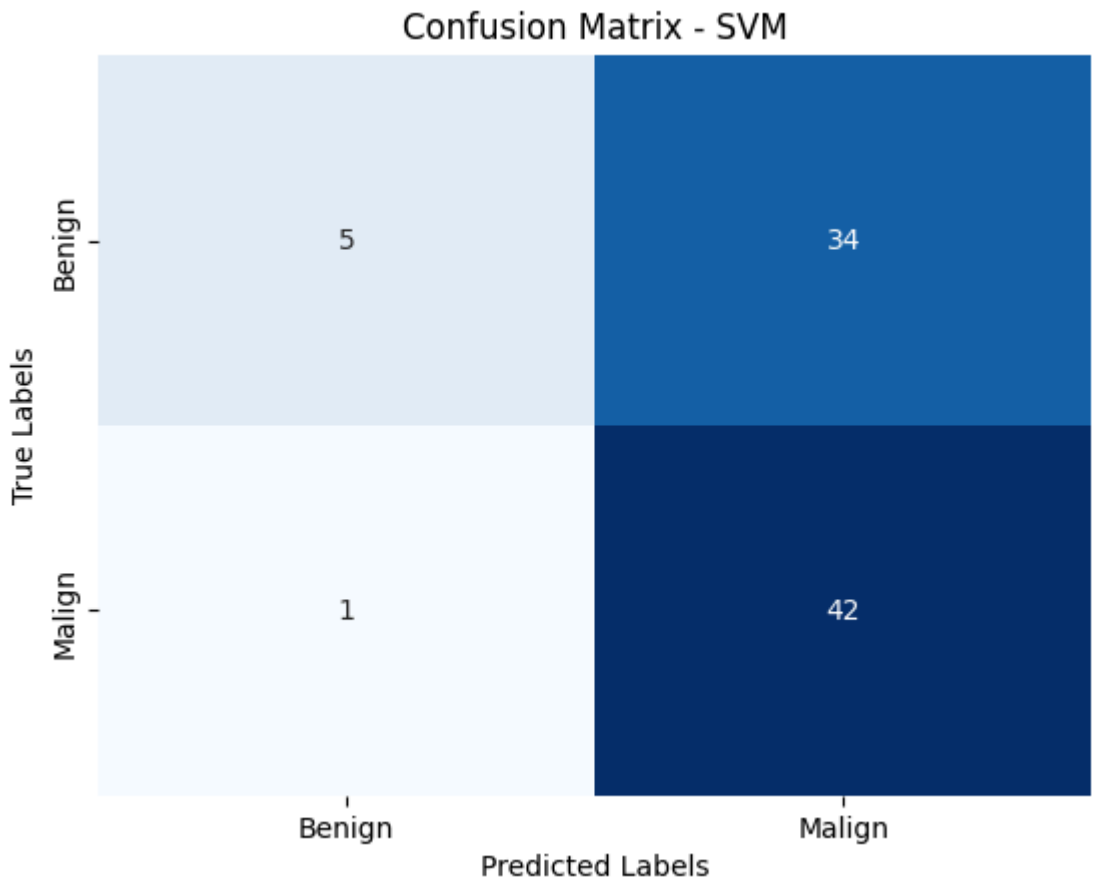

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.05      0.09        43
           1       0.48      0.97      0.64        39

    accuracy                           0.49        82
   macro avg       0.57      0.51      0.37        82
weighted avg       0.58      0.49      0.35        82

Numero do Fold: 3
```

Loading [MathJax]/extensions/Safe.js

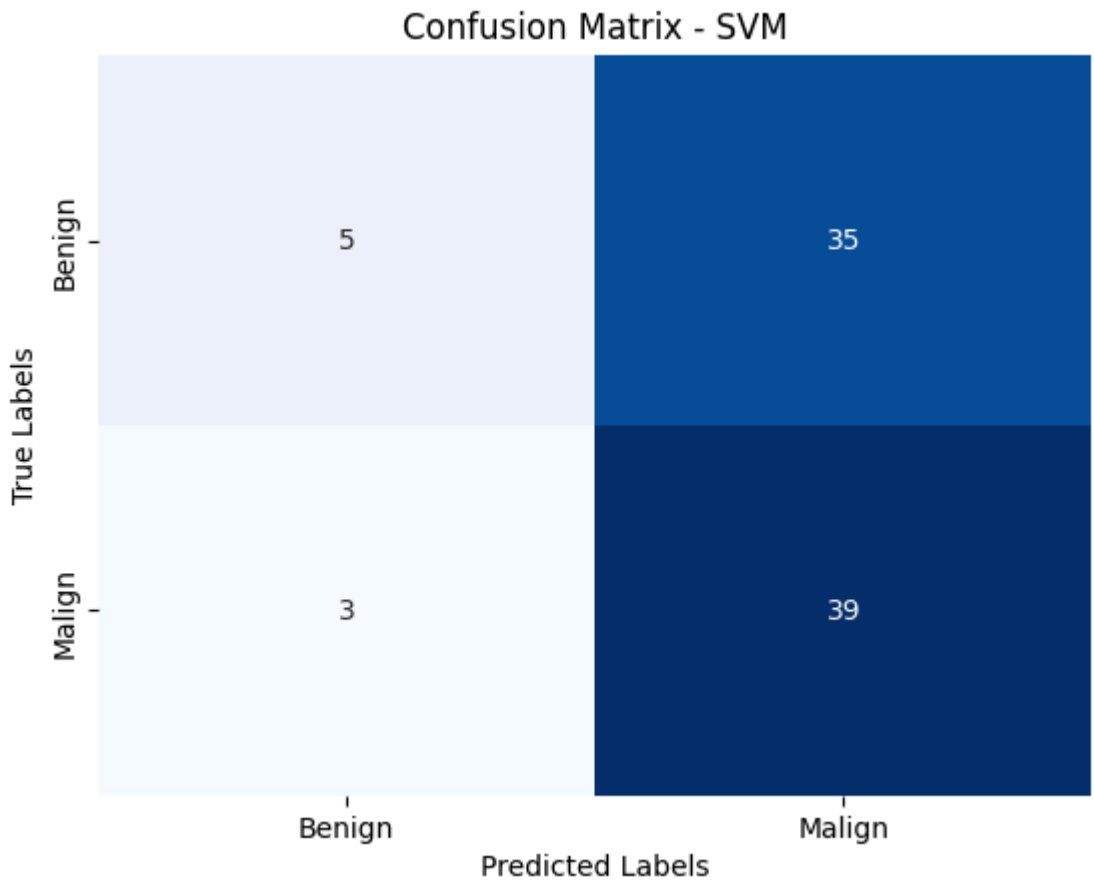



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.46      1.00      0.63        37
           1       1.00      0.02      0.04        45

    accuracy                           0.46        82
   macro avg       0.73      0.51      0.34        82
weighted avg       0.75      0.46      0.31        82

Numero do Fold: 4
```

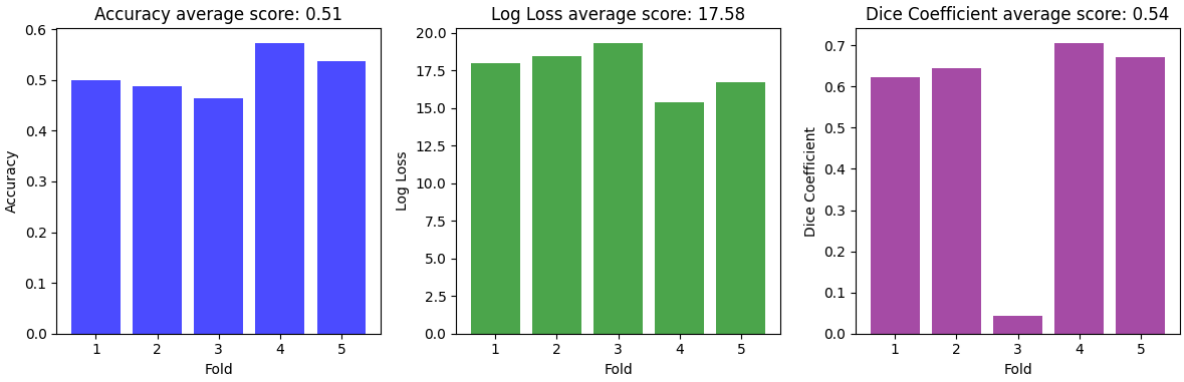
Loading [MathJax]/extensions/Safe.js




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.13      0.22        39
           1       0.55      0.98      0.71        43

    accuracy                           0.57        82
   macro avg       0.69      0.55      0.46        82
weighted avg       0.69      0.57      0.48        82

Numero do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.12      0.21        40
           1       0.53      0.93      0.67        42

    accuracy                           0.54        82
   macro avg       0.58      0.53      0.44        82
weighted avg       0.57      0.54      0.45        82
```




```
[0.5121951219512195, 17.582269945910802, 0.5379390829067068]
```

Support Vector Machine + PCA

[Voltar a Support Vector Machine]

Iremos agora passar à implementação do modelo SVM, utilizando o algoritmo PSO e preparado a partir do método de Principal Component Analysis.

-> 2D

In [42]:
```
data_pca_2d
```

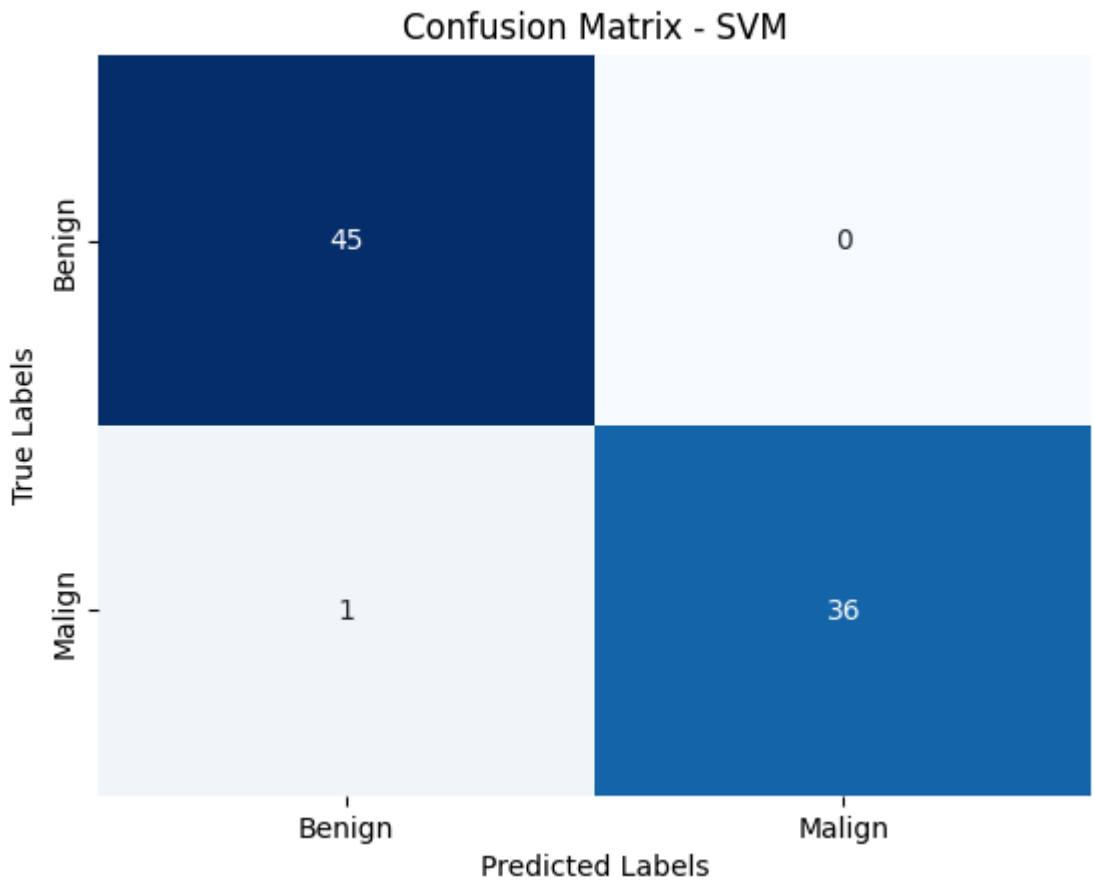
Out[42]:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	P
0	0.605527	14.312994	2.618862	-5.010918	-1.873619	0.486000	3.780943	4.509357	-3.0005
1	-2.267379	-1.819947	0.040090	0.846899	-1.514118	-0.992908	-0.811545	0.548039	-0.1913
2	5.058170	-2.480426	-3.601613	-4.459463	-1.977084	-3.833077	1.266602	3.487037	0.1742
3	26.065083	6.727777	-6.985184	4.449976	-1.804223	1.655932	-2.786482	-3.047505	1.8274
4	-2.158470	-3.958437	-0.673922	0.104410	1.383675	-1.097692	0.958196	-0.653246	0.2457
...	...	...	...	...	...	...	...	...	
405	-2.233398	-2.998710	-0.378282	0.497056	-0.144041	-1.423559	-0.648699	0.289676	0.1224
406	-2.923210	11.711044	5.856770	4.536946	0.801008	-0.007470	1.331167	-1.196007	0.6149
407	-2.716925	5.627624	3.129216	2.034017	-1.944248	1.328403	0.538597	-0.491240	0.2763
408	-2.511517	3.152825	2.056172	1.597862	-1.588072	0.221263	0.574015	-0.519261	0.3861
409	-2.493756	-2.801108	-0.163597	-0.006073	0.120255	1.392496	-1.301845	0.218497	-0.2664

410 rows × 18 columns

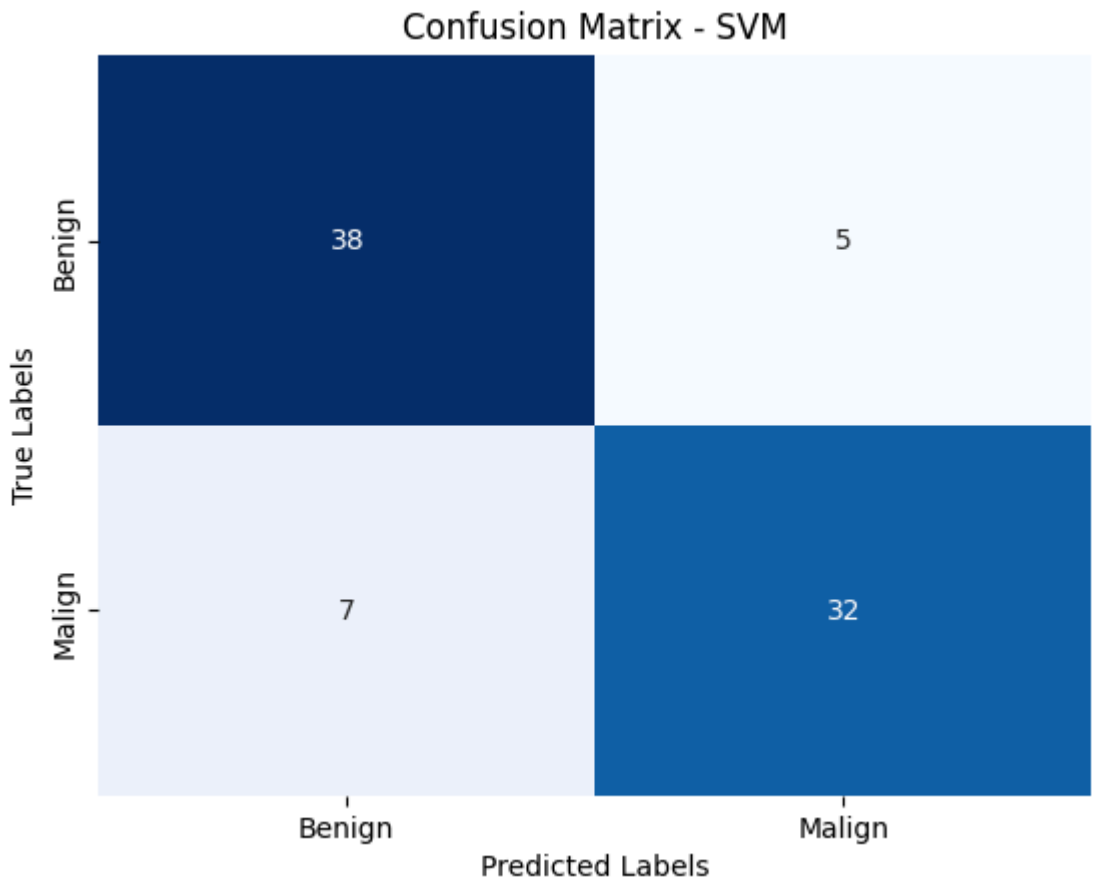
In [43]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = svm_k_fold(fold_2d_pca)
th_svm_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
print(average_metrics_2d)
results_svm_2d.append(average_metrics_2d)
```

```
Stopping search: maximum iterations reached --> 100
Melhores parâmetros a partir do PSO: C=21.47893224976315, gamma=0.001
Numero do Fold: 1
```

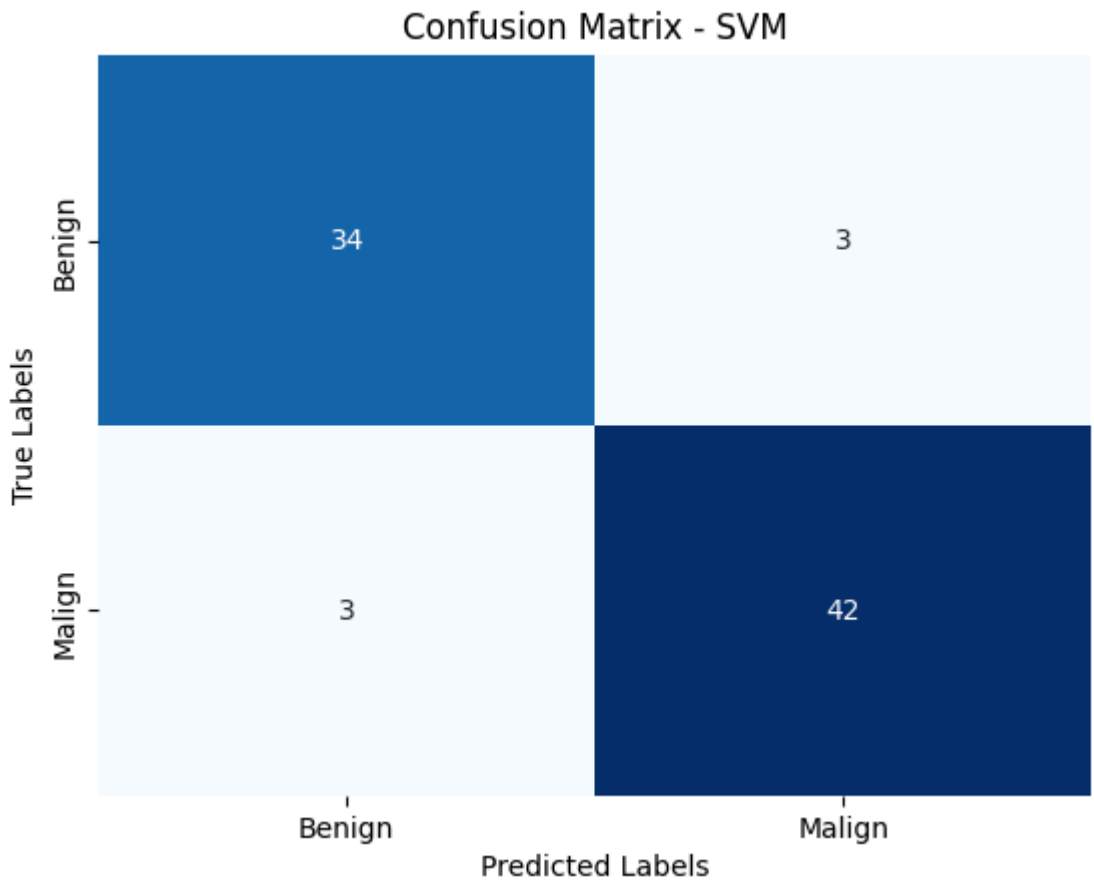



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       1.00      0.97      0.99        37

    accuracy                           0.99        82
   macro avg       0.99      0.99      0.99        82
weighted avg       0.99      0.99      0.99        82

Numero do Fold: 2
```

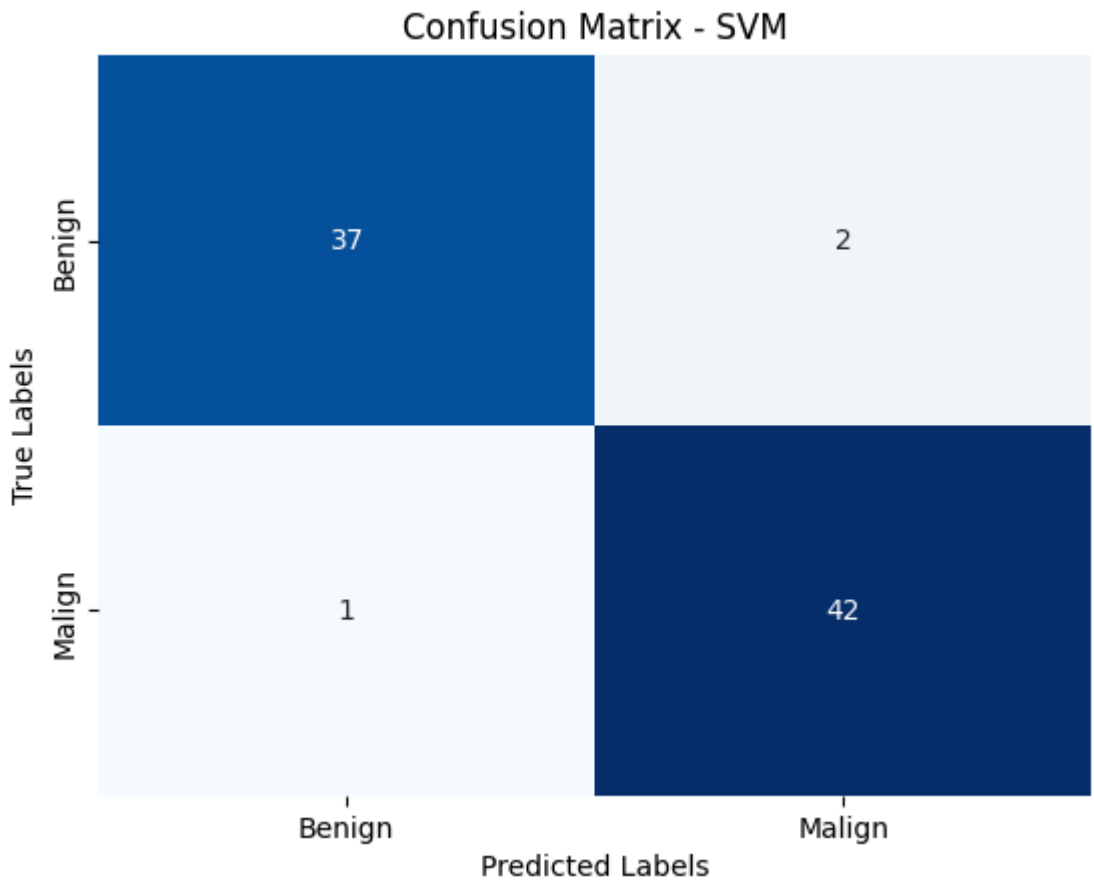
Loading [MathJax]/extensions/Safe.js



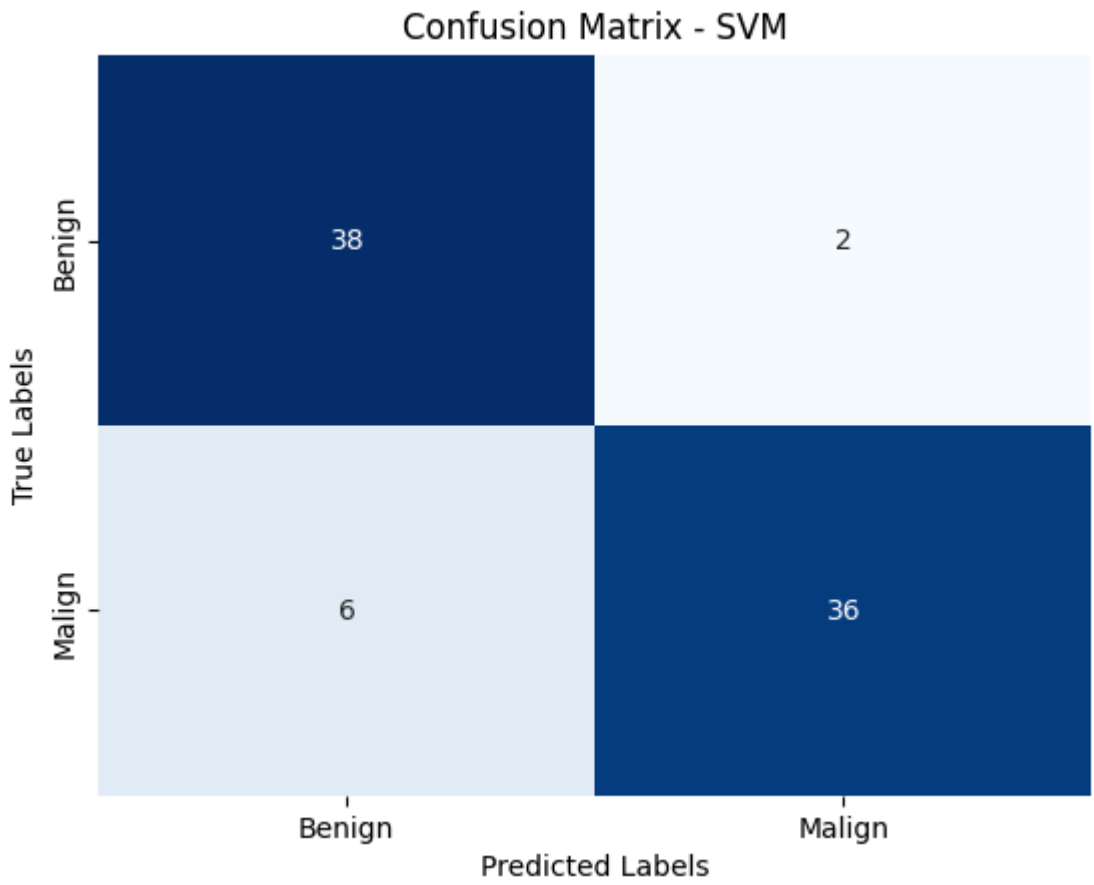

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.88      0.86        43
           1       0.86      0.82      0.84        39

    accuracy                           0.85        82
   macro avg       0.85      0.85      0.85        82
weighted avg       0.85      0.85      0.85        82

Numero do Fold: 3
```



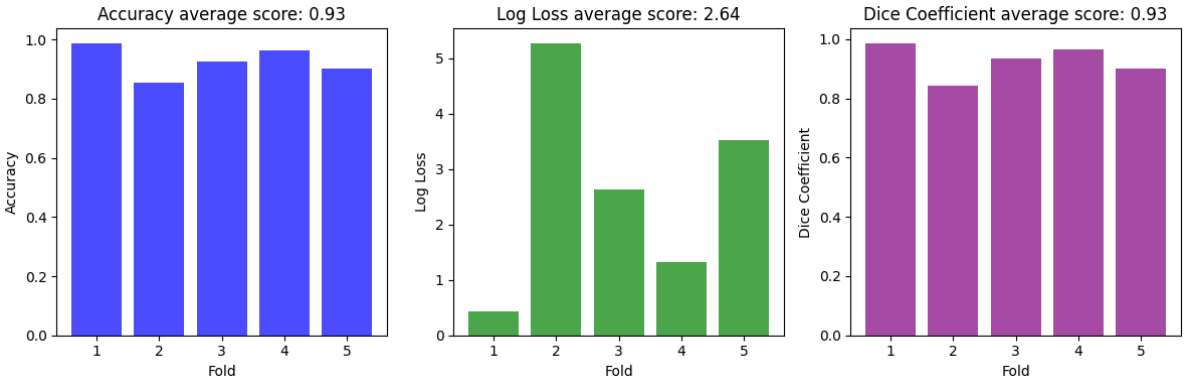

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.92      0.92        37
           1       0.93      0.93      0.93        45

    accuracy                           0.93        82
   macro avg       0.93      0.93      0.93        82
weighted avg       0.93      0.93      0.93        82

Numero do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        39
           1       0.95      0.98      0.97        43

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Numero do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.95      0.90        40
           1       0.95      0.86      0.90        42

    accuracy                           0.90        82
   macro avg       0.91      0.90      0.90        82
weighted avg       0.91      0.90      0.90        82
```




```
[0.9268292682926829, 2.637340491886621, 0.9254514415467104]
```

-> 3D

In [44]:
```
data_pca_3d
```

Loading [MathJax]/extensions/Safe.js

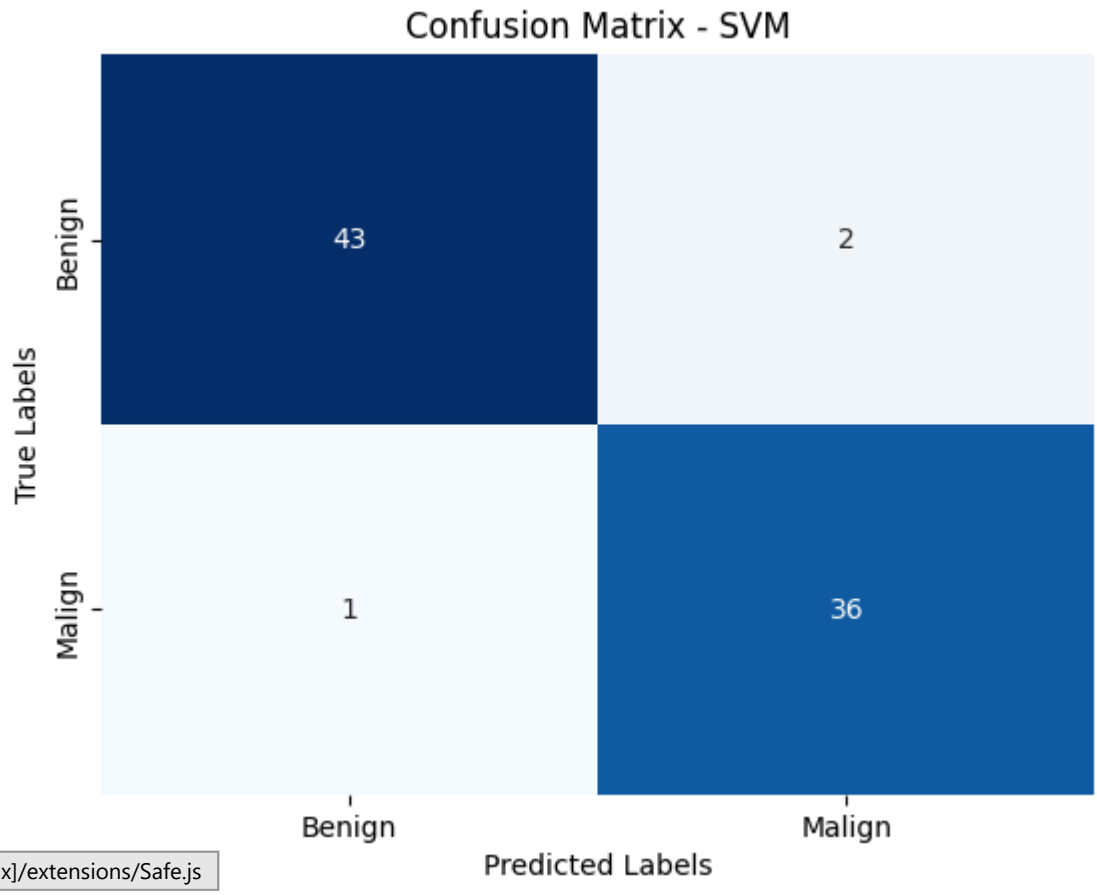
Out[44]:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	P
0	-0.782122	-4.117717	1.807559	3.238631	-2.126003	0.470964	0.041700	-1.212518	-0.2280
1	-2.459356	2.218242	-1.818623	2.089689	0.356737	1.117322	-0.424158	-0.750689	-0.4551
2	13.960910	0.629641	-1.754254	-1.768815	-1.563777	-2.868636	-2.049989	-1.101771	-1.3600
3	18.196420	-6.671389	-6.882279	1.530733	2.834788	2.809031	-2.197428	2.412911	1.9639
4	-2.639728	3.325768	-2.169896	0.994315	-0.182417	-0.915270	1.220950	1.471481	0.1514
...	...	...	...	...	...	...	...	...	
405	-2.682271	3.198533	-2.295293	1.549300	0.057934	-0.301314	0.212454	-0.055443	0.9605
406	-2.269498	-5.455861	4.720836	7.225682	2.834643	-3.934610	3.135748	-0.186134	0.6010
407	-2.505729	-0.167273	0.442583	2.612708	2.009912	0.850195	-0.439097	0.373260	-1.7192
408	-2.686433	0.246159	0.531057	1.261710	1.665685	0.842926	-0.394356	0.287286	-1.0636
409	-2.196771	3.355261	-3.609834	4.514218	-0.464621	0.370936	0.322757	-0.490927	1.1526

410 rows × 18 columns

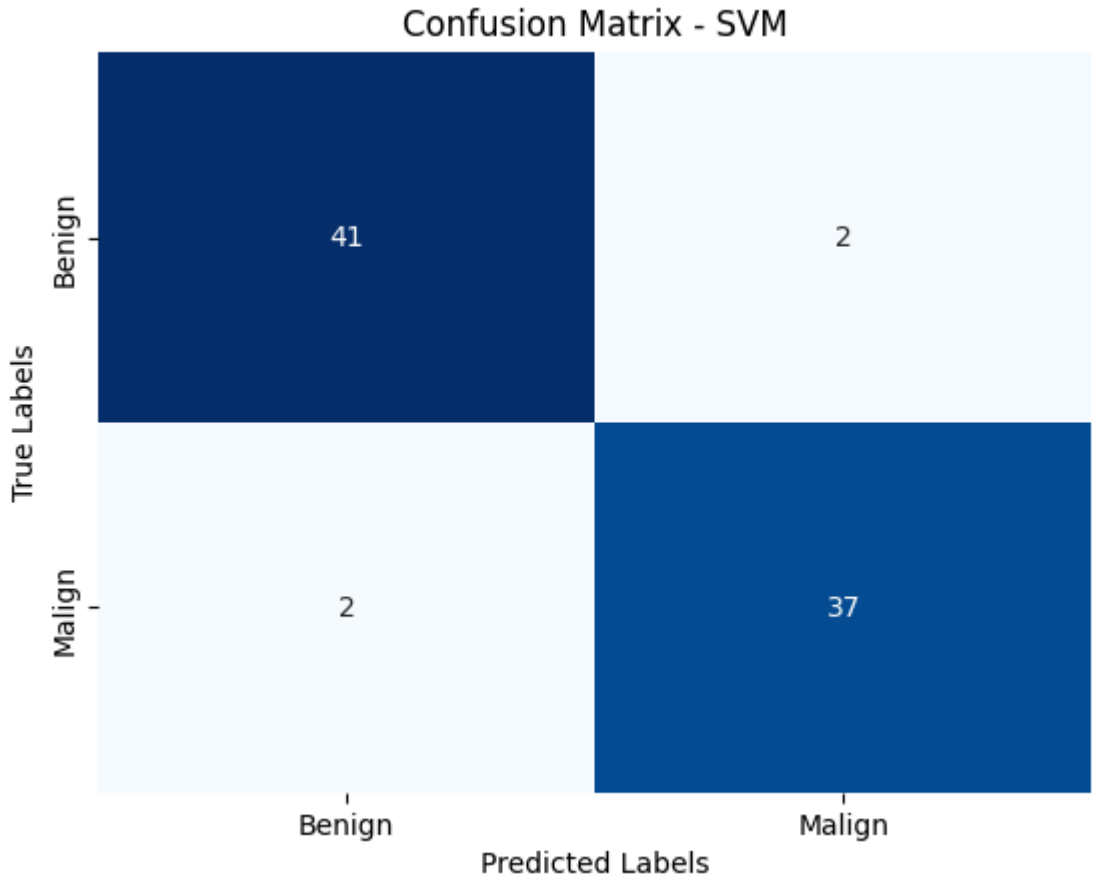
In [45]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = svm_k_fold(fold_3d_pca)
th_svm_3d.append([accuracies_3d,log_losses_3d,dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
print(average_metrics_3d)
results_svm_3d.append(average_metrics_3d)
```

```
Stopping search: maximum iterations reached --> 100
Melhores parâmetros a partir do PSO: C=43.34708969629817, gamma=0.0029508670192083
15
Numero do Fold: 1
```



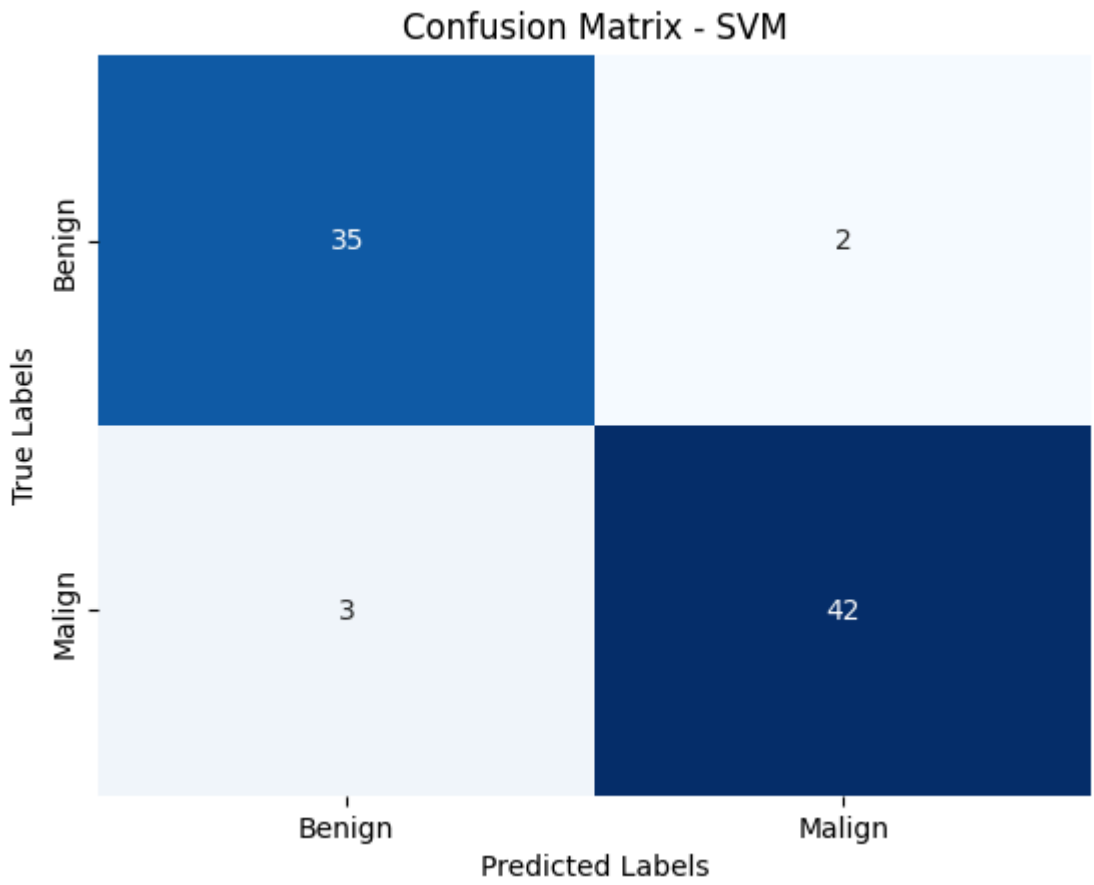
Loading [MathJax]/extensions/Safe.js

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        45
           1       0.95      0.97      0.96        37

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Numero do Fold: 2
```



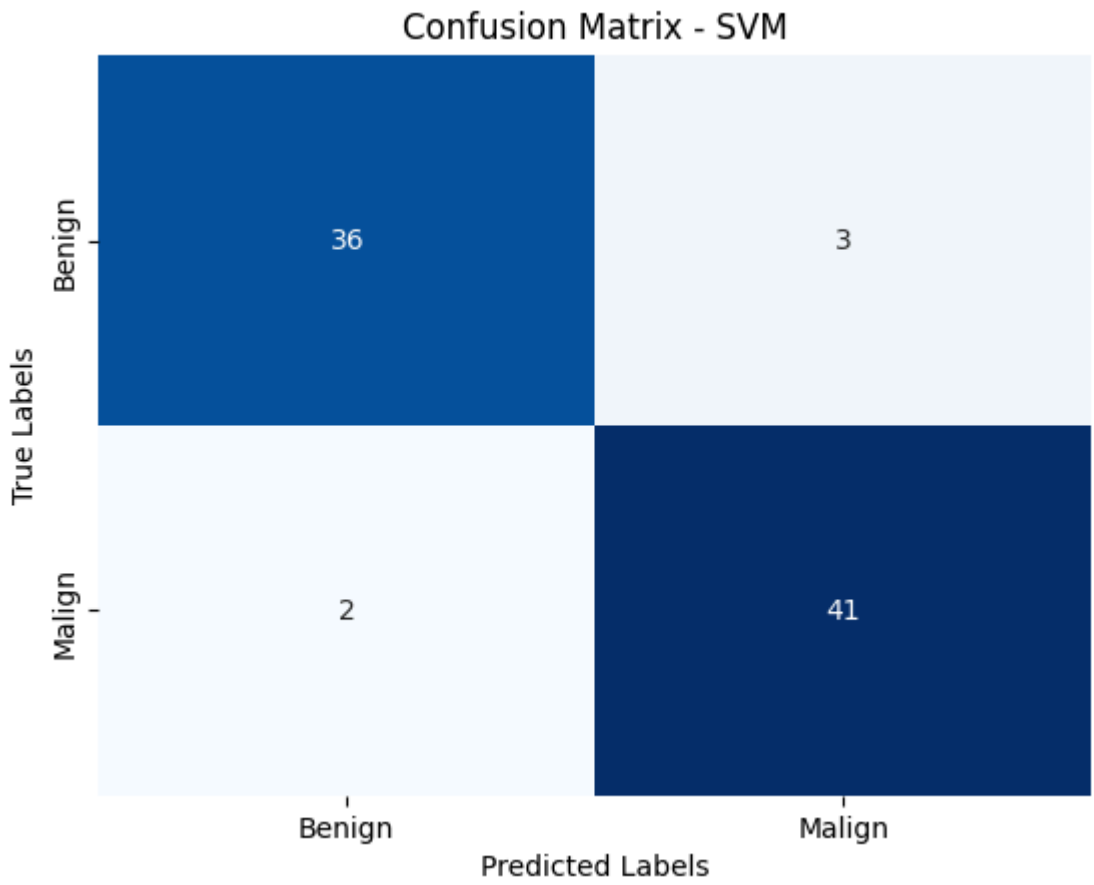

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        43
           1       0.95      0.95      0.95        39

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Numero do Fold: 3
```

Loading [MathJax]/extensions/Safe.js

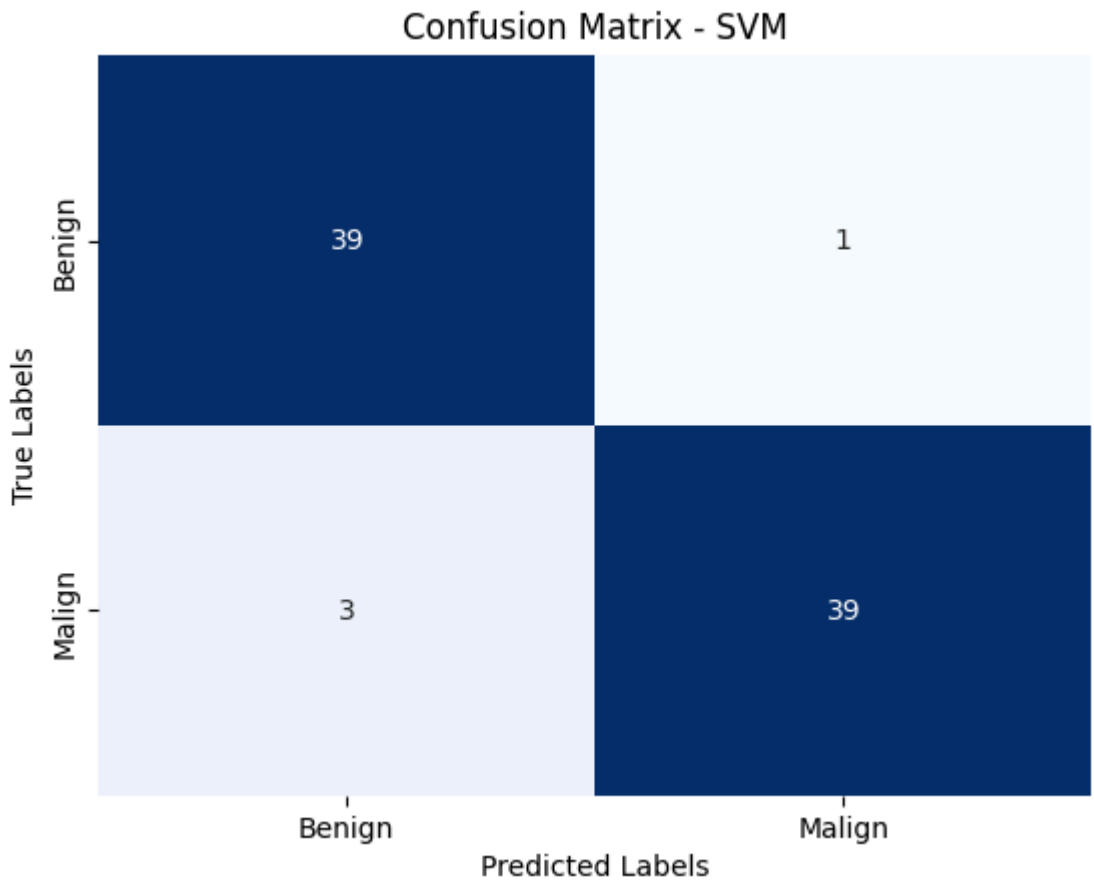



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.95      0.93        37
           1       0.95      0.93      0.94        45

    accuracy                           0.94        82
   macro avg       0.94      0.94      0.94        82
weighted avg       0.94      0.94      0.94        82

Numero do Fold: 4
```

Loading [MathJax]/extensions/Safe.js




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.92      0.94        39
           1       0.93      0.95      0.94        43

    accuracy                           0.94        82
   macro avg       0.94      0.94      0.94        82
weighted avg       0.94      0.94      0.94        82

Numero do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95        40
           1       0.97      0.93      0.95        42

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82
```


```
[0.9487804878048781, 1.8461383443206347, 0.949257284252871]
```

Support Vector Machine + t-test

[Voltar a Support Vector Machine]

Podemos repetir todo o processo usando agora o dataset obtido pela aplicação do teste de para selecionar as features mais significantes.

-> 2D

In [46]:
```
data_ttest_2d
```

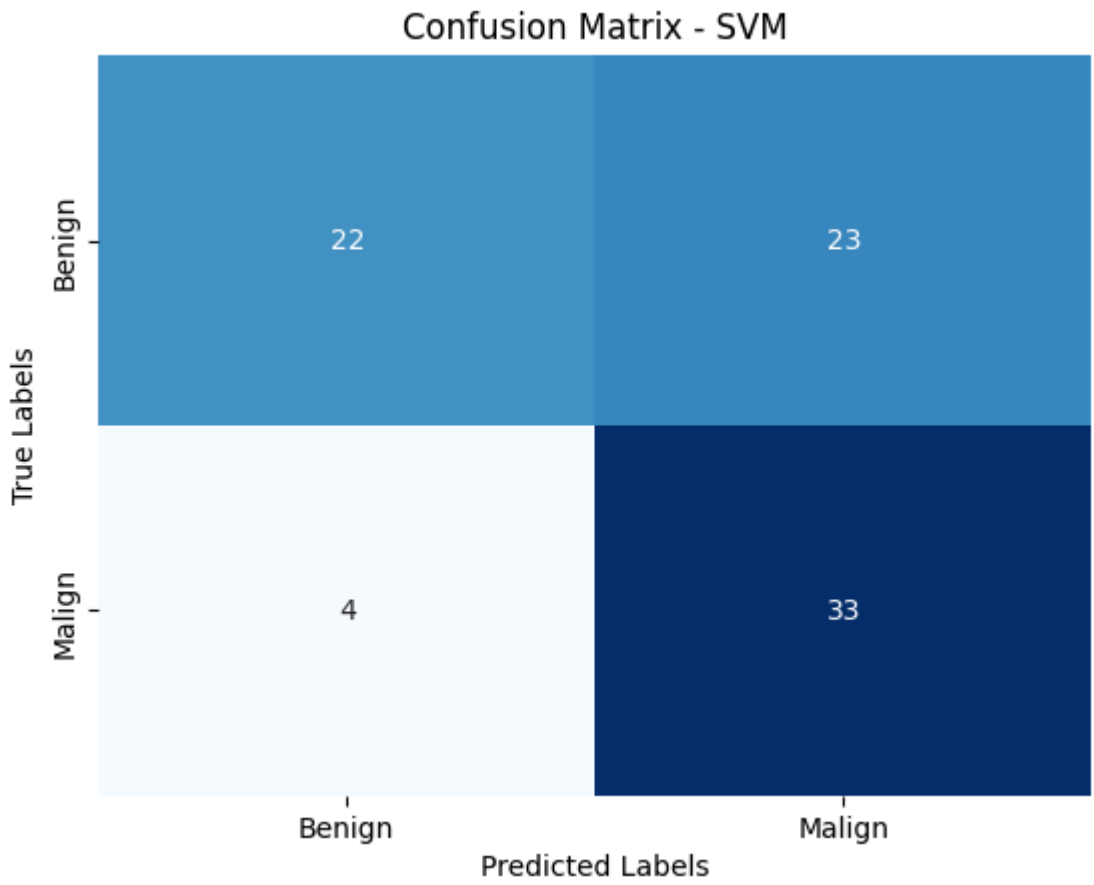
Out[46]:

	Calcification	original_shape2D_Sphericity	original_shape2D_MajorAxisLength	original_shape2D
0	6	0.116476	57.542760	
1	6	0.237705	17.090541	
2	3	0.293095	9.244896	
3	6	0.145259	40.718270	
4	3	0.288620	10.327956	
...	...	...	...	
405	6	0.245244	13.852837	
406	6	0.112044	61.575217	
407	6	0.138573	51.164214	
408	6	0.154955	41.452617	
409	6	0.276215	12.494295	

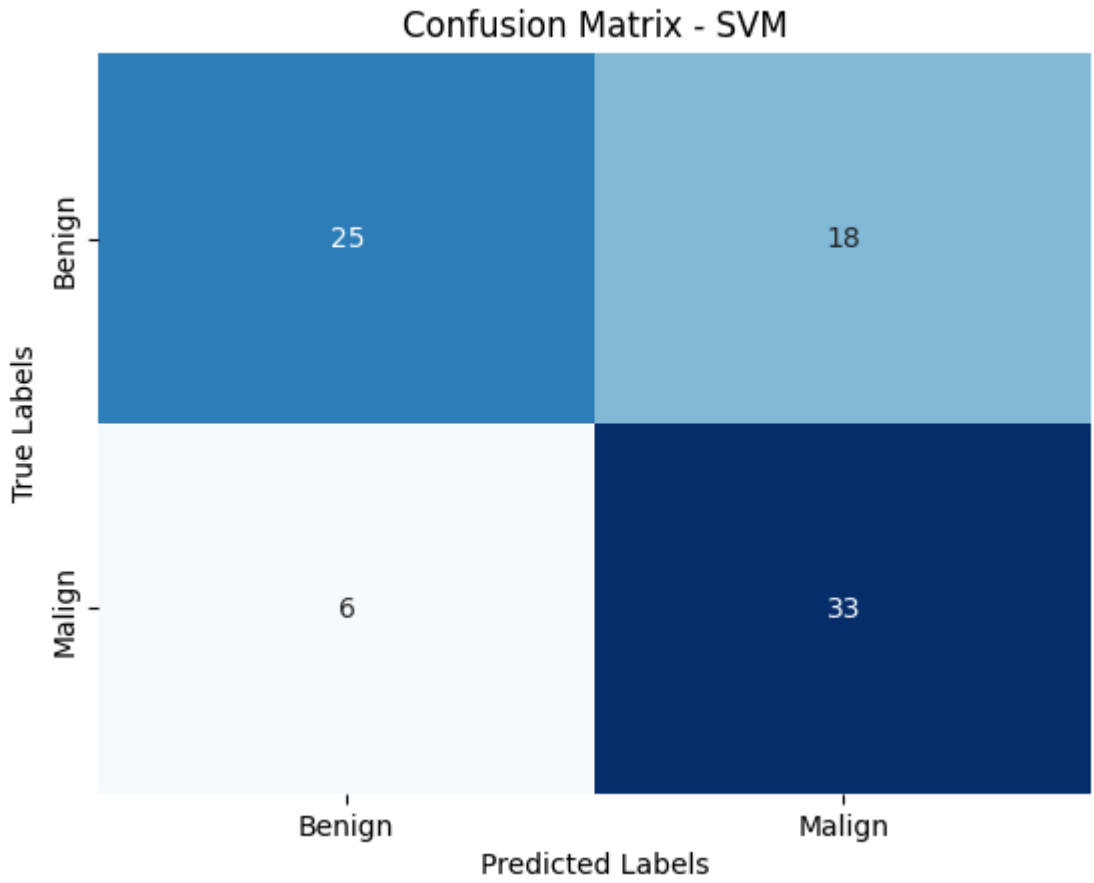
410 rows × 49 columns

In [47]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = svm_k_fold(fold_2d_ttest)
th_svm_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
print(average_metrics_2d)
results_svm_2d.append(average_metrics_2d)
```

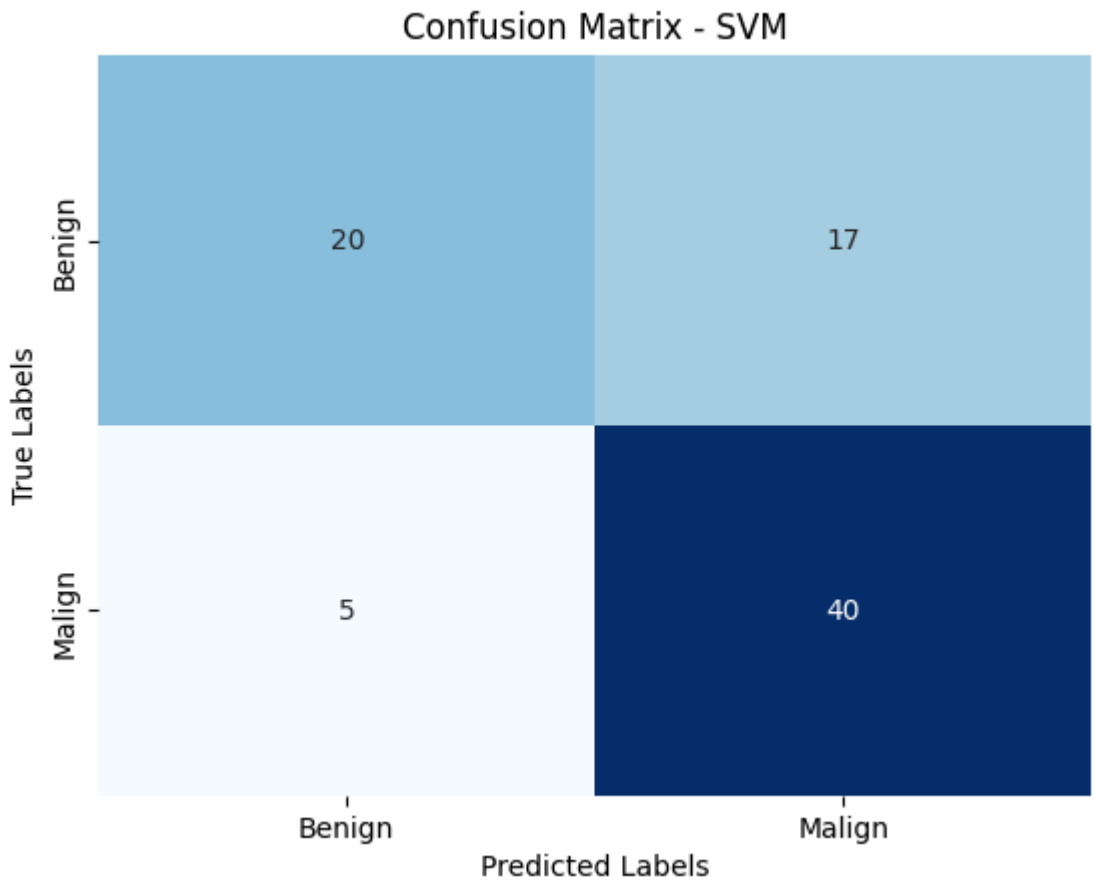
```
Stopping search: maximum iterations reached --> 100
Melhores parâmetros a partir do PSO: C=10.28815672204144, gamma=0.0194286306847707
6
Numero do Fold: 1
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.49      0.62        45
           1       0.59      0.89      0.71        37

    accuracy                           0.67        82
   macro avg       0.72      0.69      0.66        82
weighted avg       0.73      0.67      0.66        82

Numero do Fold: 2
```

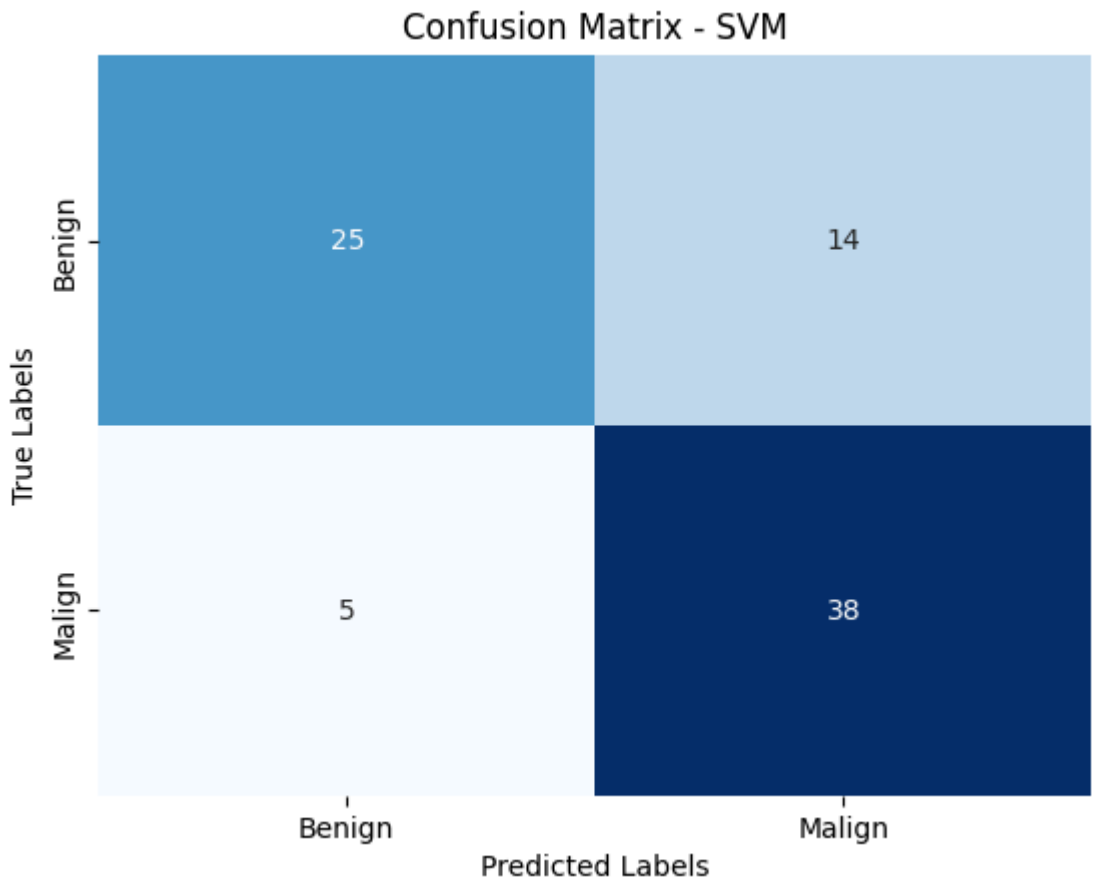



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.58      0.68        43
           1       0.65      0.85      0.73        39

    accuracy                           0.71        82
   macro avg       0.73      0.71      0.70        82
weighted avg       0.73      0.71      0.70        82

Numero do Fold: 3
```



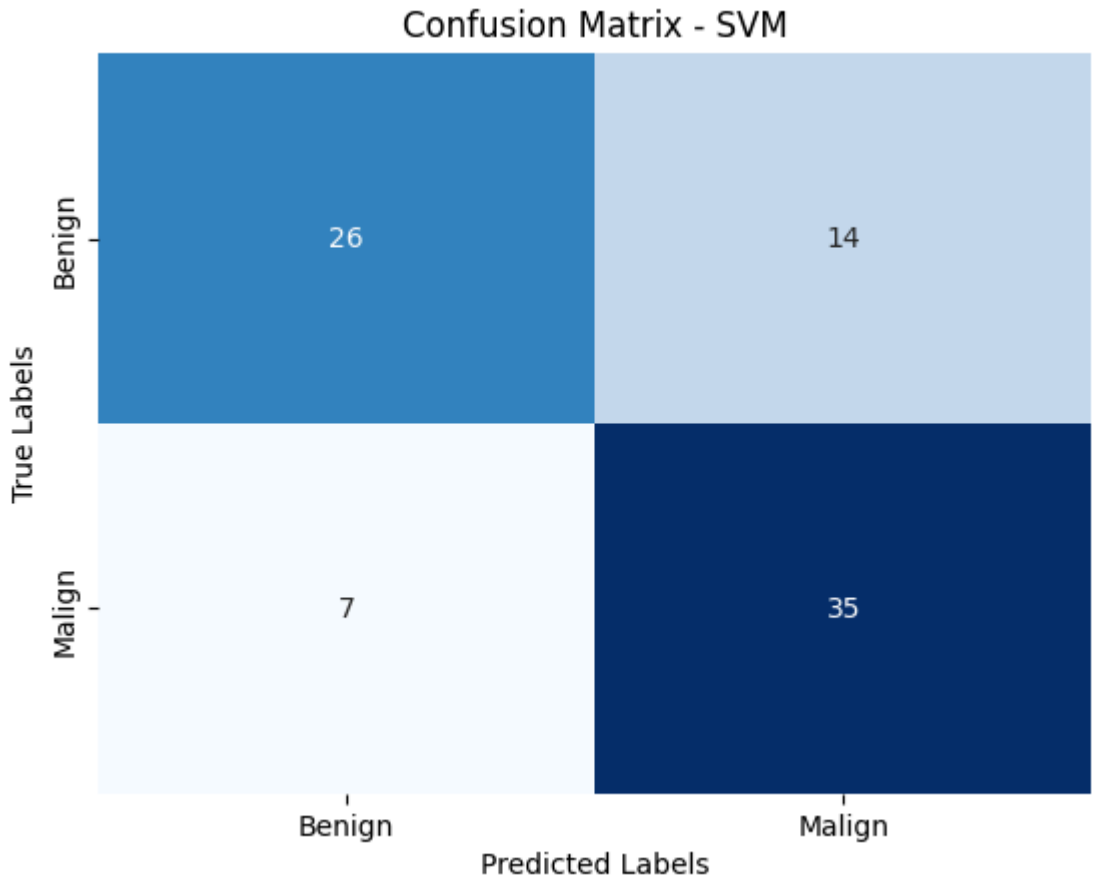

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.54      0.65        37
           1       0.70      0.89      0.78        45

    accuracy                           0.73        82
   macro avg       0.75      0.71      0.71        82
weighted avg       0.75      0.73      0.72        82

Numero do Fold: 4
```

Loading [MathJax]/extensions/Safe.js

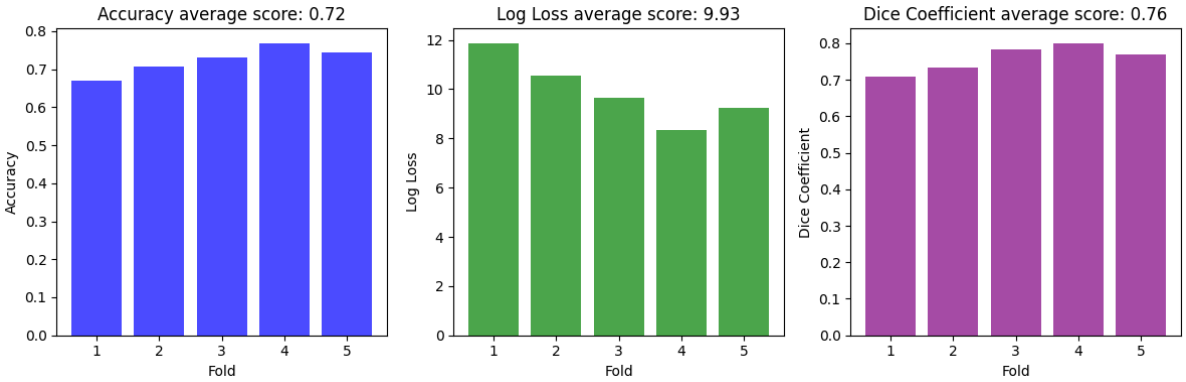



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.64      0.72        39
           1       0.73      0.88      0.80        43

    accuracy                           0.77        82
   macro avg       0.78      0.76      0.76        82
weighted avg       0.78      0.77      0.76        82

Numero do Fold: 5
```

Loading [MathJax]/extensions/Safe.js




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.65      0.71        40
           1       0.71      0.83      0.77        42

    accuracy                           0.74        82
   macro avg       0.75      0.74      0.74        82
weighted avg       0.75      0.74      0.74        82
```




```
[0.724390243902439, 9.933982519439606, 0.7593110494818275]
```

-> 3D

In [48]:
```
data_ttest_3d
```

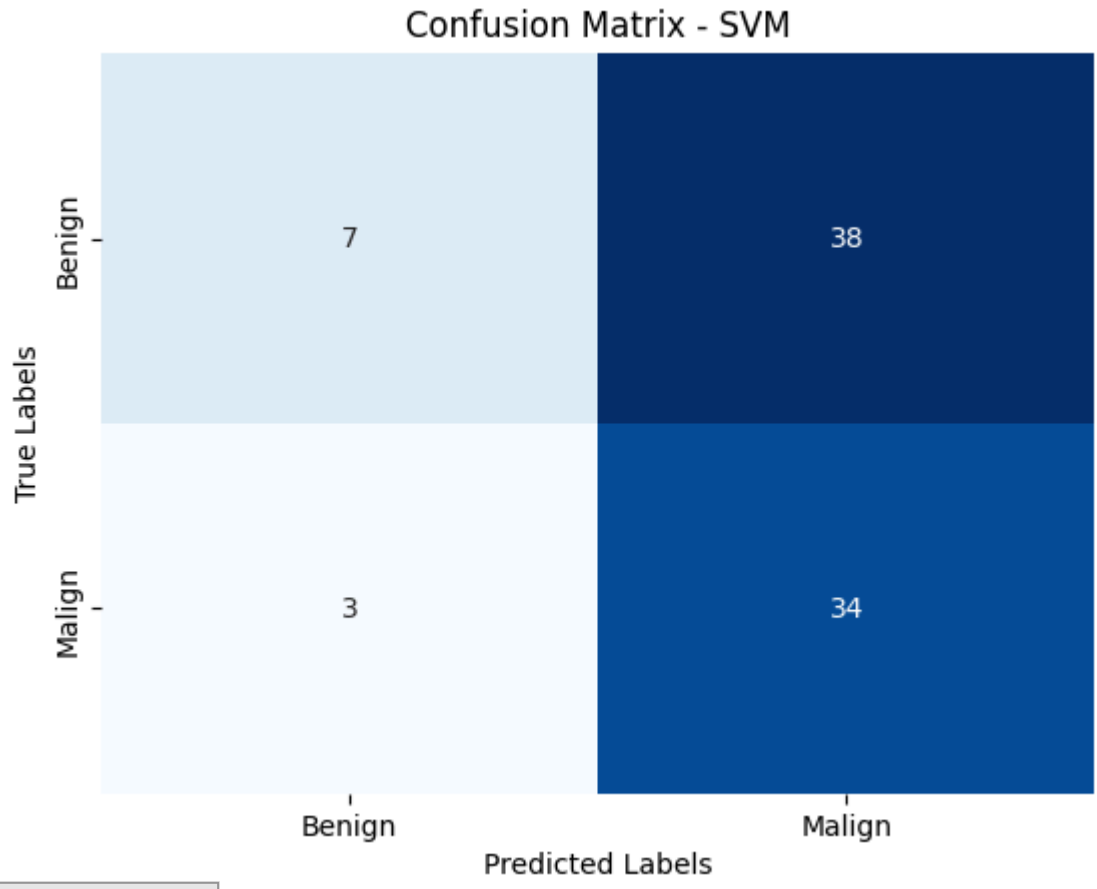
Out[48]:

	Calcification	Spiculation	Lobulation	Margin	Subtlety	original_glrlm_GrayLevelNonUniformit
0	6	5	3	4	5	757.88036
1	6	2	2	3	5	88.38461
2	3	1	1	5	2	34.51526
3	6	5	1	3	5	437.07015
4	3	1	1	5	3	33.84615
...	...	...	...	...	...	
405	6	1	1	4	3	41.76923
406	6	2	2	4	5	1239.76923
407	6	1	3	4	5	573.30769
408	6	1	1	3	5	444.76923
409	6	1	1	2	4	47.84615

410 rows × 58 columns

In [49]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = svm_k_fold(fold_3d_ttest)
th_svm_3d.append([accuracies_3d,log_losses_3d,dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
print(average_metrics_3d)
results_svm_3d.append(average_metrics_3d)
```

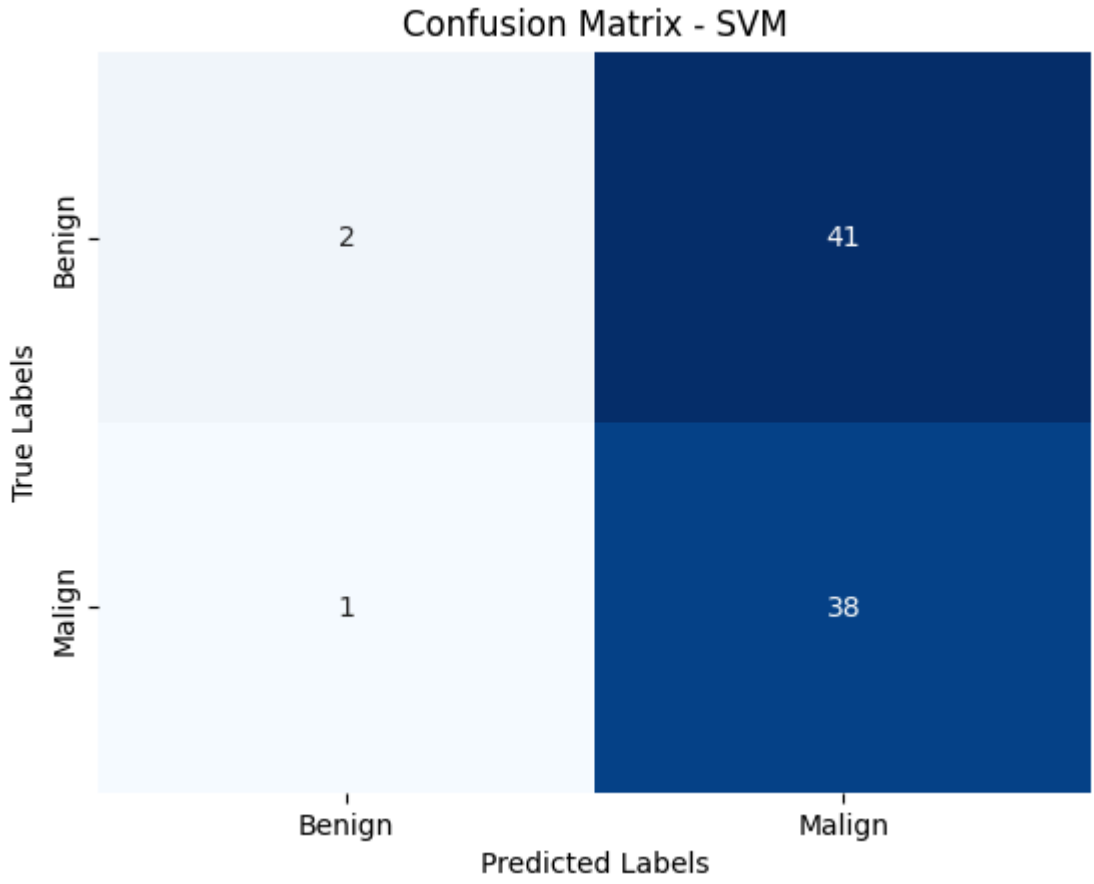
```
Stopping search: maximum iterations reached --> 100
Melhores parâmetros a partir do PSO: C=39.135151846548524, gamma=0.001
Numero do Fold: 1
```



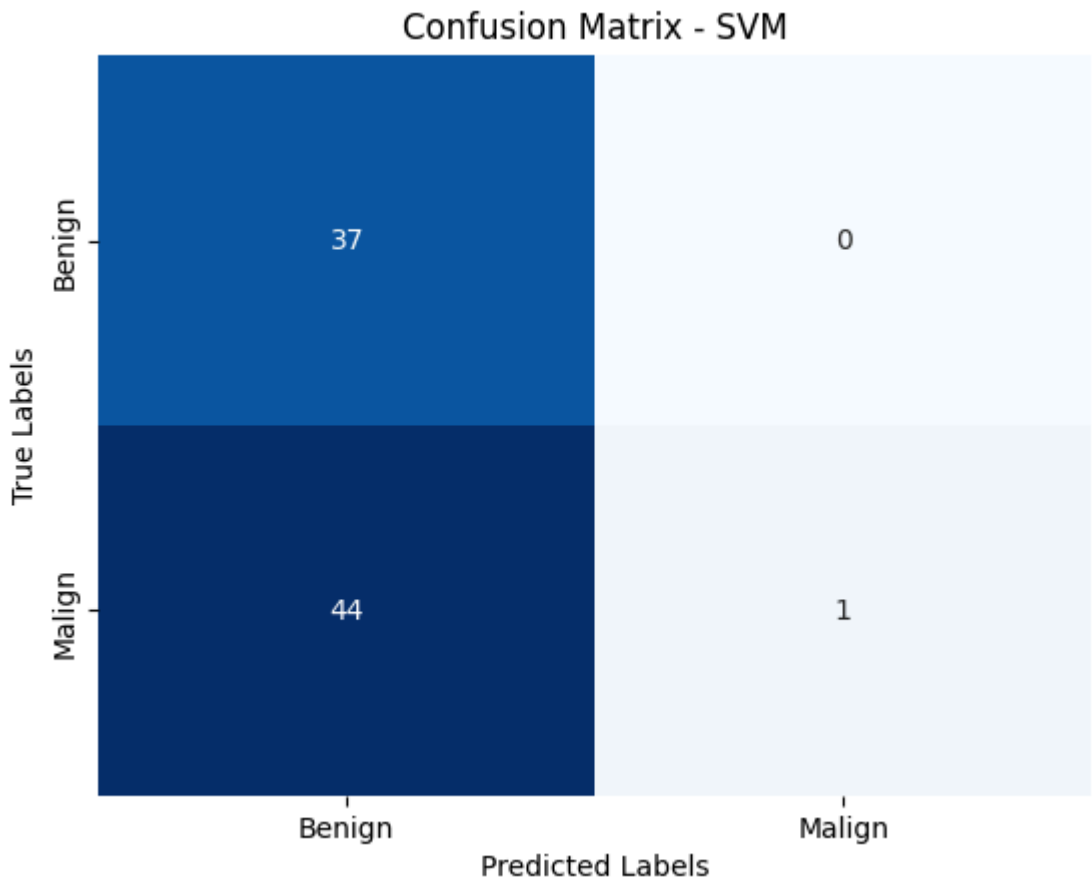
Loading [MathJax]/extensions/Safe.js

```
<Figure size 500x500 with 0 Axes>
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.16      0.25        45
           1       0.47      0.92      0.62        37

    accuracy                           0.50        82
   macro avg       0.59      0.54      0.44        82
weighted avg       0.60      0.50      0.42        82

Numero do Fold: 2
```

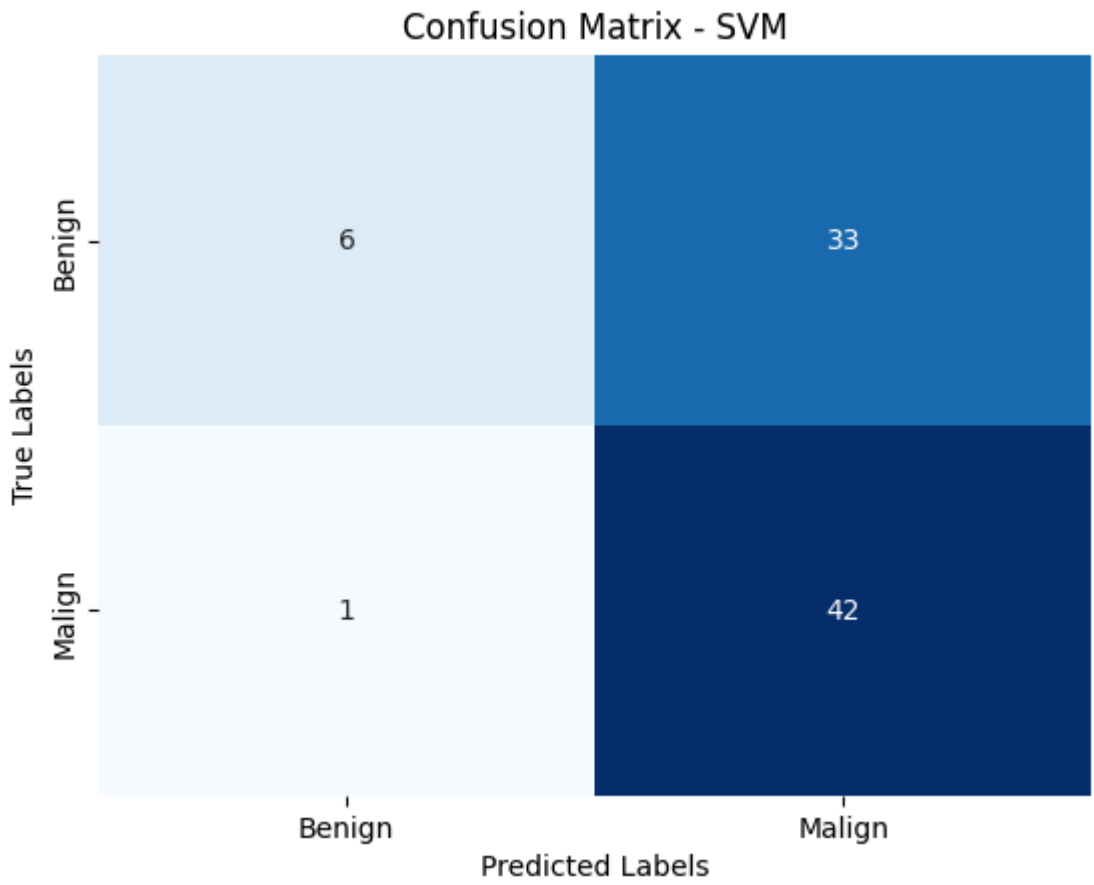



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.05      0.09        43
           1       0.48      0.97      0.64        39

    accuracy                           0.49        82
   macro avg       0.57      0.51      0.37        82
weighted avg       0.58      0.49      0.35        82

Numero do Fold: 3
```

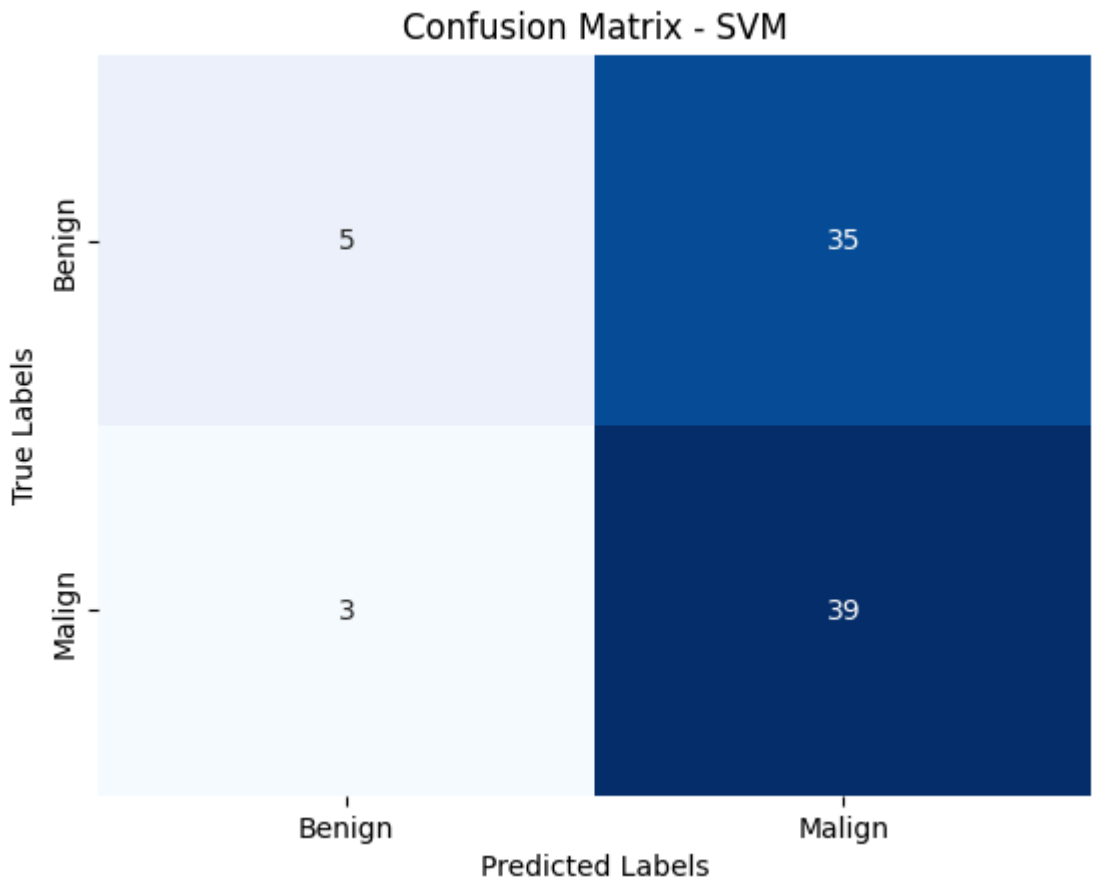



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.46      1.00      0.63        37
           1       1.00      0.02      0.04        45

    accuracy                           0.46        82
   macro avg       0.73      0.51      0.34        82
weighted avg       0.75      0.46      0.31        82

Numero do Fold: 4
```

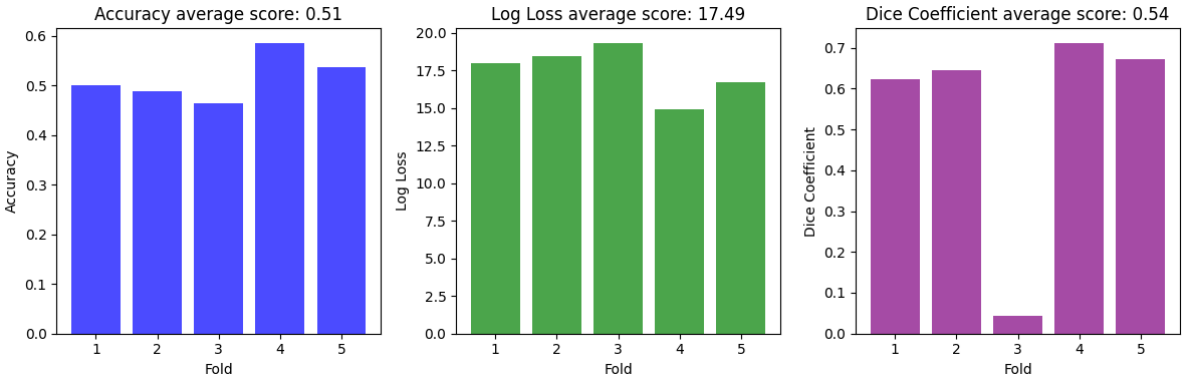
Loading [MathJax]/extensions/Safe.js




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.15      0.26        39
           1       0.56      0.98      0.71        43

    accuracy                           0.59        82
   macro avg       0.71      0.57      0.49        82
weighted avg       0.70      0.59      0.50        82

Numero do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.12      0.21        40
           1       0.53      0.93      0.67        42

    accuracy                           0.54        82
   macro avg       0.58      0.53      0.44        82
weighted avg       0.57      0.54      0.45        82
```




```
[0.5146341463414634, 17.49435859618125, 0.5391354936744037]
```

Support Vector Machine + Random Forest (seleção de features)

[Voltar a Support Vector Machine]

Por fim, repetimos o processo para o dataset obtido pelo Random Forest, usado na seleção

-> 2D

In [50]:
```
data_rf_2d
```

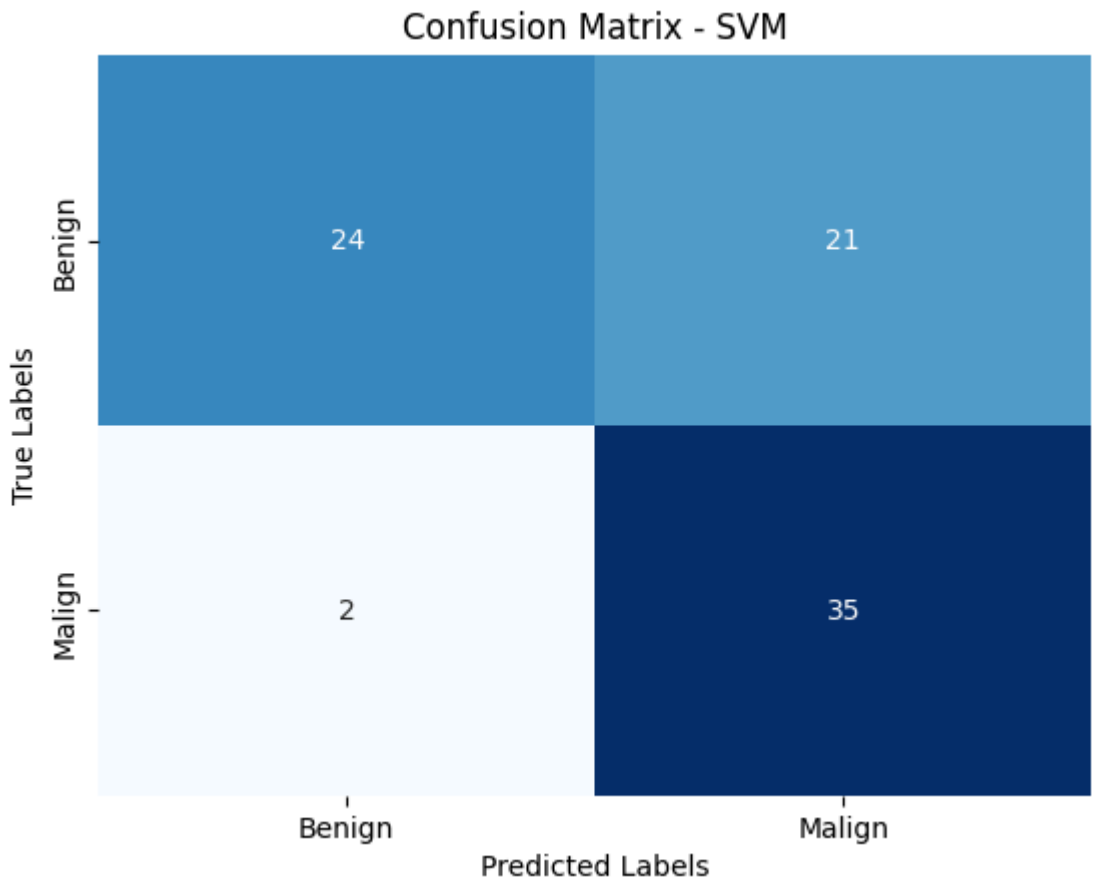
Out[50]:

	Calcification	Lobulation	Spiculation	original_gldm_GrayLevelNonUniformity	original_shape2D
0	6	3	5	160.036585	
1	6	2	2	33.000000	
2	3	1	1	12.142857	
3	6	1	5	43.079208	
4	3	1	1	18.000000	
...	...	...	...	...	
405	6	1	1	24.000000	
406	6	2	2	157.000000	
407	6	3	1	101.000000	
408	6	1	1	79.000000	
409	6	1	1	21.000000	

410 rows × 47 columns

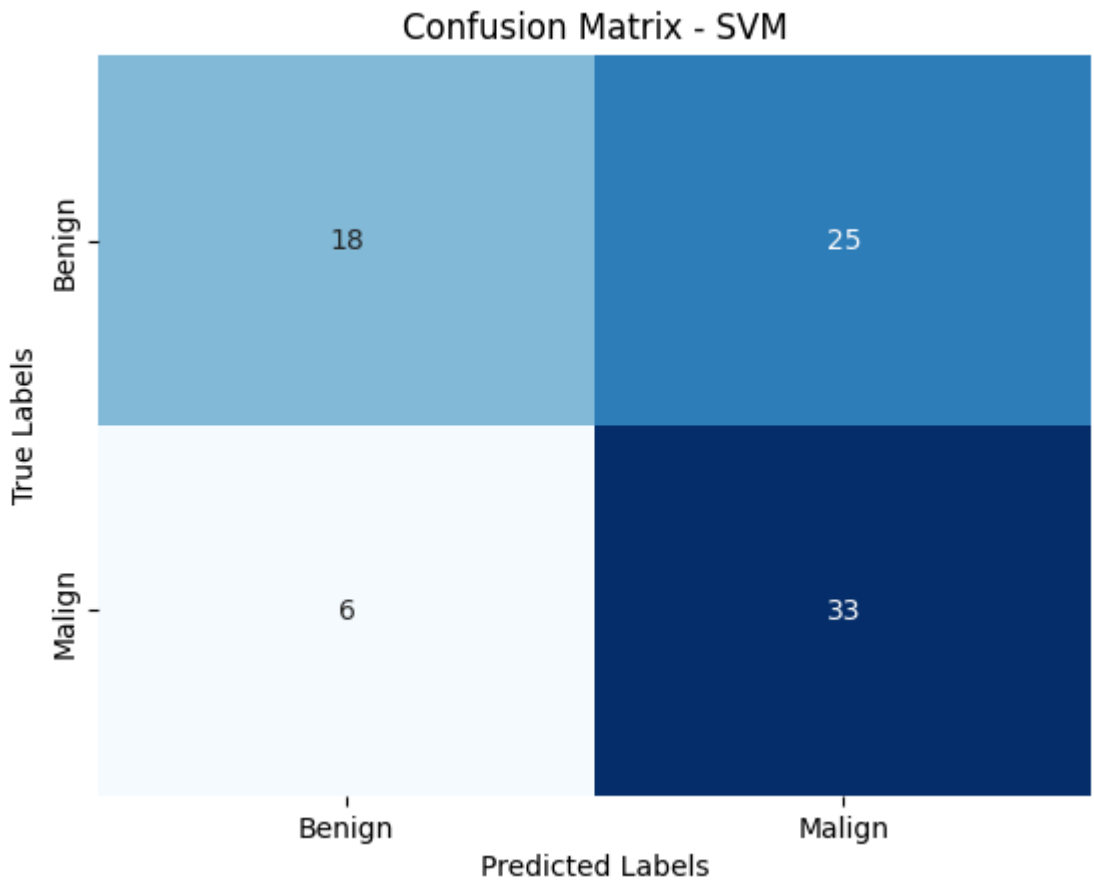
In [51]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = svm_k_fold(fold_2d_rf)
th_svm_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
print(average_metrics_2d)
results_svm_2d.append(average_metrics_2d)
```

```
Stopping search: maximum iterations reached --> 100
Melhores parâmetros a partir do PSO: C=74.52995768319452, gamma=0.001
Numero do Fold: 1
```

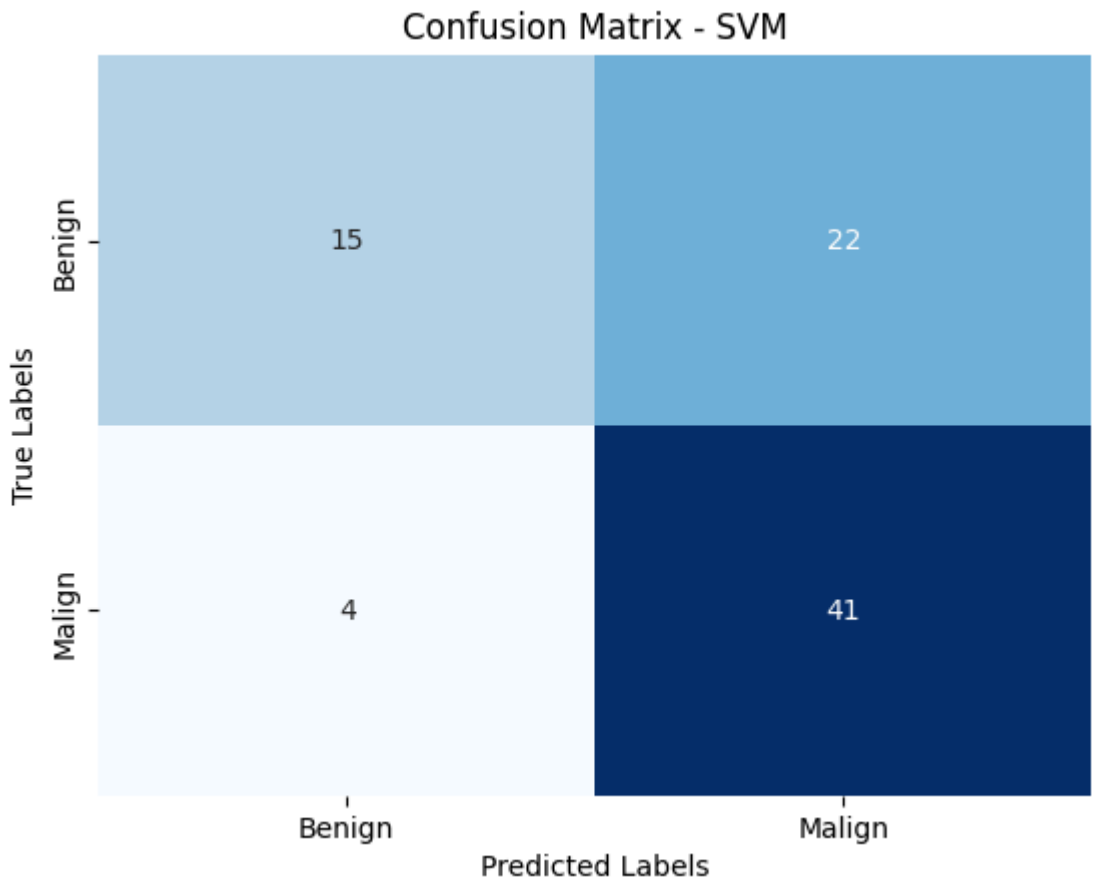



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.53      0.68        45
           1       0.62      0.95      0.75        37

    accuracy                           0.72        82
   macro avg       0.77      0.74      0.71        82
weighted avg       0.79      0.72      0.71        82

Numero do Fold: 2
```

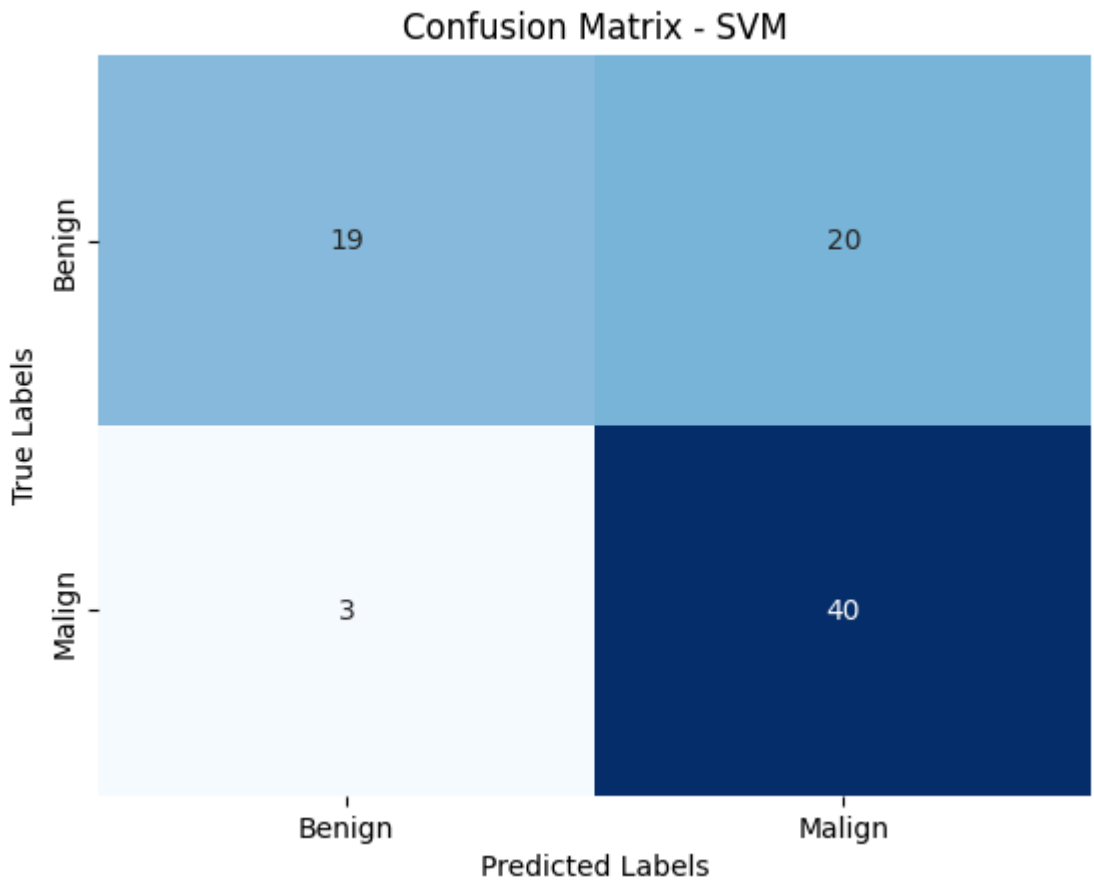
Loading [MathJax]/extensions/Safe.js



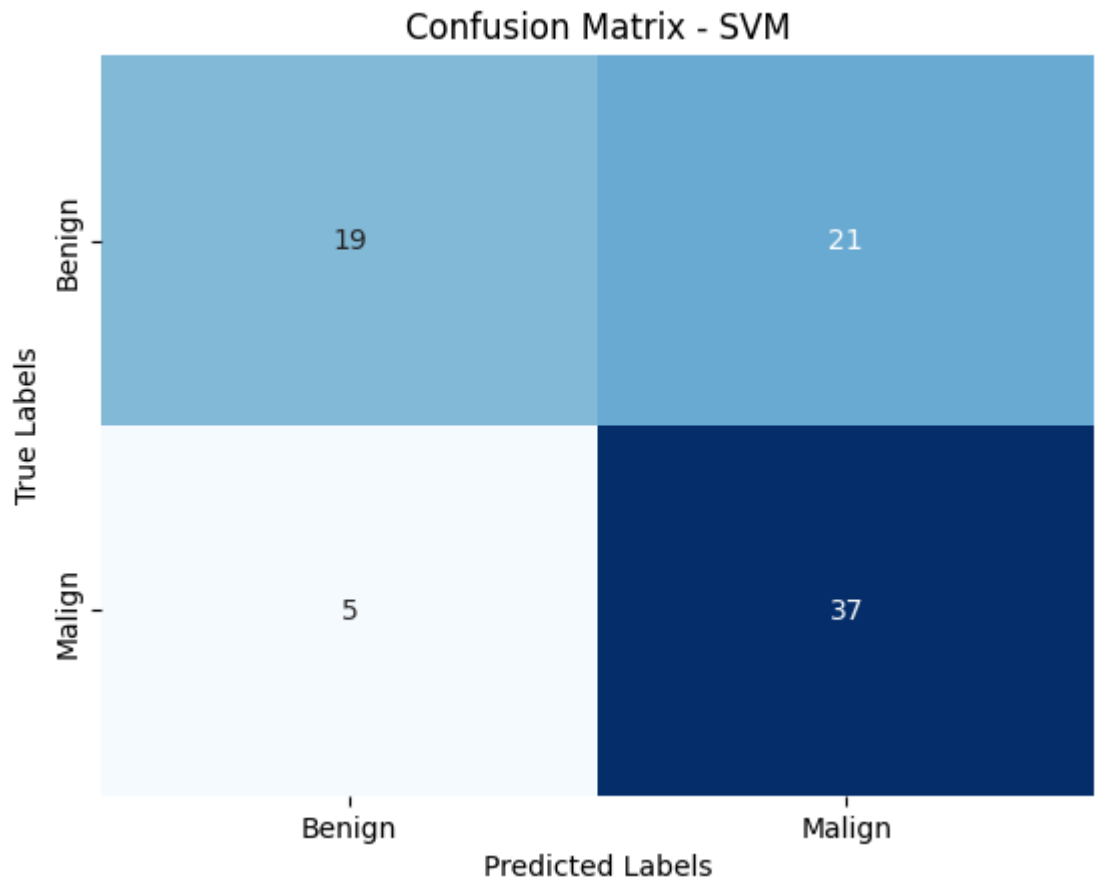

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.42      0.54        43
           1       0.57      0.85      0.68        39

    accuracy                           0.62        82
   macro avg       0.66      0.63      0.61        82
weighted avg       0.66      0.62      0.61        82

Numero do Fold: 3
```



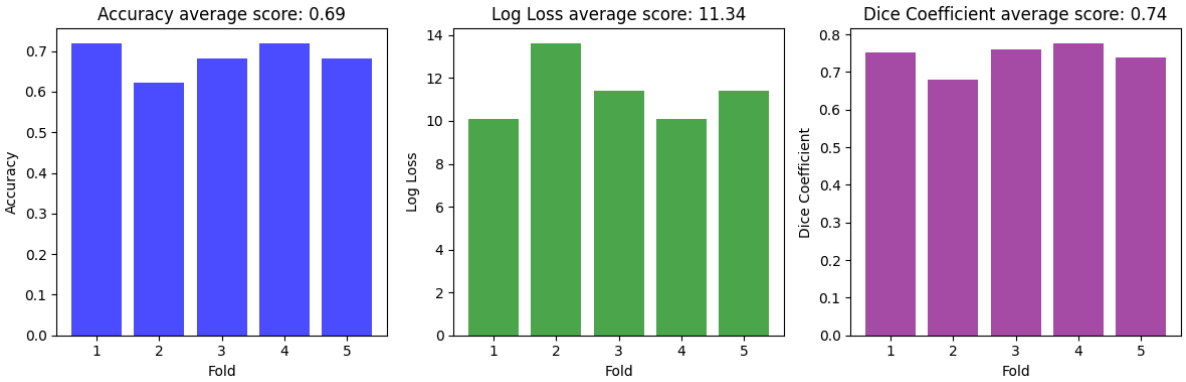

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.41      0.54        37
           1       0.65      0.91      0.76        45

    accuracy                           0.68        82
   macro avg       0.72      0.66      0.65        82
weighted avg       0.71      0.68      0.66        82

Numero do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.49      0.62        39
           1       0.67      0.93      0.78        43

    accuracy                           0.72        82
   macro avg       0.77      0.71      0.70        82
weighted avg       0.76      0.72      0.70        82

Numero do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.47      0.59        40
           1       0.64      0.88      0.74        42

    accuracy                           0.68        82
   macro avg       0.71      0.68      0.67        82
weighted avg       0.71      0.68      0.67        82
```




```
[0.6853658536585366, 11.34056411511247, 0.7418117663125008]
```

-> 3D

In [52]:
```
data_rf_3d
```

Loading [MathJax]/extensions/Safe.js

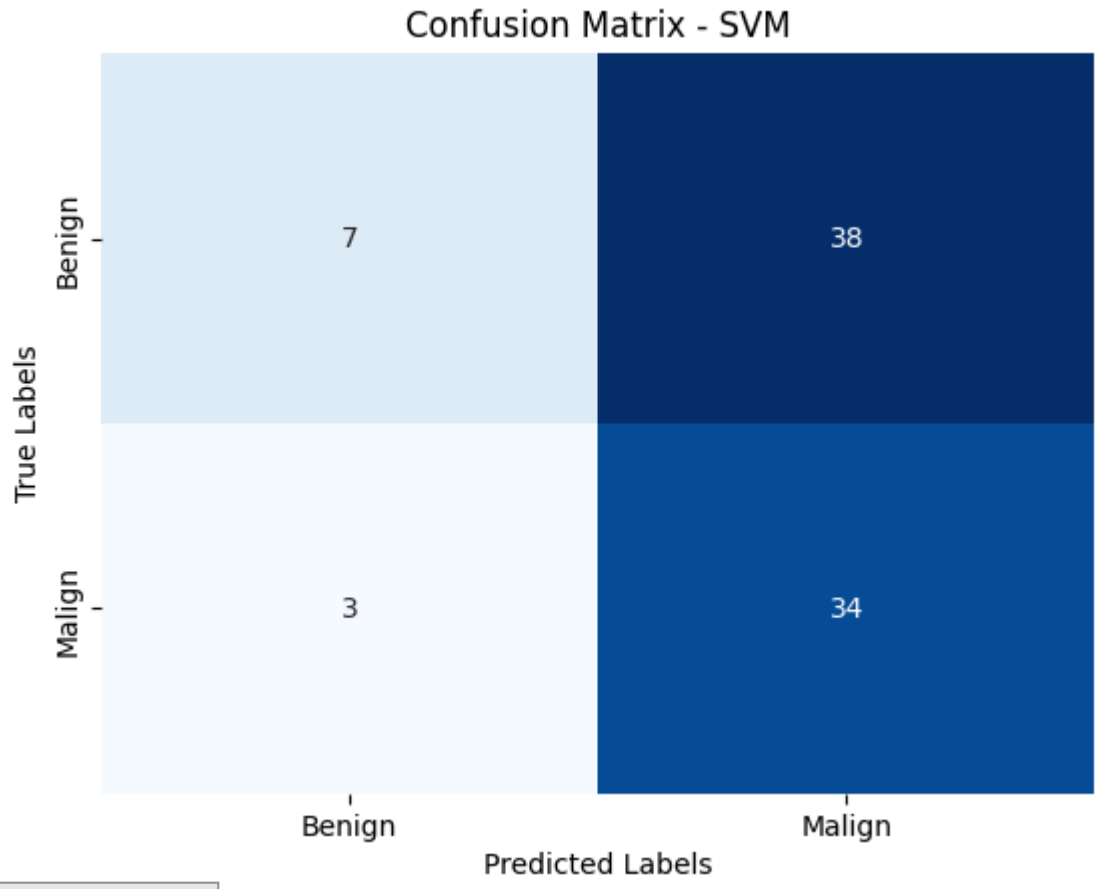
Out[52]:

	Calcification	Spiculation	Lobulation	Margin	original_glrlm_GrayLevelNonUniformity	origina
0	6	5	3	4	757.880362	
1	6	2	2	3	88.384615	
2	3	1	1	5	34.515260	
3	6	5	1	3	437.070150	
4	3	1	1	5	33.846154	
...	...	...	...	...	...	
405	6	1	1	4	41.769231	
406	6	2	2	4	1239.769231	
407	6	1	3	4	573.307692	
408	6	1	1	3	444.769231	
409	6	1	1	2	47.846154	

410 rows × 53 columns

In [53]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = svm_k_fold(fold_3d_rf)
th_svm_3d.append([accuracies_3d,log_losses_3d,dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
print(average_metrics_3d)
results_svm_3d.append(average_metrics_3d)
```

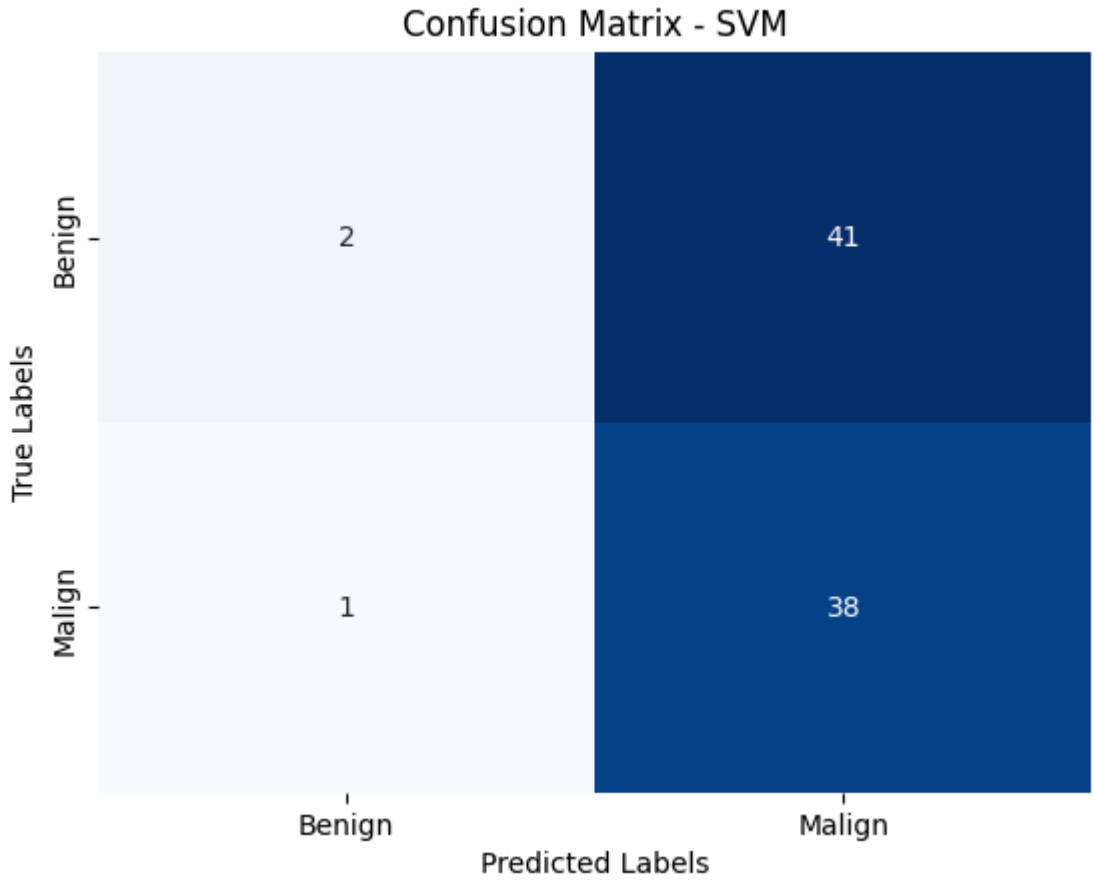
```
Stopping search: maximum iterations reached --> 100
Melhores parâmetros a partir do PSO: C=100.0, gamma=0.001
Numero do Fold: 1
```



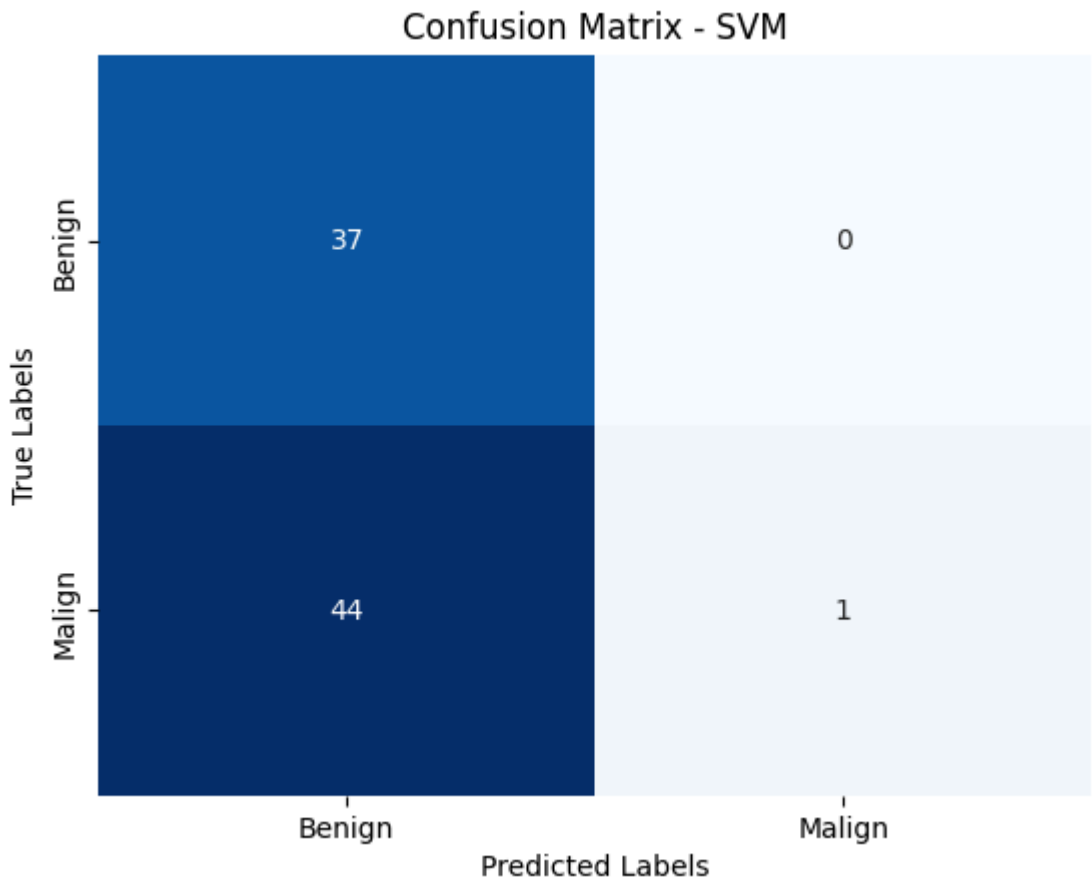
Loading [MathJax]/extensions/Safe.js

```
<Figure size 500x500 with 0 Axes>
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.16      0.25        45
           1       0.47      0.92      0.62        37

    accuracy                           0.50        82
   macro avg       0.59      0.54      0.44        82
weighted avg       0.60      0.50      0.42        82

Numero do Fold: 2
```

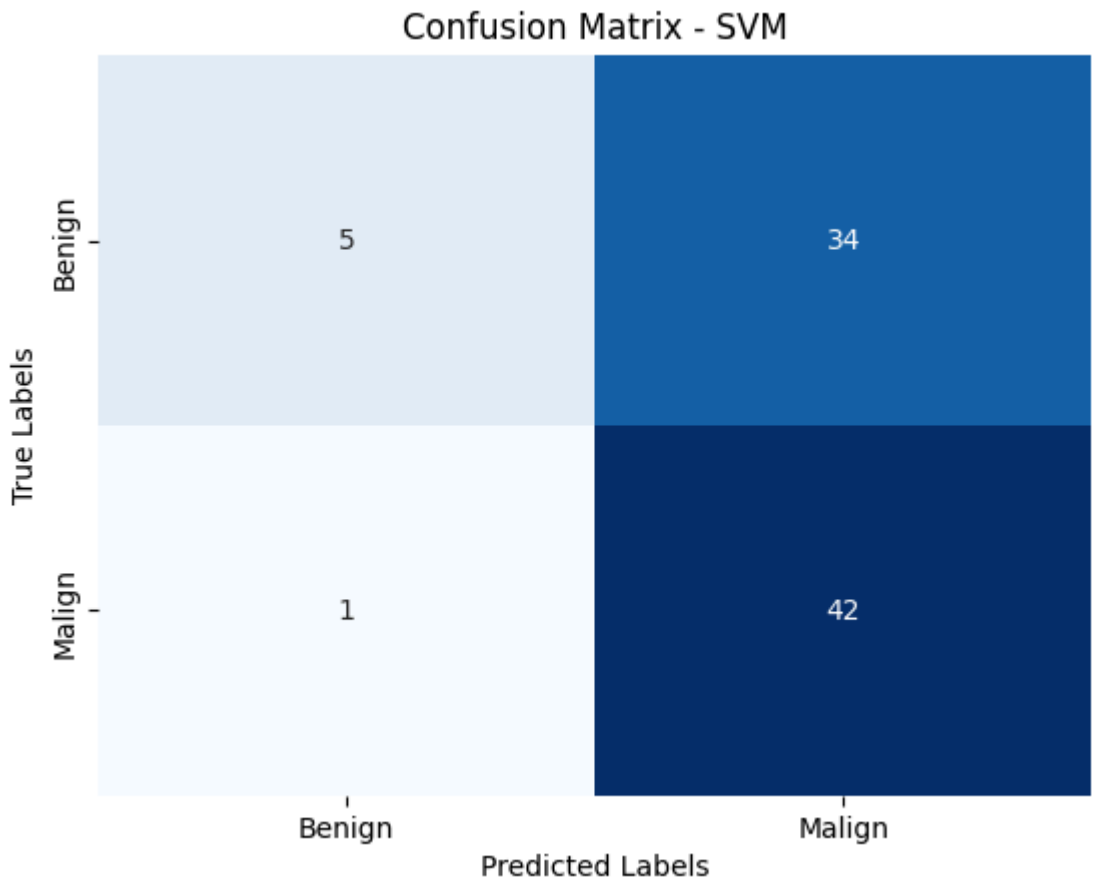



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.05      0.09        43
           1       0.48      0.97      0.64        39

    accuracy                           0.49        82
   macro avg       0.57      0.51      0.37        82
weighted avg       0.58      0.49      0.35        82

Numero do Fold: 3
```

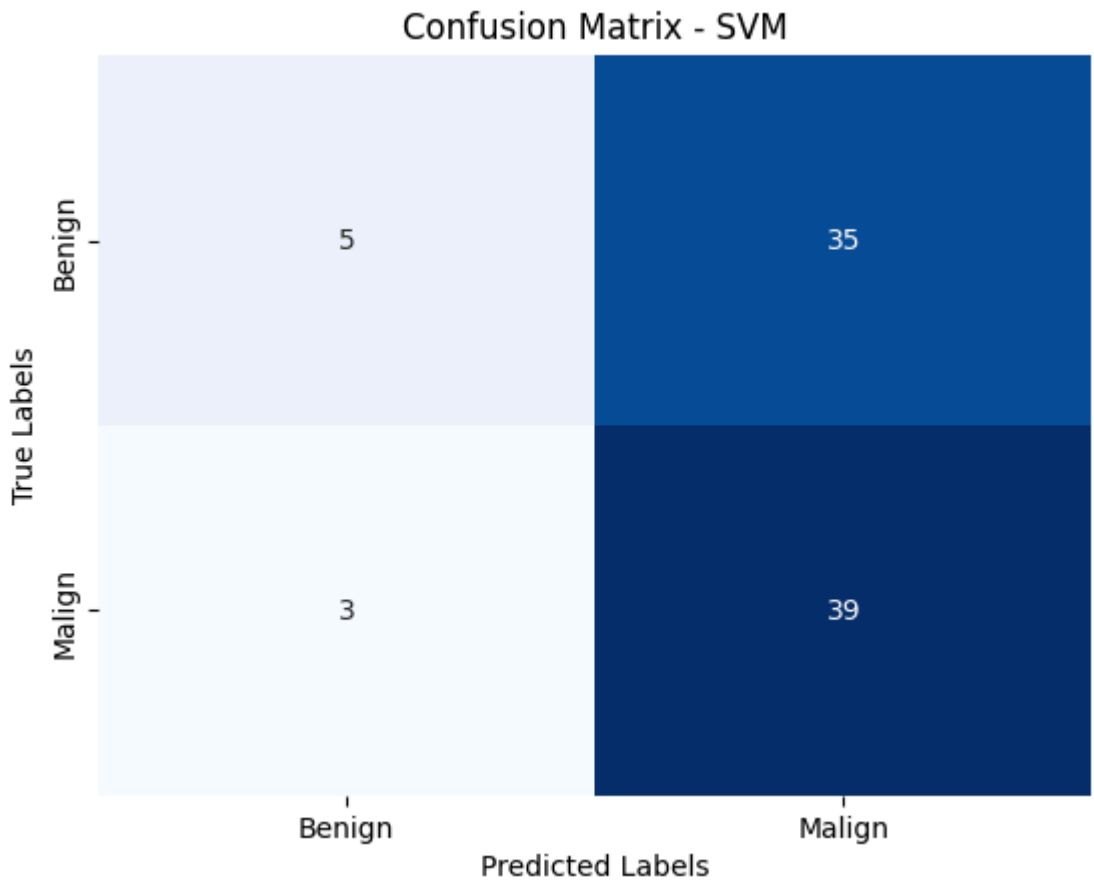



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.46      1.00      0.63        37
           1       1.00      0.02      0.04        45

    accuracy                           0.46        82
   macro avg       0.73      0.51      0.34        82
weighted avg       0.75      0.46      0.31        82

Numero do Fold: 4
```

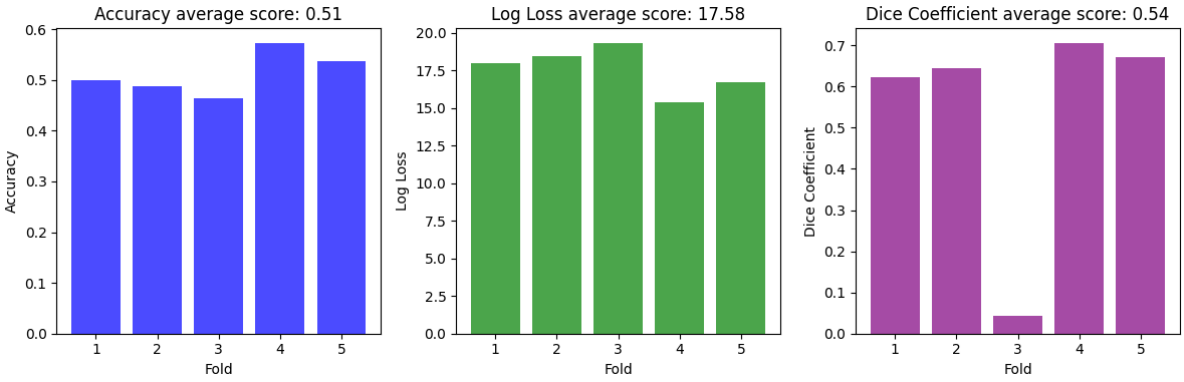
Loading [MathJax]/extensions/Safe.js




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.13      0.22        39
           1       0.55      0.98      0.71        43

    accuracy                           0.57        82
   macro avg       0.69      0.55      0.46        82
weighted avg       0.69      0.57      0.48        82

Numero do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.12      0.21        40
           1       0.53      0.93      0.67        42

    accuracy                           0.54        82
   macro avg       0.58      0.53      0.44        82
weighted avg       0.57      0.54      0.45        82
```




```
[0.5121951219512195, 17.582269945910802, 0.5379390829067068]
```

Resultados Support Vector Machine

[Voltar a Support Vector Machine]

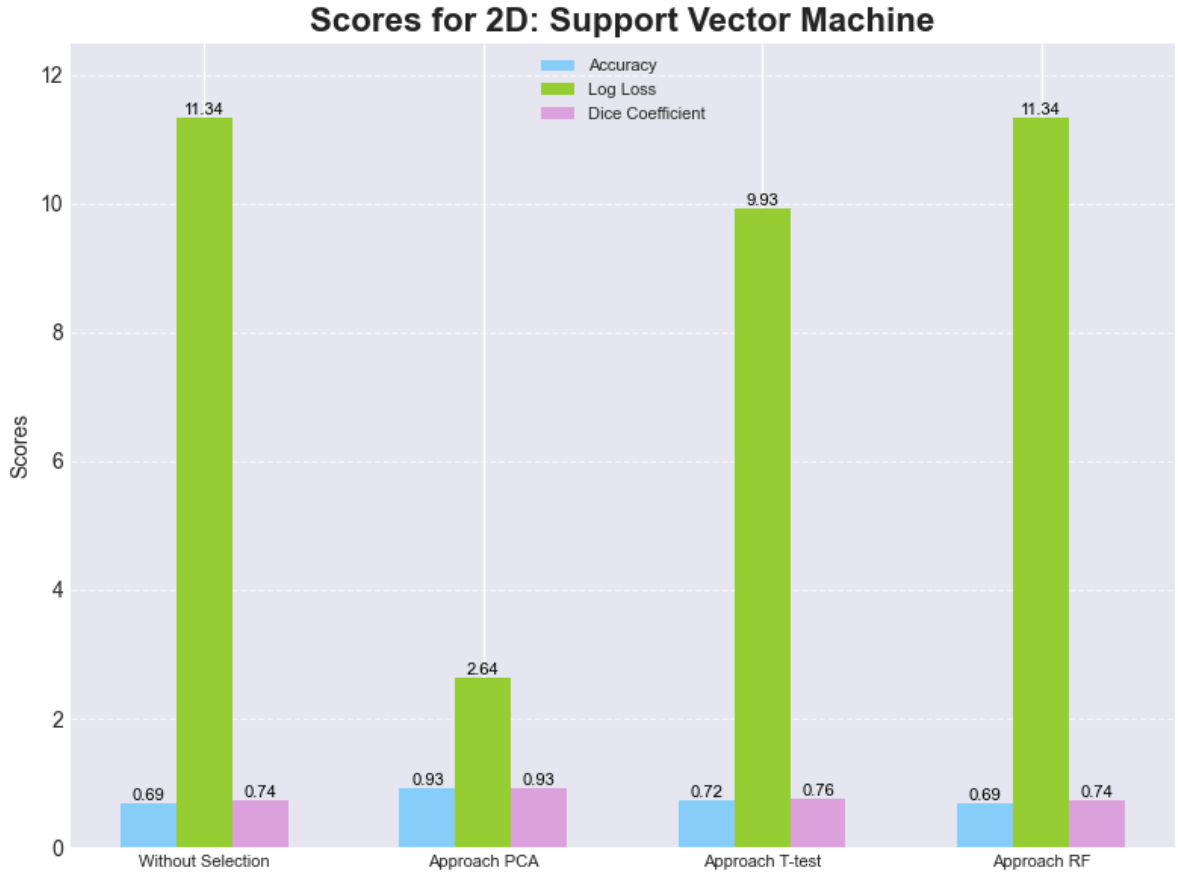
-> 2D

In [54]:
```
print(results_svm_2d)
```

```
[[0.6853658536585366, 11.34056411511247, 0.7418117663125008], [0.9268292682926829,
2.637340491886621, 0.9254514415467104], [0.724390243902439, 9.933982519439606, 0.7
593110494818275], [0.6853658536585366, 11.34056411511247, 0.7418117663125008]]
```

In [55]:
```
plot_scores(results_svm_2d, "2D: Support Vector Machine")
```

```
/var/folders/2t/mv0q1c7j2z97cgvtlhbfrhhm0000gn/T/ipykernel_67329/3843270632.py:9:
MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are depreca
ted since 3.6, as they no longer correspond to the styles shipped by seaborn. Howe
ver, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly
use the seaborn API instead.
  plt.style.use('seaborn-darkgrid')
```



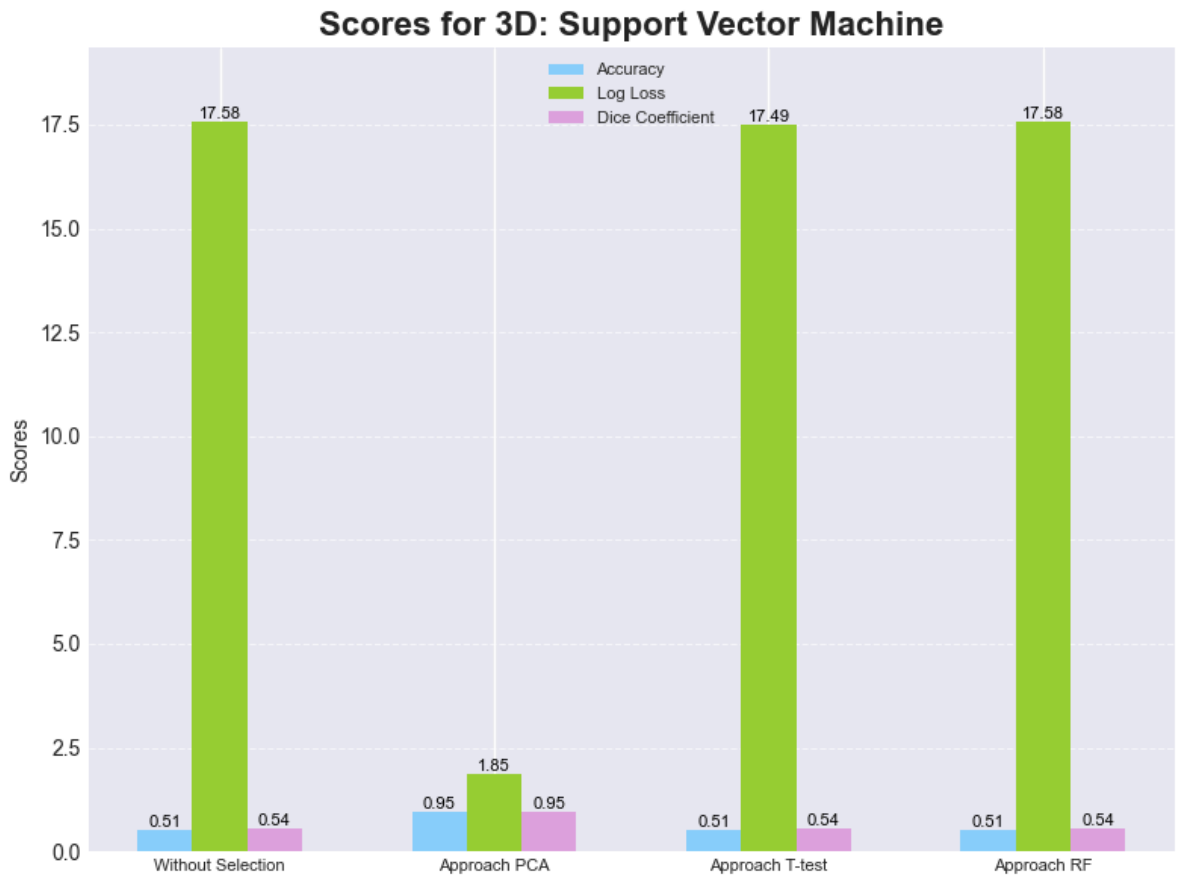
-> 3D

In [56]:
```
print(results_svm_3d)
```

```
[[0.5121951219512195, 17.582269945910802, 0.5379390829067068], [0.948780487804878
1, 1.8461383443206347, 0.949257284252871], [0.5146341463414634, 17.49435859618125,
0.5391354936744037], [0.5121951219512195, 17.582269945910802, 0.5379390829067068]]
```

In [57]:
```
plot_scores(results_svm_3d, "3D: Support Vector Machine")
```

```
/var/folders/2t/mv0q1c7j2z97cgvtlhbfrhhm0000gn/T/ipykernel_67329/3843270632.py:9:
MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are depreca
ted since 3.6, as they no longer correspond to the styles shipped by seaborn. Howe
ver, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly
use the seaborn API instead.
  plt.style.use('seaborn-darkgrid')
```



Random Forest

[Voltar a Training Models]

O algoritmo Random Forest também é um dos algoritmos mais vulgarmente utilizados para a fase de classificação dos nódulos como malignos ou benignos. É um algoritmo que lida bem com a variabilidade dos dados e é robusto contra overlap de variáveis, o que pode ser útil tendo em conta que temos muitas features a considerar.

Estudos que usam Random Forest na classificação:

- LG18 (Random Forest + PCA)

Código que iremos usar para treinar os modelos:

In [58]:
```
def random_forest_kfold(fold, k=5, random_state=42):

    accuracies = []
    log_losses = []
    dice_coefs = []
    n_fold = 1
    model = RandomForestClassifier(random_state=random_state)
    for  (X_train, X_test, y_train, y_test) in fold:
        print(f"Número do Fold: {n_fold}" )
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)
        acc, logloss, dice = metrics_testing(y_test, predictions, "Random Forest")
        accuracies.append(acc)
        log_losses.append(logloss)
        dice_coefs.append(dice)
        n_fold += 1
```

```
    return accuracies, log_losses, dice_coefs
```

In [59]:
```
results_rf_2d = []
results_rf_3d = []
th_rf_2d = []
th_rf_3d = []
```

Random Forest sem seleção de features

[Voltar a Random Forest]

-> 2D

In [60]:
```
data_2d
```

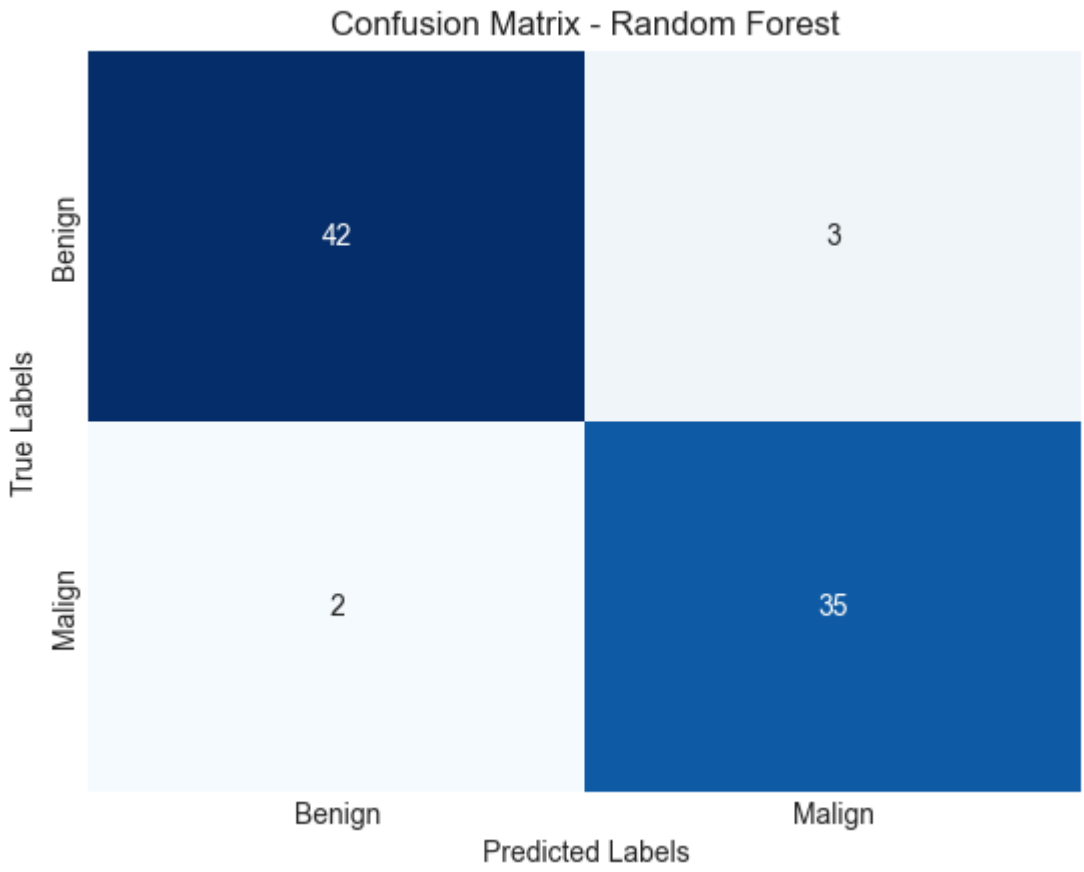
Out[60]:

	Spiculation	Lobulation	Sphericity	Margin	Subtlety	Texture	Calcification	Malignancy	dia ori
0	5	3	3	4	5	5	6	1	
1	2	2	4	3	5	4	6	1	
2	1	1	2	5	2	5	3	0	
3	5	1	4	3	5	5	6	1	
4	1	1	5	5	3	5	3	0	
...	...	...	...	...	...	...	...	...	
405	1	1	3	4	3	5	6	0	
406	2	2	4	4	5	5	6	1	
407	1	3	3	4	5	5	6	1	
408	1	1	3	3	5	5	6	1	
409	1	1	4	2	4	5	6	0	

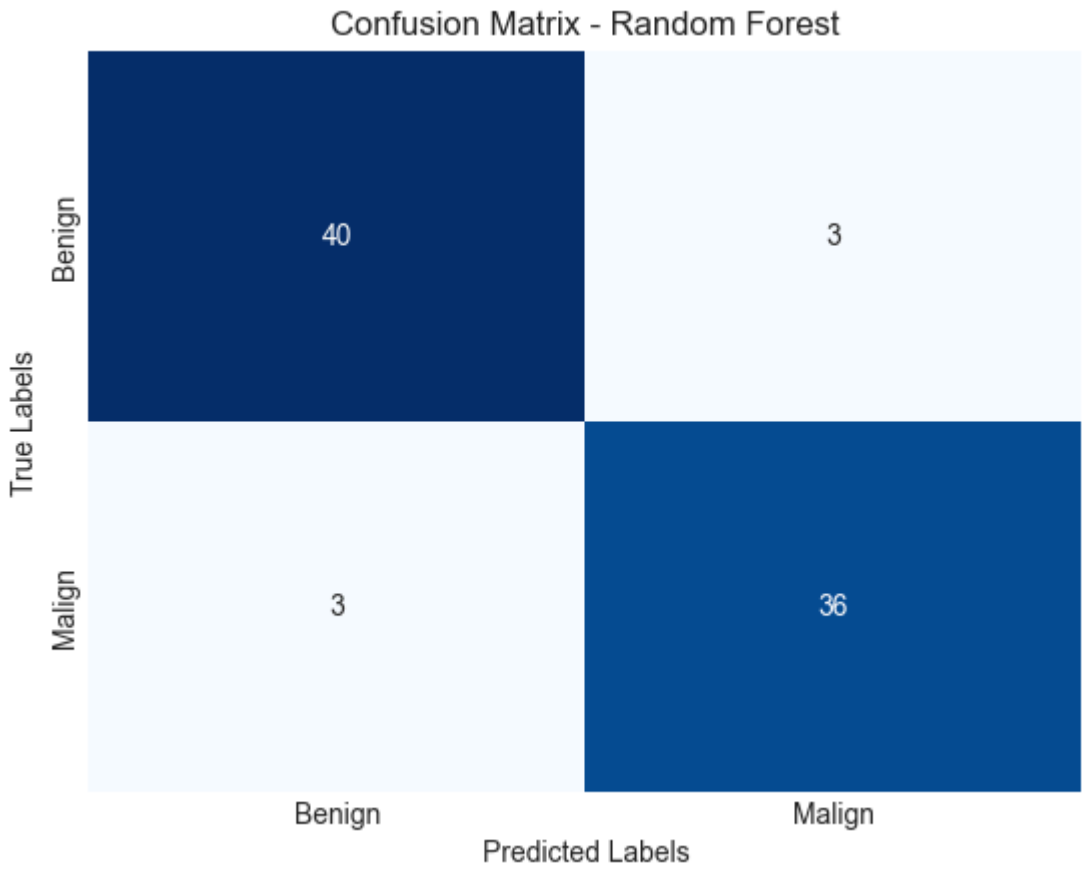
410 rows × 109 columns

In [61]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = random_forest_kfold(fold_2d)
th_rf_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
print(average_metrics_2d)
results_rf_2d.append(average_metrics_2d)
```

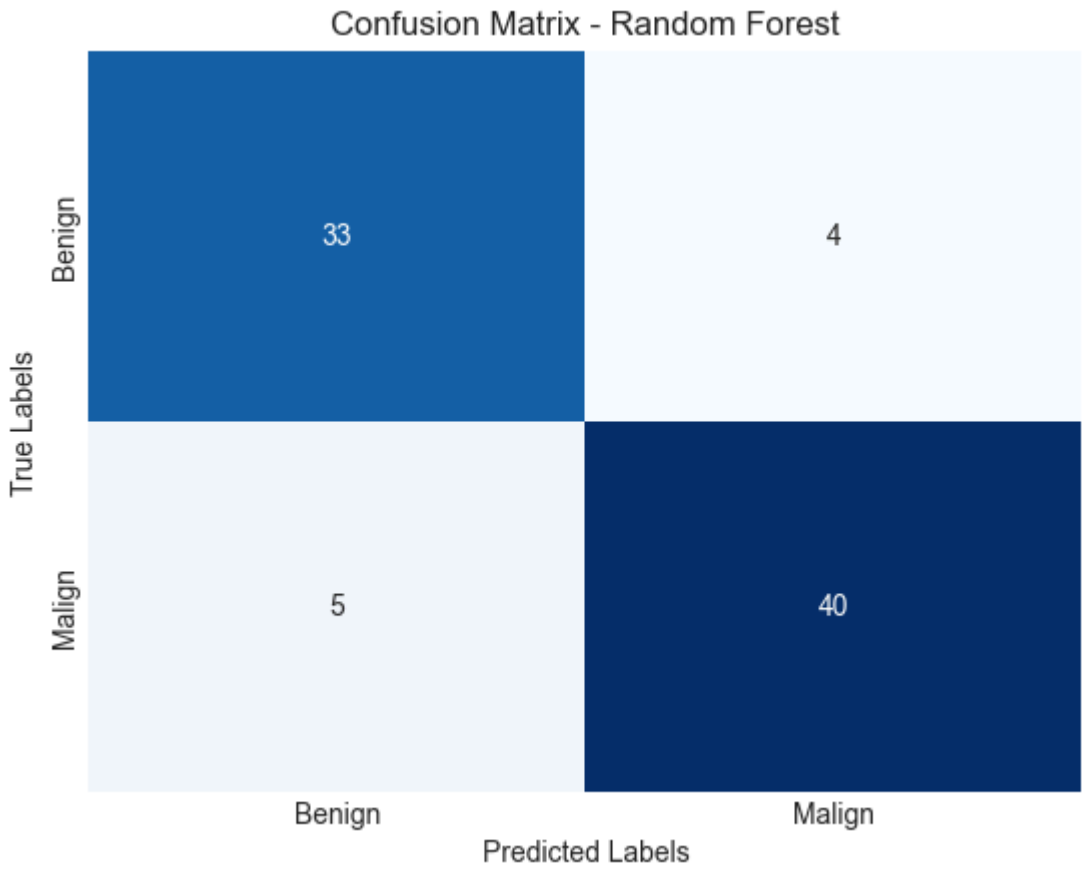
```
Número do Fold: 1
```



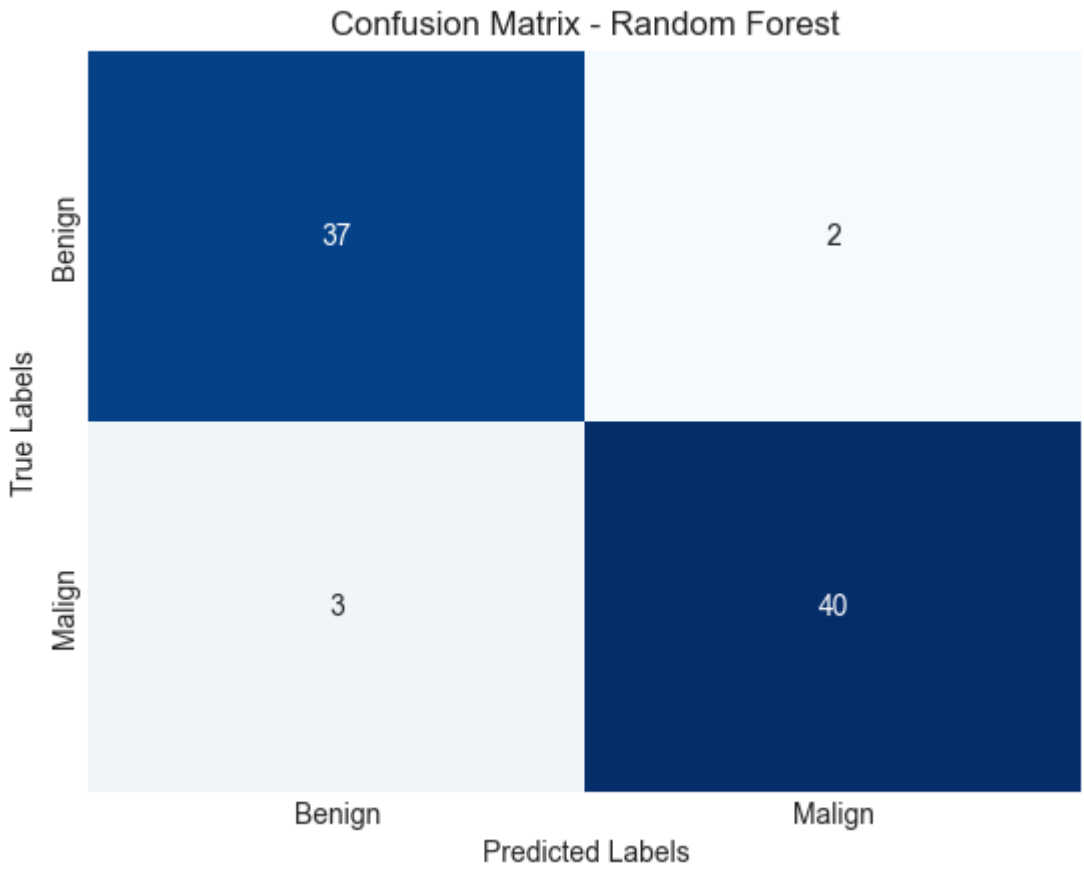

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        45
           1       0.92      0.95      0.93        37

    accuracy                           0.94        82
   macro avg       0.94      0.94      0.94        82
weighted avg       0.94      0.94      0.94        82

Número do Fold: 2
```



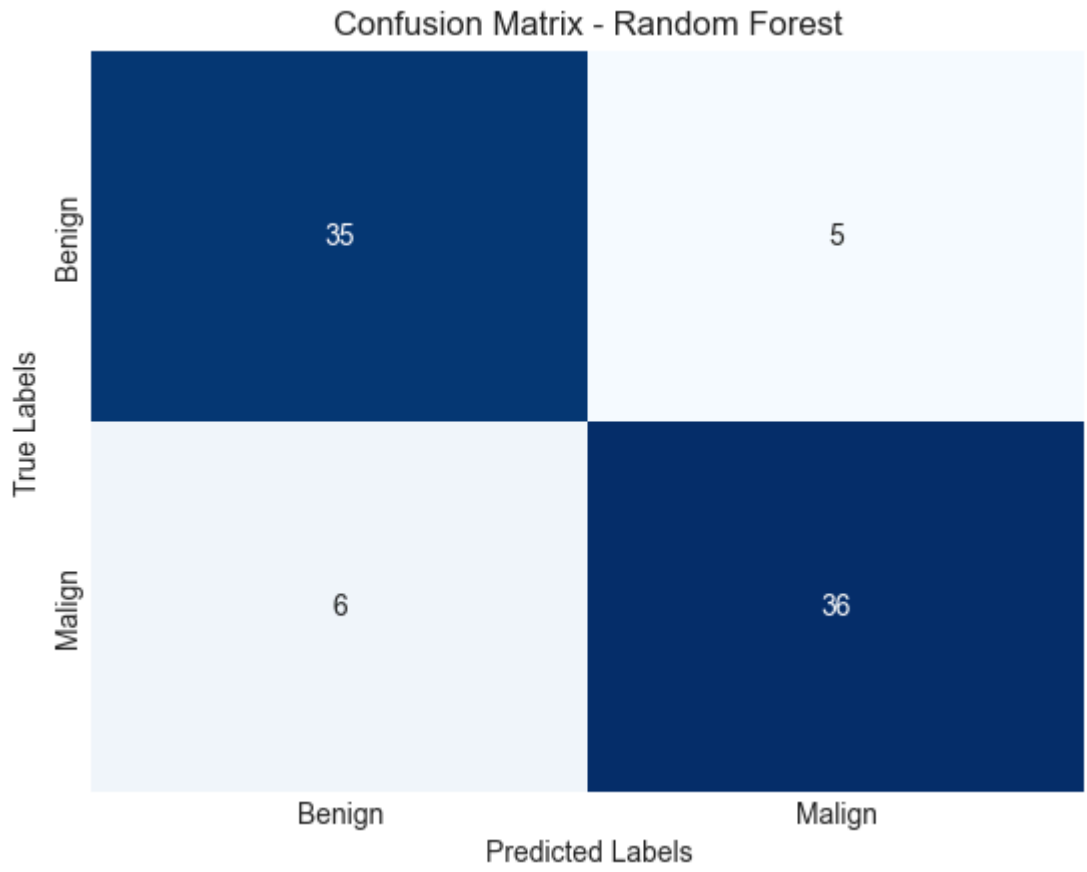

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        43
           1       0.92      0.92      0.92        39

    accuracy                           0.93        82
   macro avg       0.93      0.93      0.93        82
weighted avg       0.93      0.93      0.93        82

Número do Fold: 3
```



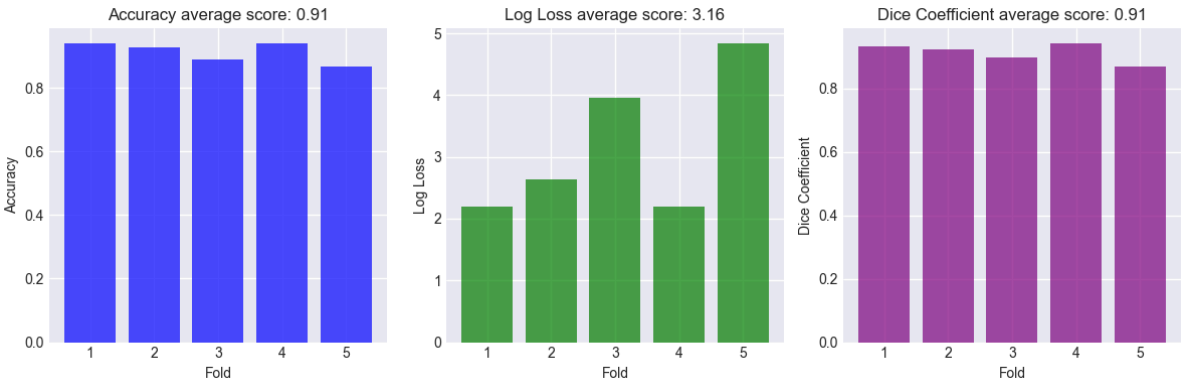

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.89      0.88        37
           1       0.91      0.89      0.90        45

    accuracy                           0.89        82
   macro avg       0.89      0.89      0.89        82
weighted avg       0.89      0.89      0.89        82

Número do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.95      0.94        39
           1       0.95      0.93      0.94        43

    accuracy                           0.94        82
   macro avg       0.94      0.94      0.94        82
weighted avg       0.94      0.94      0.94        82

Número do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.88      0.86        40
           1       0.88      0.86      0.87        42

    accuracy                           0.87        82
   macro avg       0.87      0.87      0.87        82
weighted avg       0.87      0.87      0.87        82
```




```
[0.9121951219512194, 3.164808590263945, 0.9127866022021893]
```

-> 3D

In [62]:
```
data_3d
```

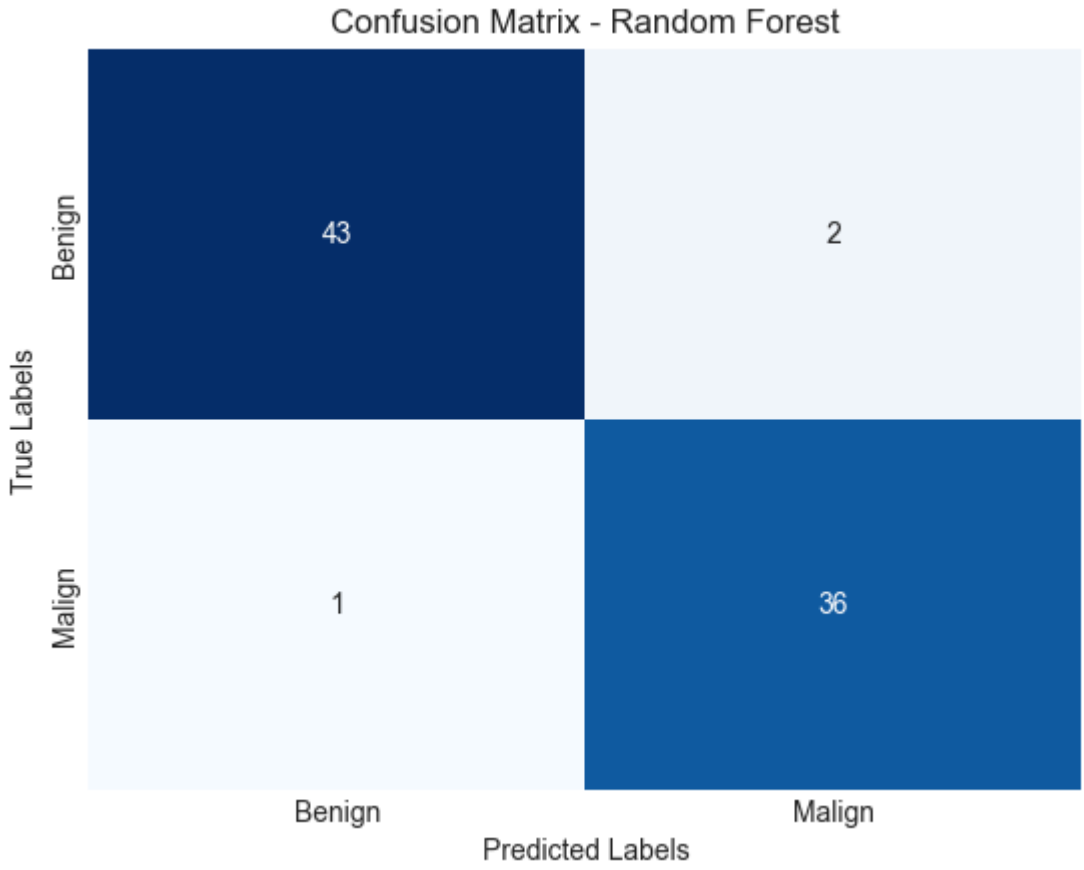
Out[62]:

	Spiculation	Lobulation	Sphericity	Margin	Subtlety	Texture	Calcification	Malignancy	dia ori
0	5	3	3	4	5	5	6	1	
1	2	2	4	3	5	4	6	1	
2	1	1	2	5	2	5	3	0	
3	5	1	4	3	5	5	6	1	
4	1	1	5	5	3	5	3	0	
...	...	...	...	...	...	...	...	...	
405	1	1	3	4	3	5	6	0	
406	2	2	4	4	5	5	6	1	
407	1	3	3	4	5	5	6	1	
408	1	1	3	3	5	5	6	1	
409	1	1	4	2	4	5	6	0	

410 rows × 105 columns

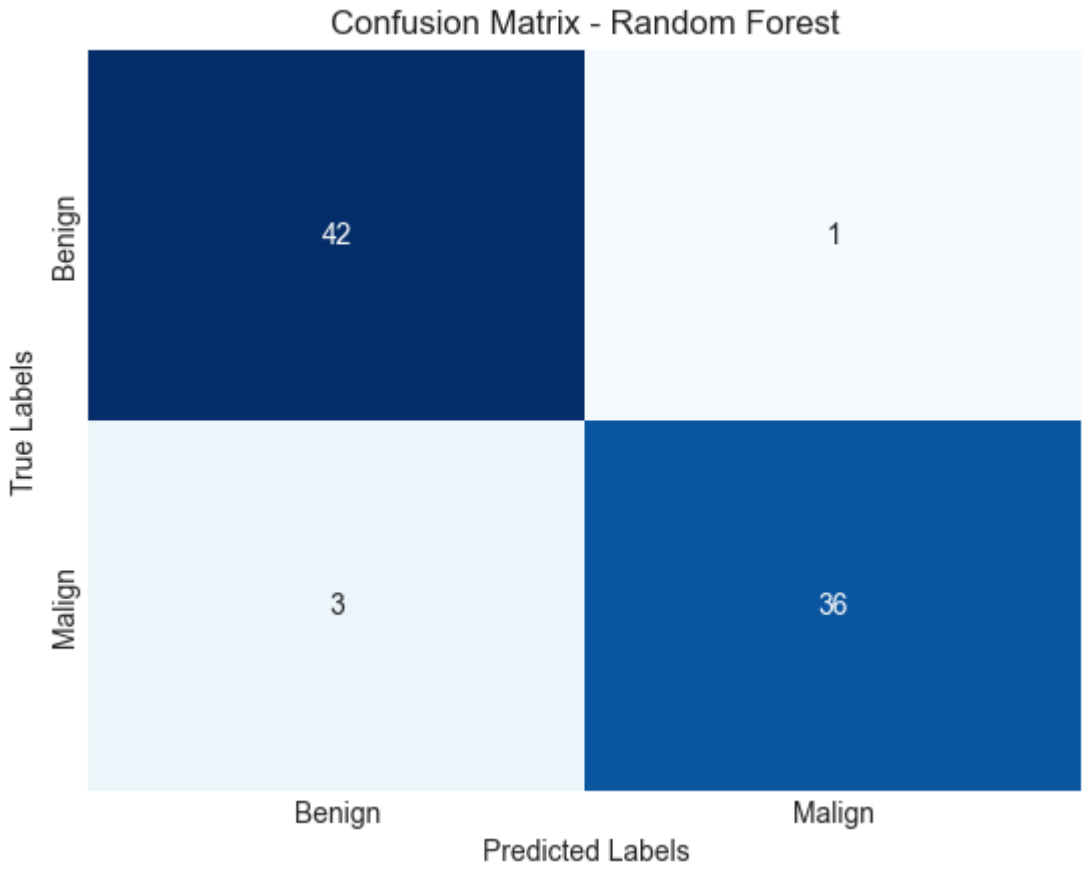
In [63]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = random_forest_kfold(fold_3d)
th_rf_3d.append([accuracies_3d,log_losses_3d,dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
print(average_metrics_3d)
results_rf_3d.append(average_metrics_3d)
```

```
Número do Fold: 1
```

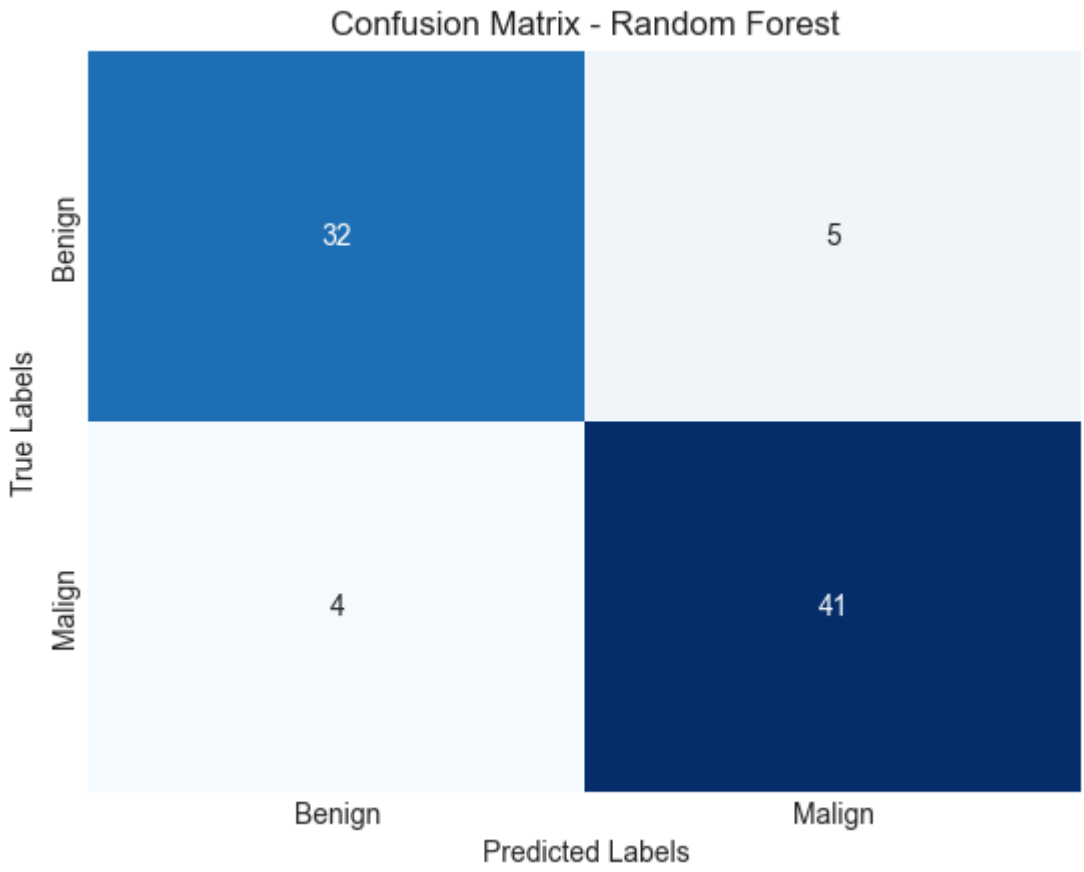



```
<Figure size 500x500 with 0 Axes>
```

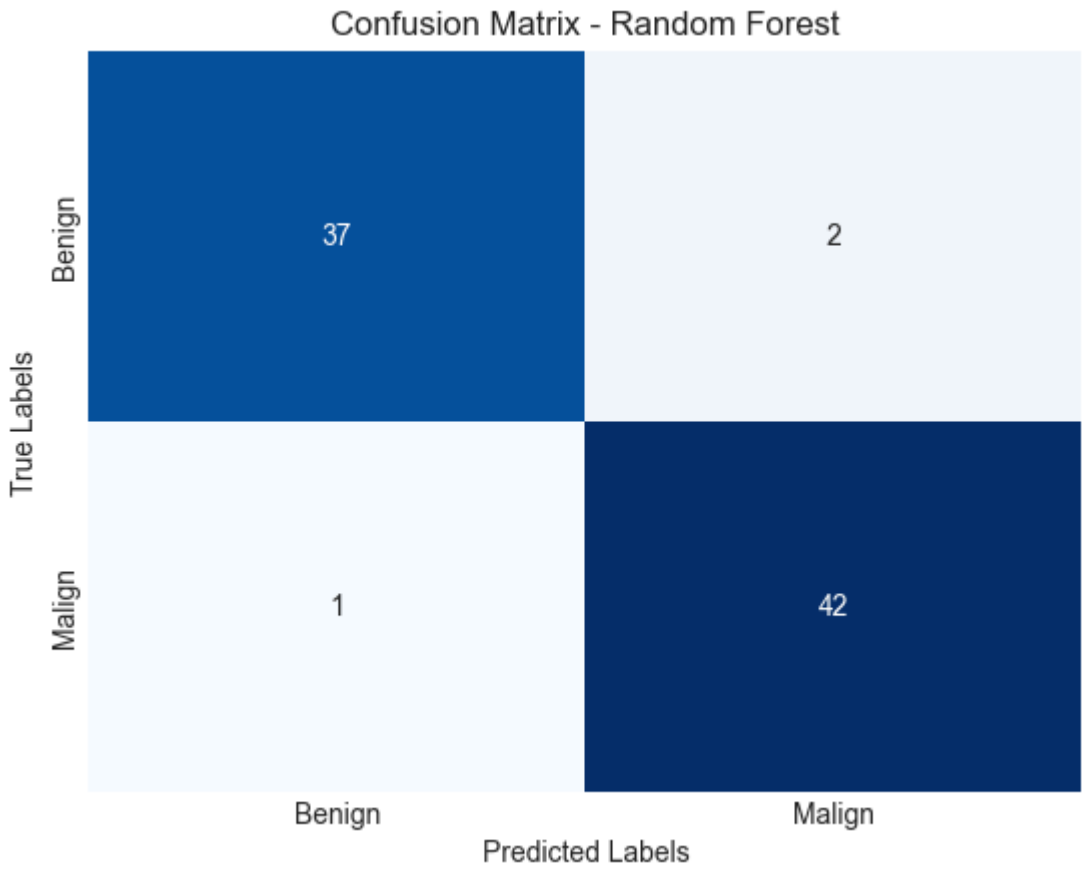
```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        45
           1       0.95      0.97      0.96        37

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Número do Fold: 2
```



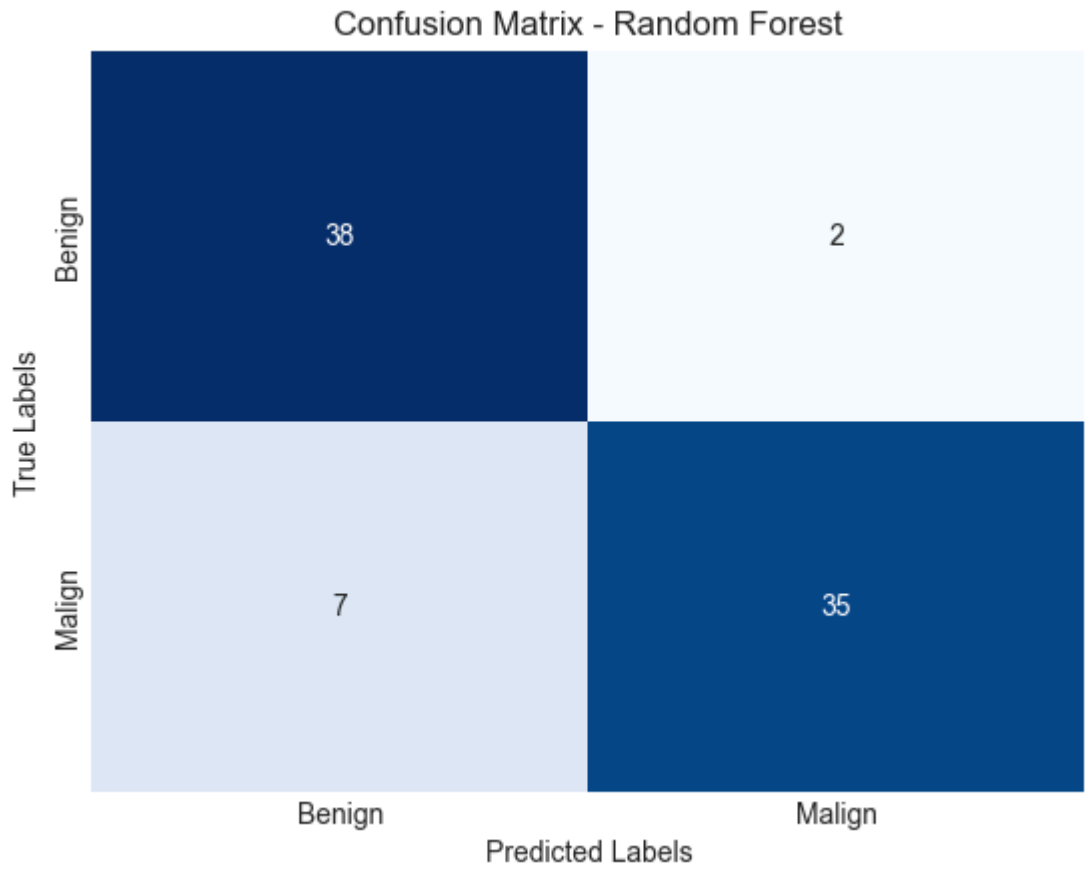

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        43
           1       0.97      0.92      0.95        39

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número do Fold: 3
```



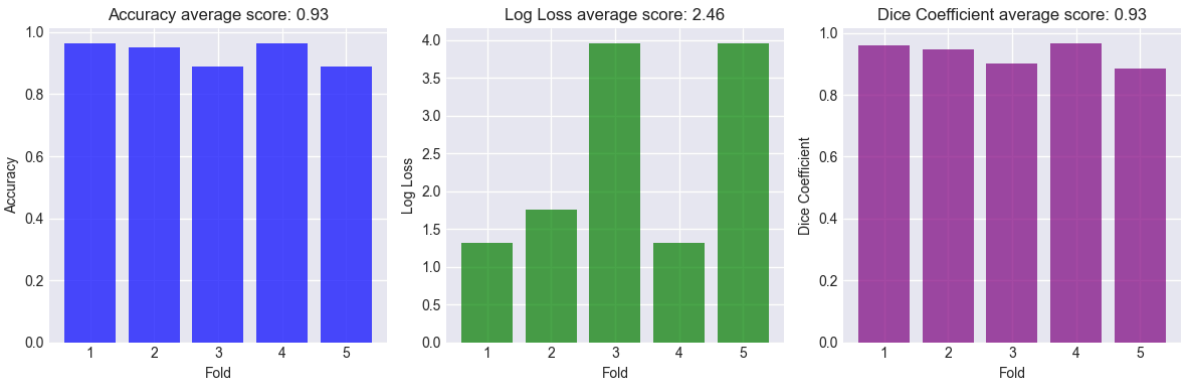

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.88        37
           1       0.89      0.91      0.90        45

    accuracy                           0.89        82
   macro avg       0.89      0.89      0.89        82
weighted avg       0.89      0.89      0.89        82

Número do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        39
           1       0.95      0.98      0.97        43

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Número do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.95      0.89        40
           1       0.95      0.83      0.89        42

    accuracy                           0.89        82
   macro avg       0.90      0.89      0.89        82
weighted avg       0.90      0.89      0.89        82
```




```
[0.9317073170731707, 2.461517792427513, 0.9320121025795863]
```

Random Forest + PCA - abordagem usada pelo estudo LG18

[Voltar a Random Forest]

Iremos agora passar à implementação do modelo Random Forest, usando o dataset preparado a partir do método de Principal Component Analysis, data_pca.

-> 2D

In [64]:
```
data_pca_2d
```

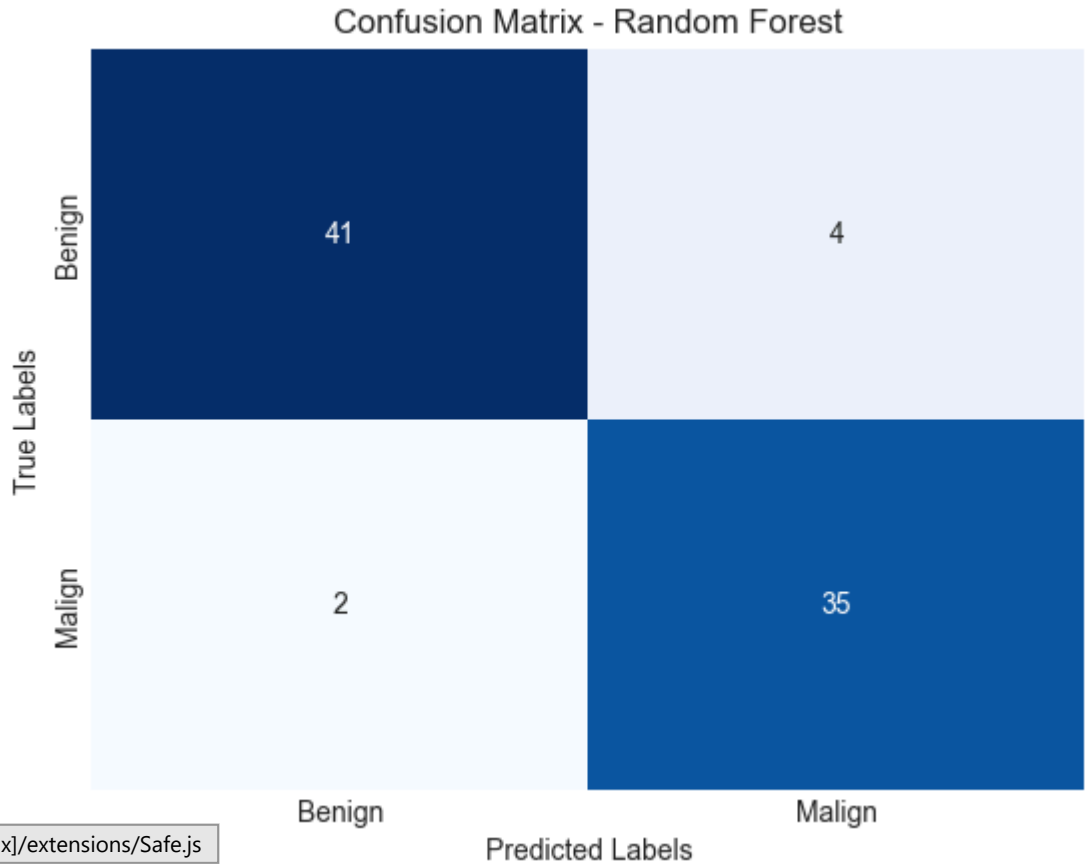
Out[64]:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	P
0	0.605527	14.312994	2.618862	-5.010918	-1.873619	0.486000	3.780943	4.509357	-3.0005
1	-2.267379	-1.819947	0.040090	0.846899	-1.514118	-0.992908	-0.811545	0.548039	-0.1913
2	5.058170	-2.480426	-3.601613	-4.459463	-1.977084	-3.833077	1.266602	3.487037	0.1742
3	26.065083	6.727777	-6.985184	4.449976	-1.804223	1.655932	-2.786482	-3.047505	1.8274
4	-2.158470	-3.958437	-0.673922	0.104410	1.383675	-1.097692	0.958196	-0.653246	0.2457
...	...	...	...	...	...	...	...	...	
405	-2.233398	-2.998710	-0.378282	0.497056	-0.144041	-1.423559	-0.648699	0.289676	0.1224
406	-2.923210	11.711044	5.856770	4.536946	0.801008	-0.007470	1.331167	-1.196007	0.6149
407	-2.716925	5.627624	3.129216	2.034017	-1.944248	1.328403	0.538597	-0.491240	0.2763
408	-2.511517	3.152825	2.056172	1.597862	-1.588072	0.221263	0.574015	-0.519261	0.3861
409	-2.493756	-2.801108	-0.163597	-0.006073	0.120255	1.392496	-1.301845	0.218497	-0.2664

410 rows × 18 columns

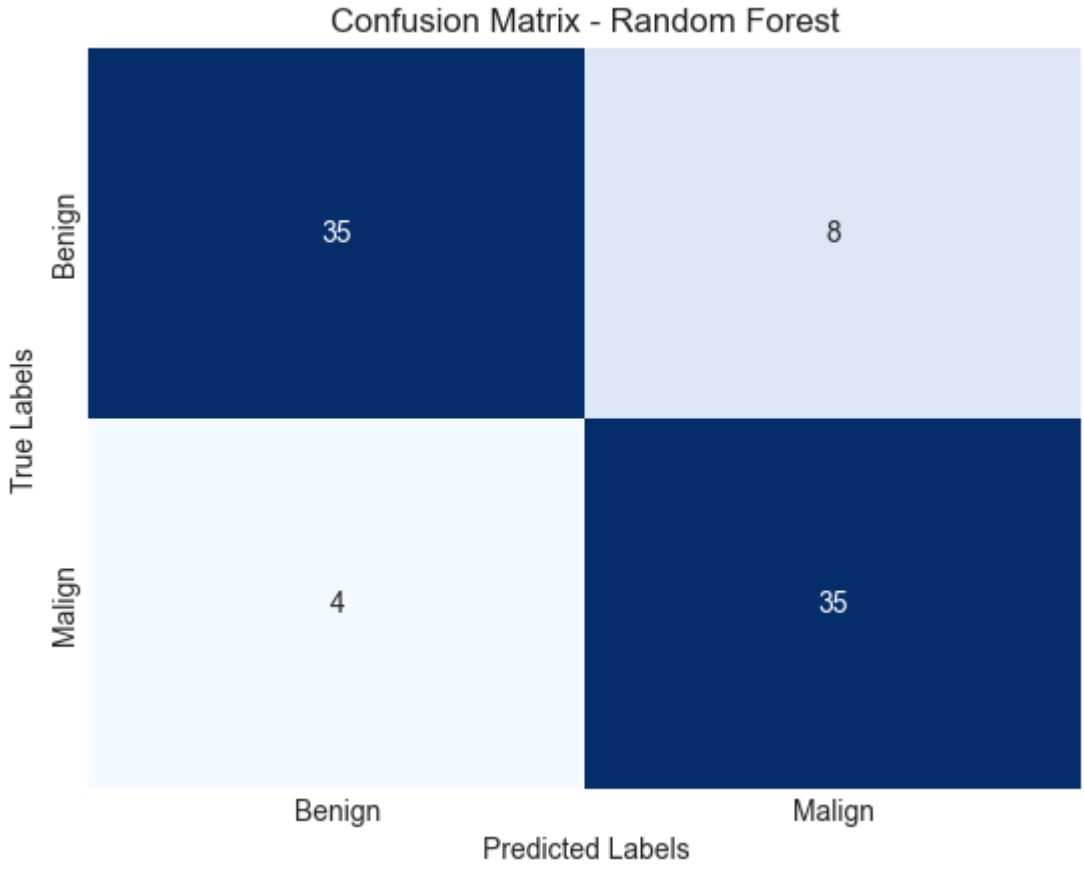
In [65]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = random_forest_kfold(fold_2d_pca)
th_rf_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
results_rf_2d.append(average_metrics_2d)
```

```
Número do Fold: 1
```



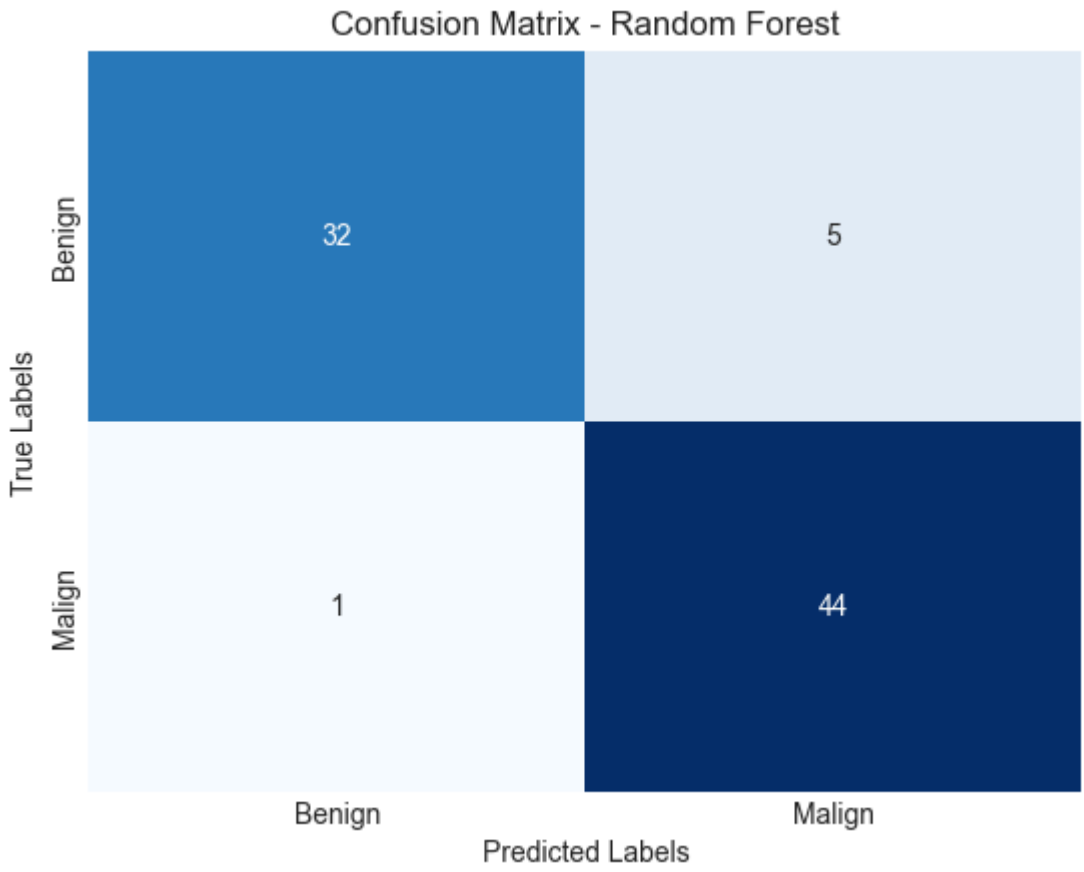
Loading [MathJax]/extensions/Safe.js

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.91      0.93        45
           1       0.90      0.95      0.92        37

    accuracy                           0.93        82
   macro avg       0.93      0.93      0.93        82
weighted avg       0.93      0.93      0.93        82

Número do Fold: 2
```

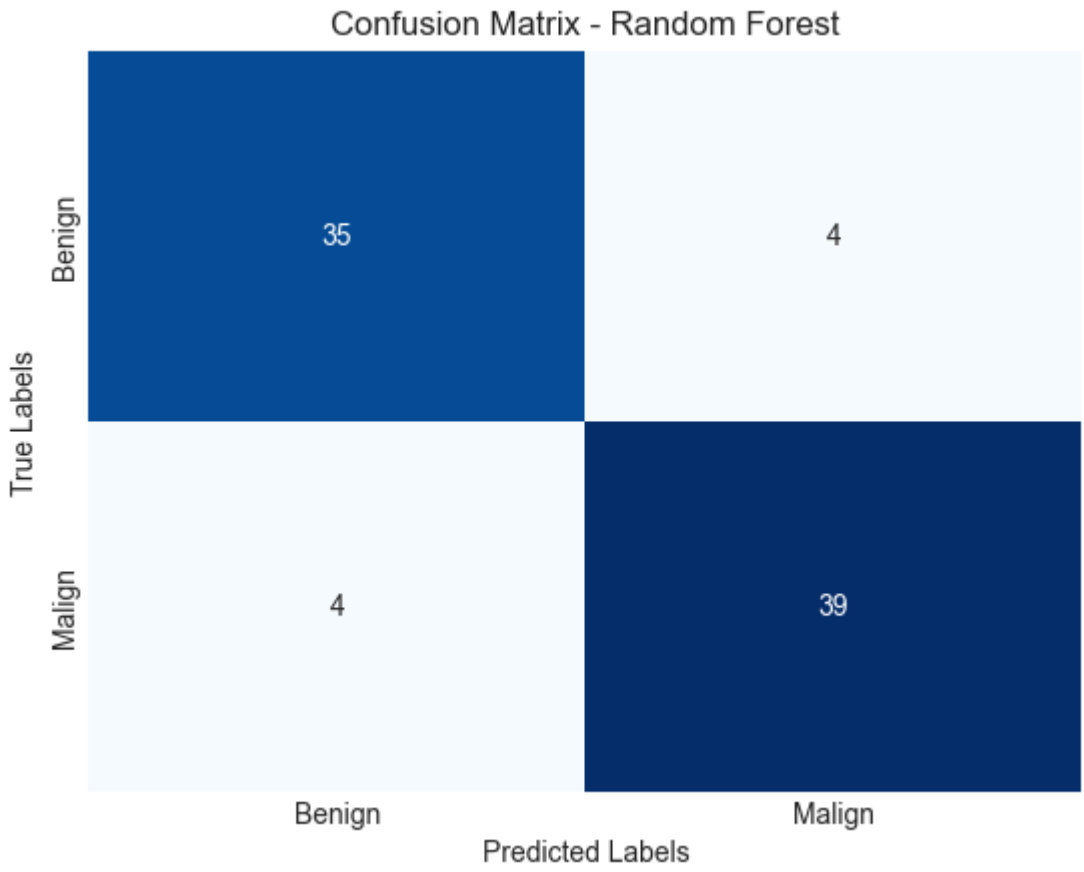



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.81      0.85        43
           1       0.81      0.90      0.85        39

    accuracy                           0.85        82
   macro avg       0.86      0.86      0.85        82
weighted avg       0.86      0.85      0.85        82

Número do Fold: 3
```

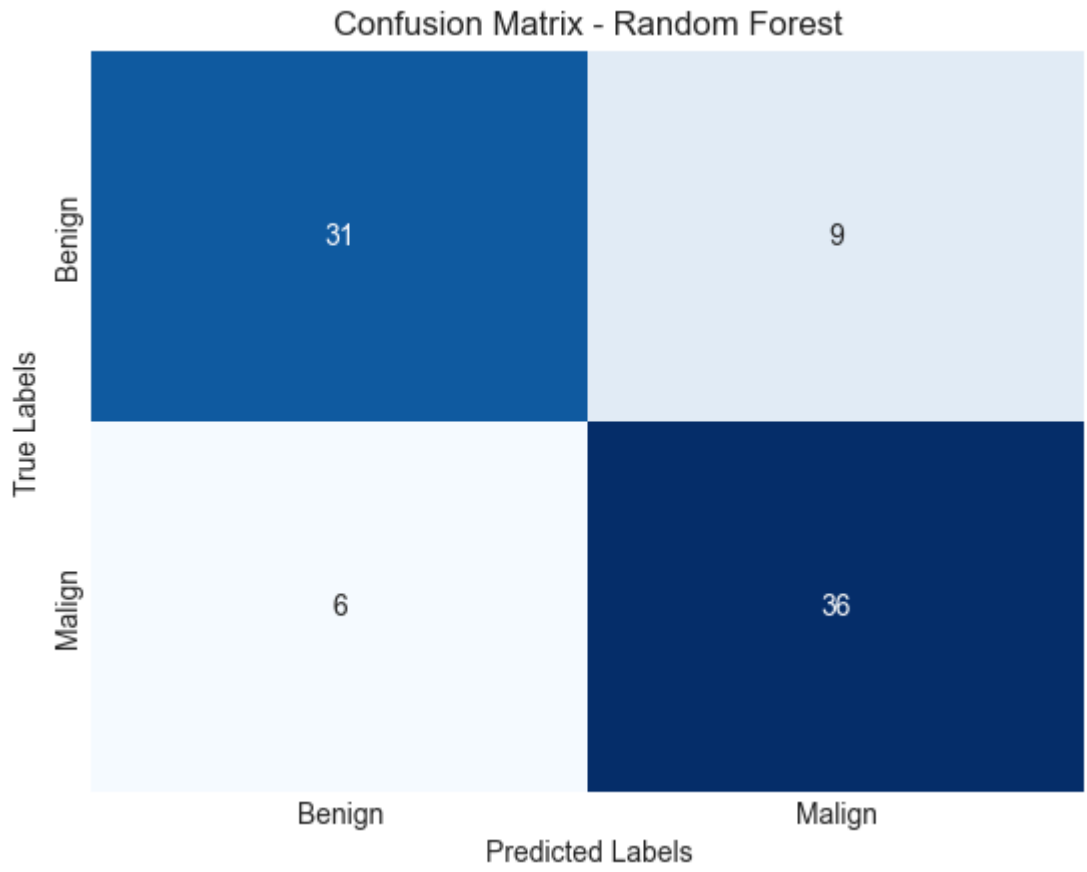
Loading [MathJax]/extensions/Safe.js



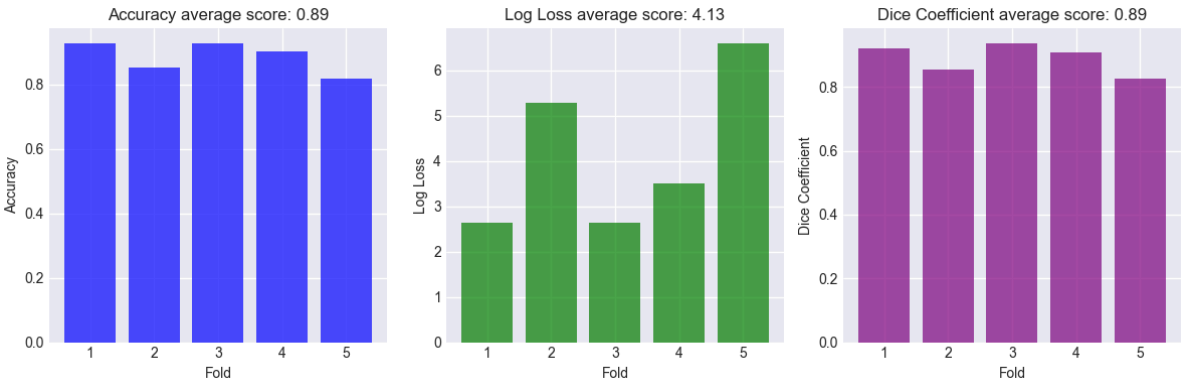

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.86      0.91        37
           1       0.90      0.98      0.94        45

    accuracy                           0.93        82
   macro avg       0.93      0.92      0.93        82
weighted avg       0.93      0.93      0.93        82

Número do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        39
           1       0.91      0.91      0.91        43

    accuracy                           0.90        82
   macro avg       0.90      0.90      0.90        82
weighted avg       0.90      0.90      0.90        82

Número do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.78      0.81        40
           1       0.80      0.86      0.83        42

    accuracy                           0.82        82
   macro avg       0.82      0.82      0.82        82
weighted avg       0.82      0.82      0.82        82
```



-> 3D

In [66]:
```
data_pca_3d
```

Loading [MathJax]/extensions/Safe.js

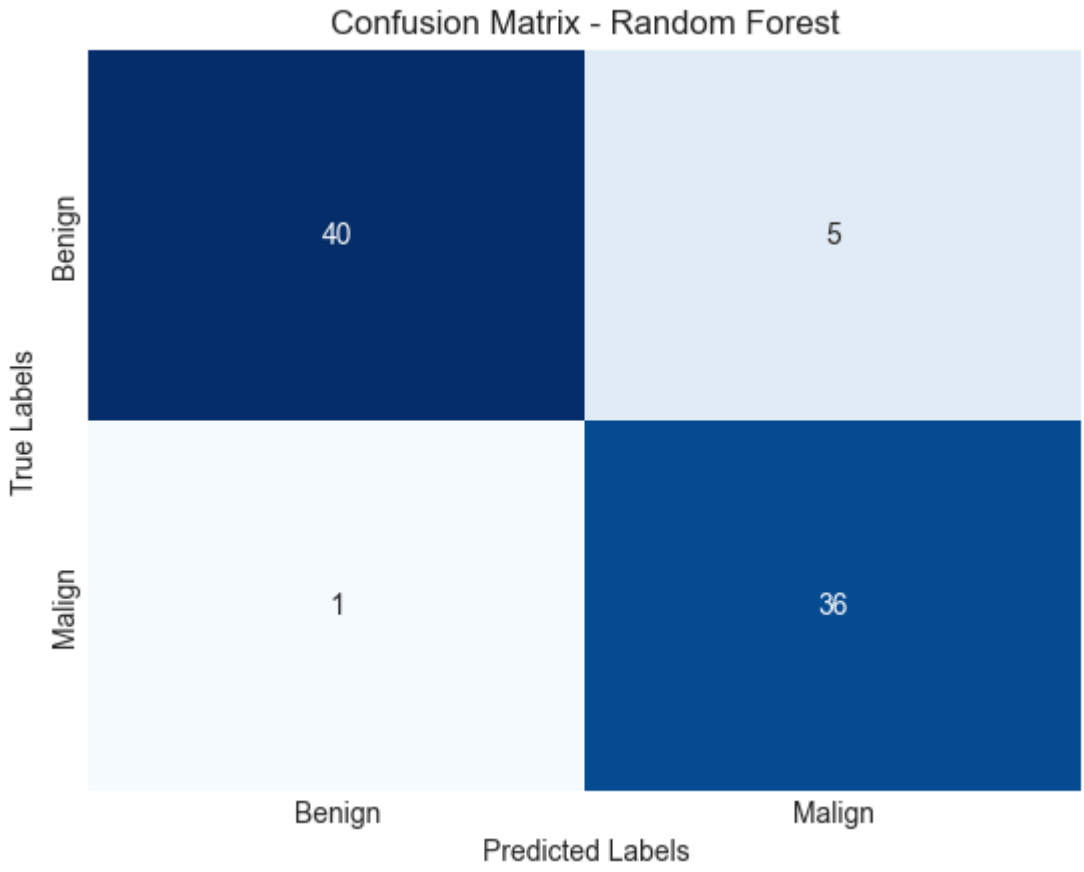
Out[66]:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	P
0	-0.782122	-4.117717	1.807559	3.238631	-2.126003	0.470964	0.041700	-1.212518	-0.2280
1	-2.459356	2.218242	-1.818623	2.089689	0.356737	1.117322	-0.424158	-0.750689	-0.4551
2	13.960910	0.629641	-1.754254	-1.768815	-1.563777	-2.868636	-2.049989	-1.101771	-1.3600
3	18.196420	-6.671389	-6.882279	1.530733	2.834788	2.809031	-2.197428	2.412911	1.9639
4	-2.639728	3.325768	-2.169896	0.994315	-0.182417	-0.915270	1.220950	1.471481	0.1514
...	...	...	...	...	...	...	...	...	
405	-2.682271	3.198533	-2.295293	1.549300	0.057934	-0.301314	0.212454	-0.055443	0.9605
406	-2.269498	-5.455861	4.720836	7.225682	2.834643	-3.934610	3.135748	-0.186134	0.6010
407	-2.505729	-0.167273	0.442583	2.612708	2.009912	0.850195	-0.439097	0.373260	-1.7192
408	-2.686433	0.246159	0.531057	1.261710	1.665685	0.842926	-0.394356	0.287286	-1.0636
409	-2.196771	3.355261	-3.609834	4.514218	-0.464621	0.370936	0.322757	-0.490927	1.1526

410 rows × 18 columns

In [67]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = random_forest_kfold(fold_3d_pca)
th_rf_3d.append([accuracies_3d,log_losses_3d,dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
results_rf_3d.append(average_metrics_3d)
```

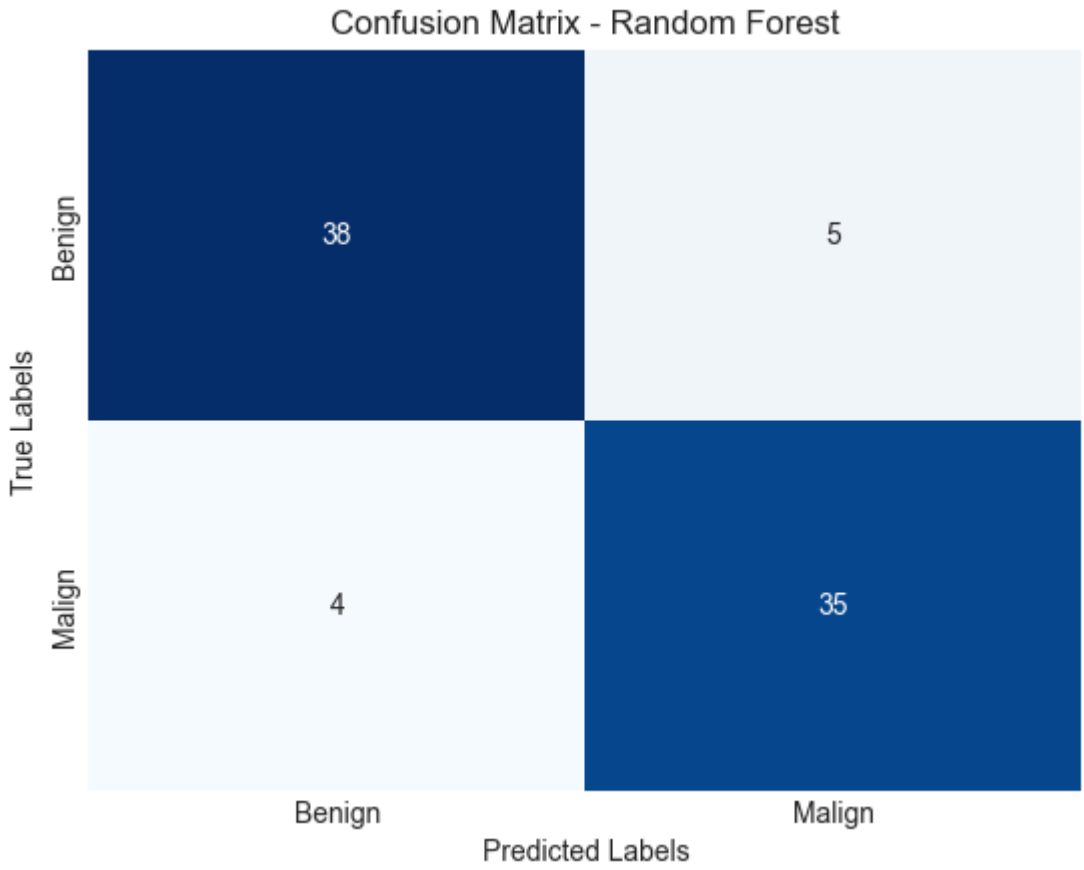
```
Número do Fold: 1
```



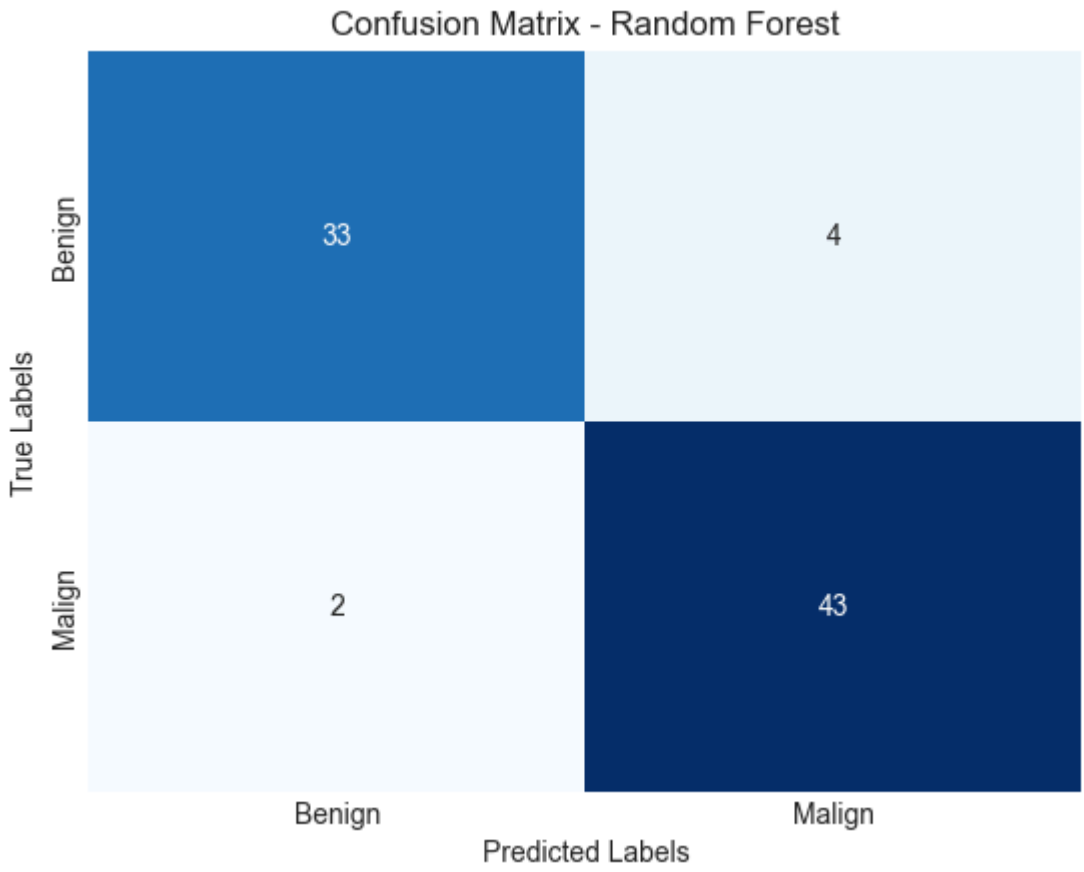

```
<Figure size 500x500 with 0 Axes>
```

Loading [MathJax]/extensions/Safe.js

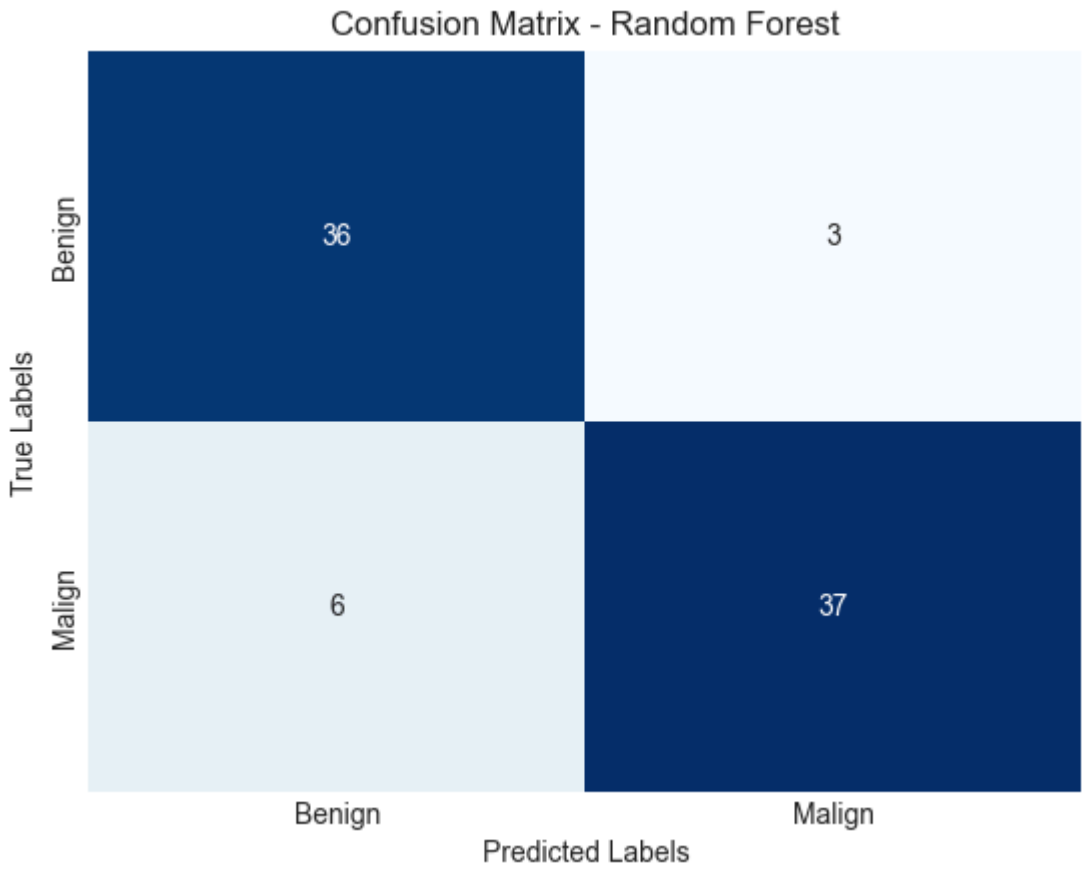
```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.89      0.93        45
           1       0.88      0.97      0.92        37

    accuracy                           0.93        82
   macro avg       0.93      0.93      0.93        82
weighted avg       0.93      0.93      0.93        82

Número do Fold: 2
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89        43
           1       0.88      0.90      0.89        39

    accuracy                           0.89        82
   macro avg       0.89      0.89      0.89        82
weighted avg       0.89      0.89      0.89        82

Número do Fold: 3
```

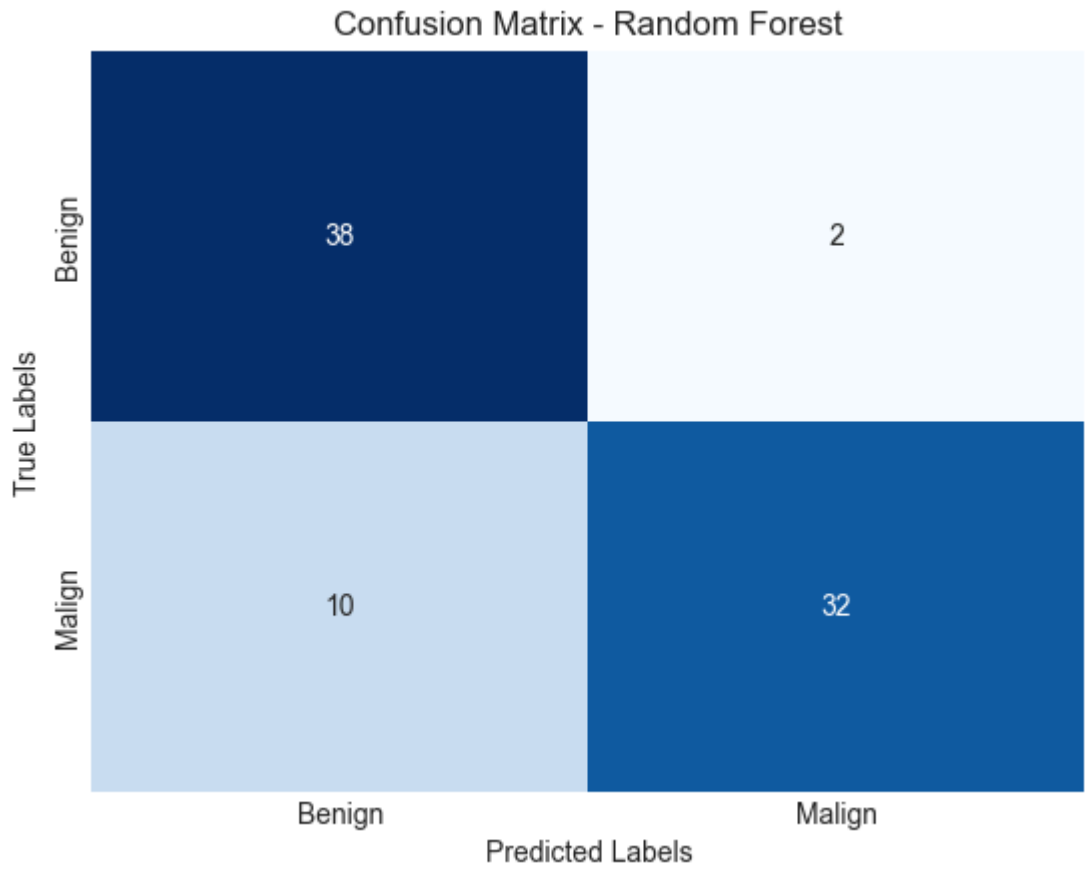



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.89      0.92        37
           1       0.91      0.96      0.93        45

    accuracy                           0.93        82
   macro avg       0.93      0.92      0.93        82
weighted avg       0.93      0.93      0.93        82

Número do Fold: 4
```

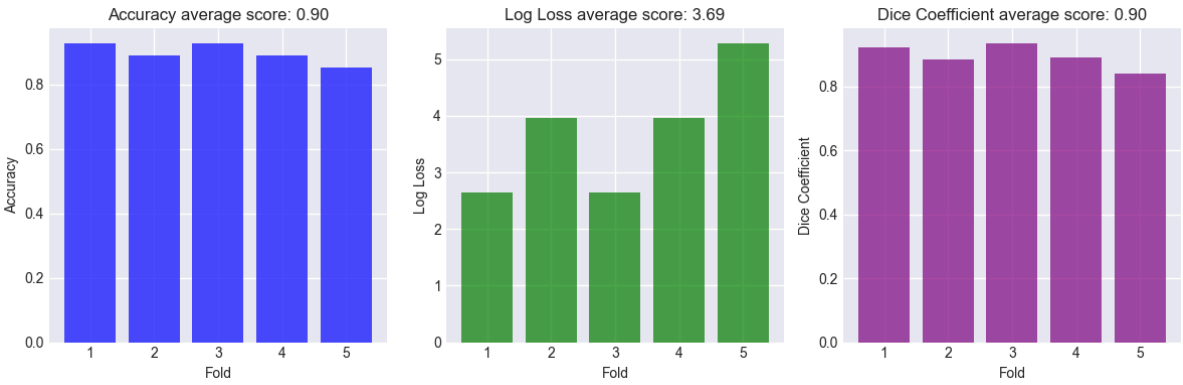



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.92      0.89        39
           1       0.93      0.86      0.89        43

    accuracy                           0.89        82
   macro avg       0.89      0.89      0.89        82
weighted avg       0.89      0.89      0.89        82

Número do Fold: 5
```

Loading [MathJax]/extensions/Safe.js




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.95      0.86        40
           1       0.94      0.76      0.84        42

    accuracy                           0.85        82
   macro avg       0.87      0.86      0.85        82
weighted avg       0.87      0.85      0.85        82
```



Random Forest + t-test

[Voltar a Random Forest]

Repetimos o processo anterior, utilizando este dataset para o treino e teste do modelo Random Forest.

-> 2D

In [68]:
```
data_ttest_2d
```

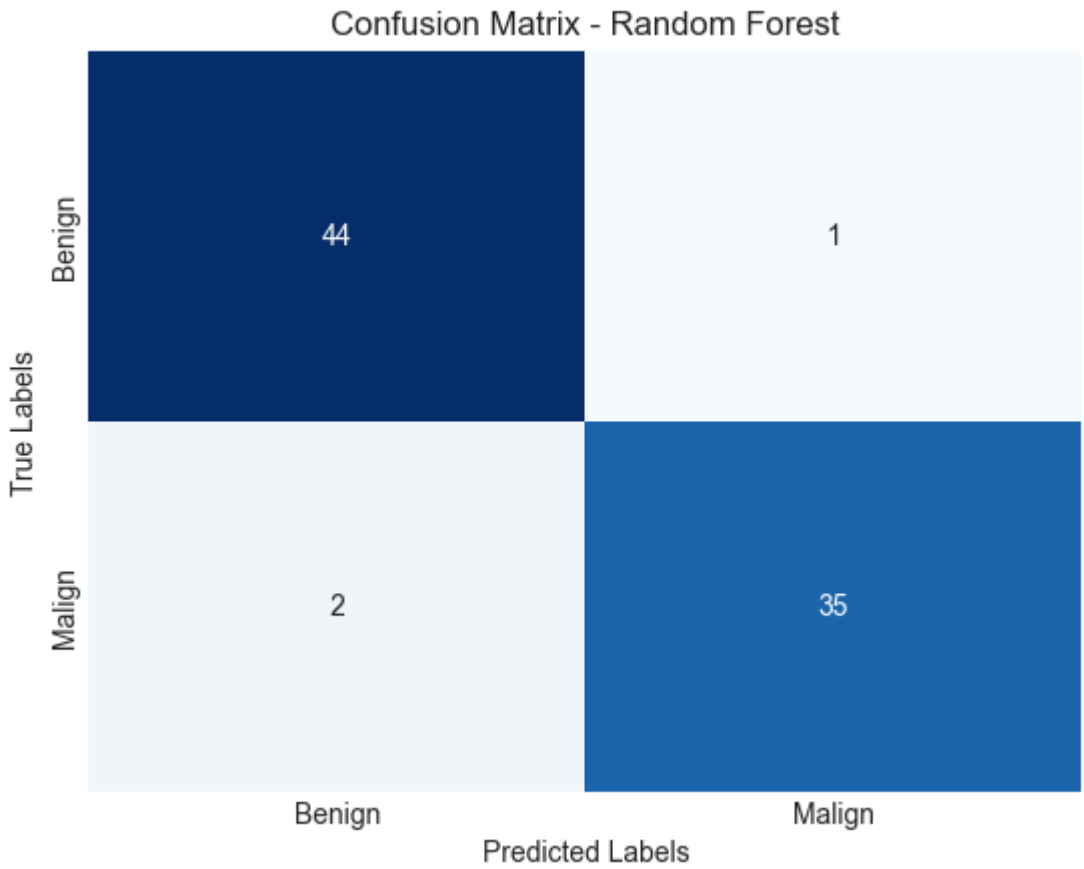
Out[68]:

	Calcification	original_shape2D_Sphericity	original_shape2D_MajorAxisLength	original_shape2D
0	6	0.116476	57.542760	
1	6	0.237705	17.090541	
2	3	0.293095	9.244896	
3	6	0.145259	40.718270	
4	3	0.288620	10.327956	
...	...	...	...	
405	6	0.245244	13.852837	
406	6	0.112044	61.575217	
407	6	0.138573	51.164214	
408	6	0.154955	41.452617	
409	6	0.276215	12.494295	

410 rows × 49 columns

In [69]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = random_forest_kfold(fold_2d_ttest)
th_rf_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
results_rf_2d.append(average_metrics_2d)
```

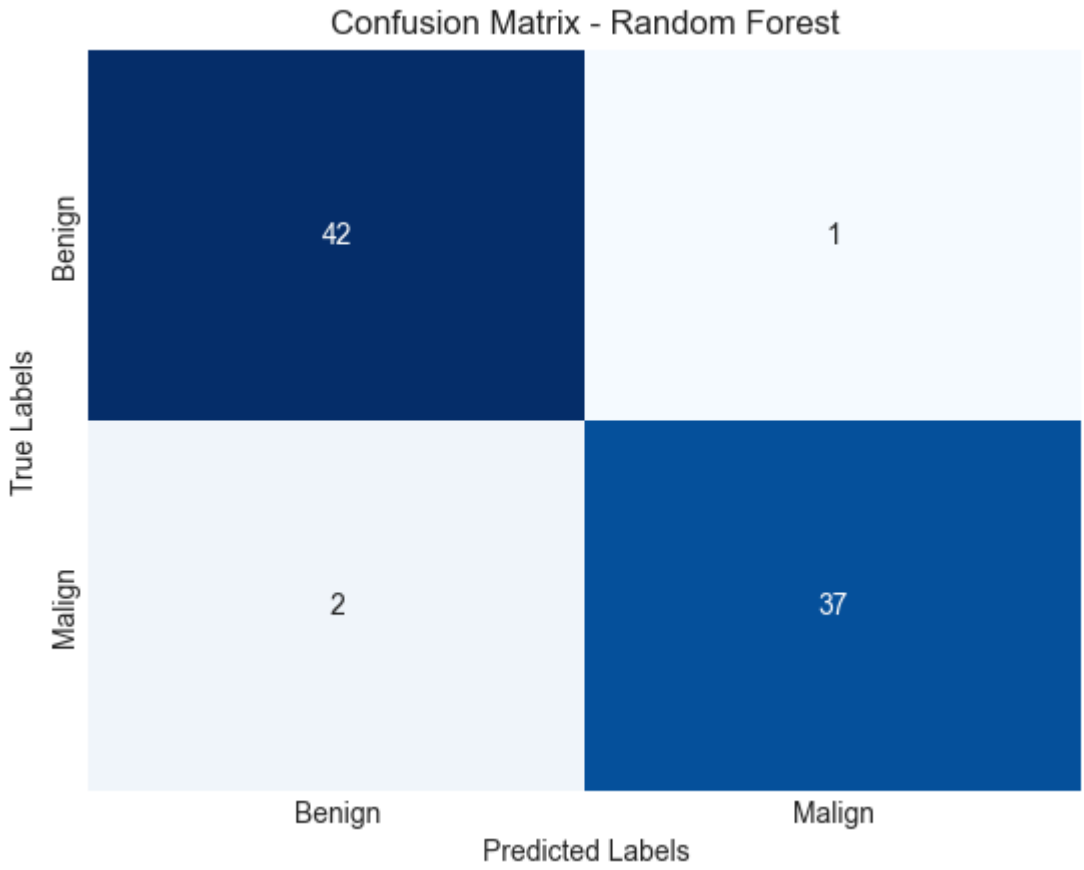
```
Número do Fold: 1
```



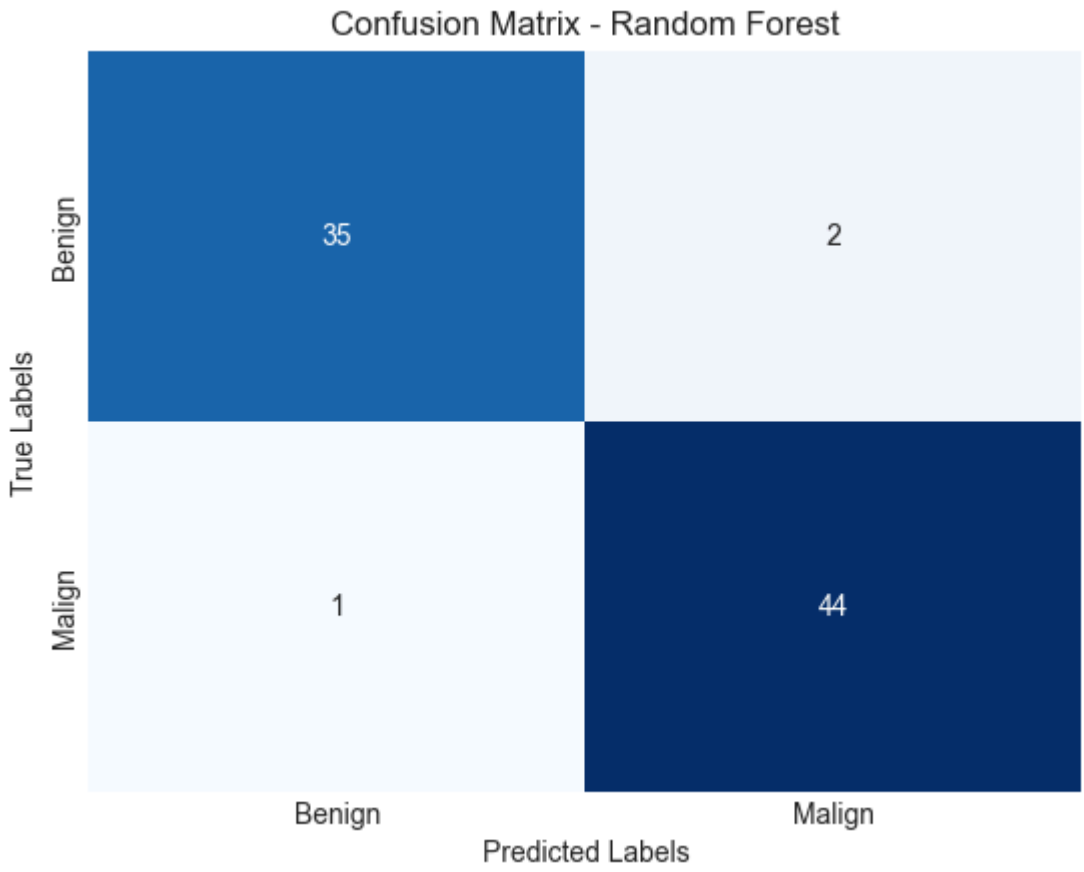
Loading [MathJax]/extensions/Safe.js

```
0x500 with 0 Axes>
```

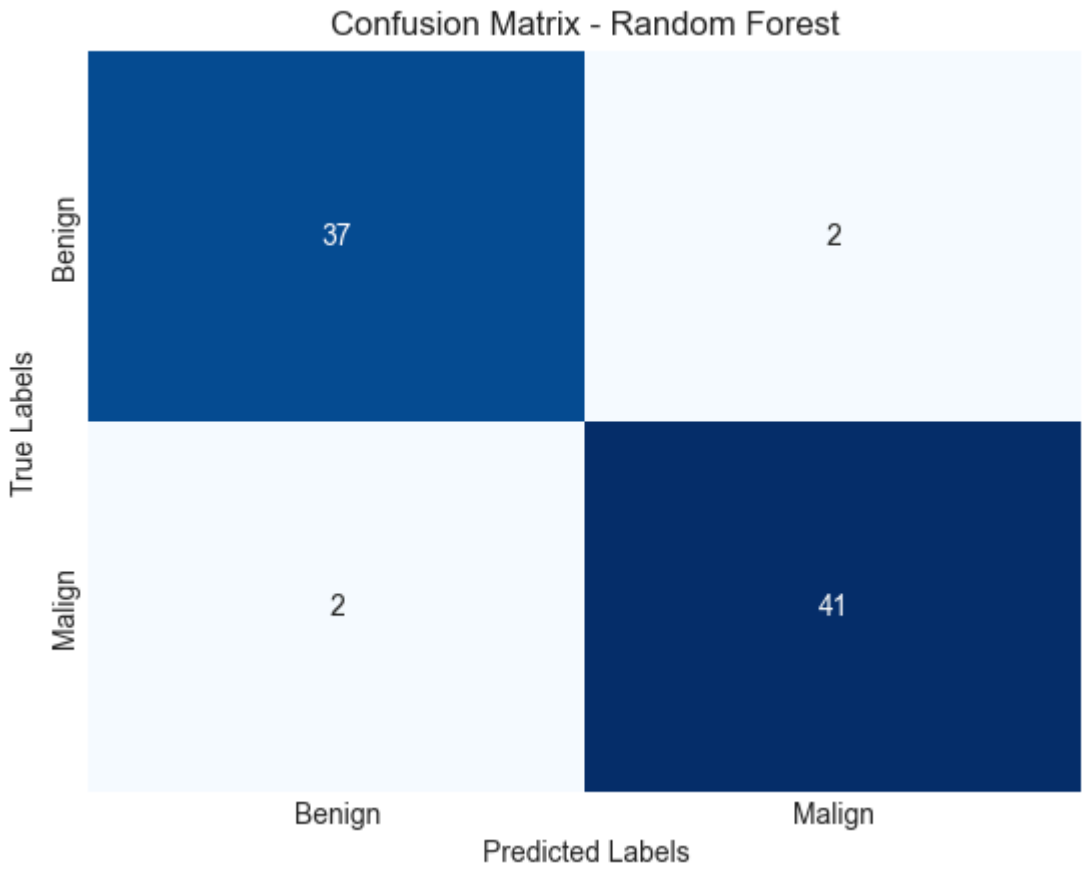
```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97        45
           1       0.97      0.95      0.96        37

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Número do Fold: 2
```



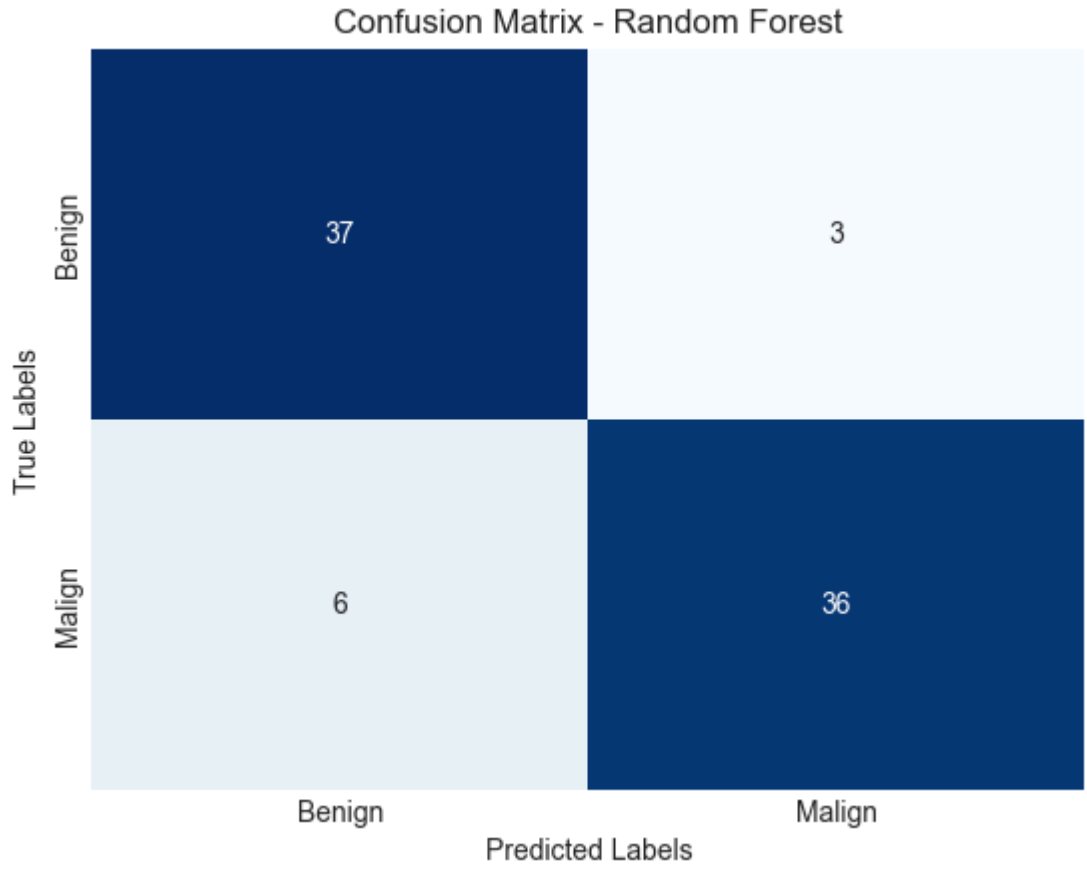

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.97        43
           1       0.97      0.95      0.96        39

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Número do Fold: 3
```



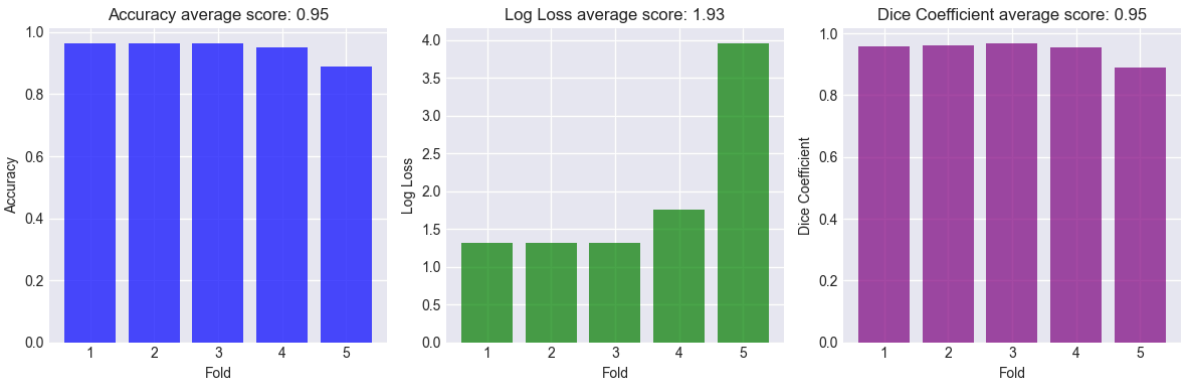

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        37
           1       0.96      0.98      0.97        45

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Número do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        39
           1       0.95      0.95      0.95        43

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.93      0.89        40
           1       0.92      0.86      0.89        42

    accuracy                           0.89        82
   macro avg       0.89      0.89      0.89        82
weighted avg       0.89      0.89      0.89        82
```



-> 3D

In [70]:
```
data_ttest_3d
```

Loading [MathJax]/extensions/Safe.js

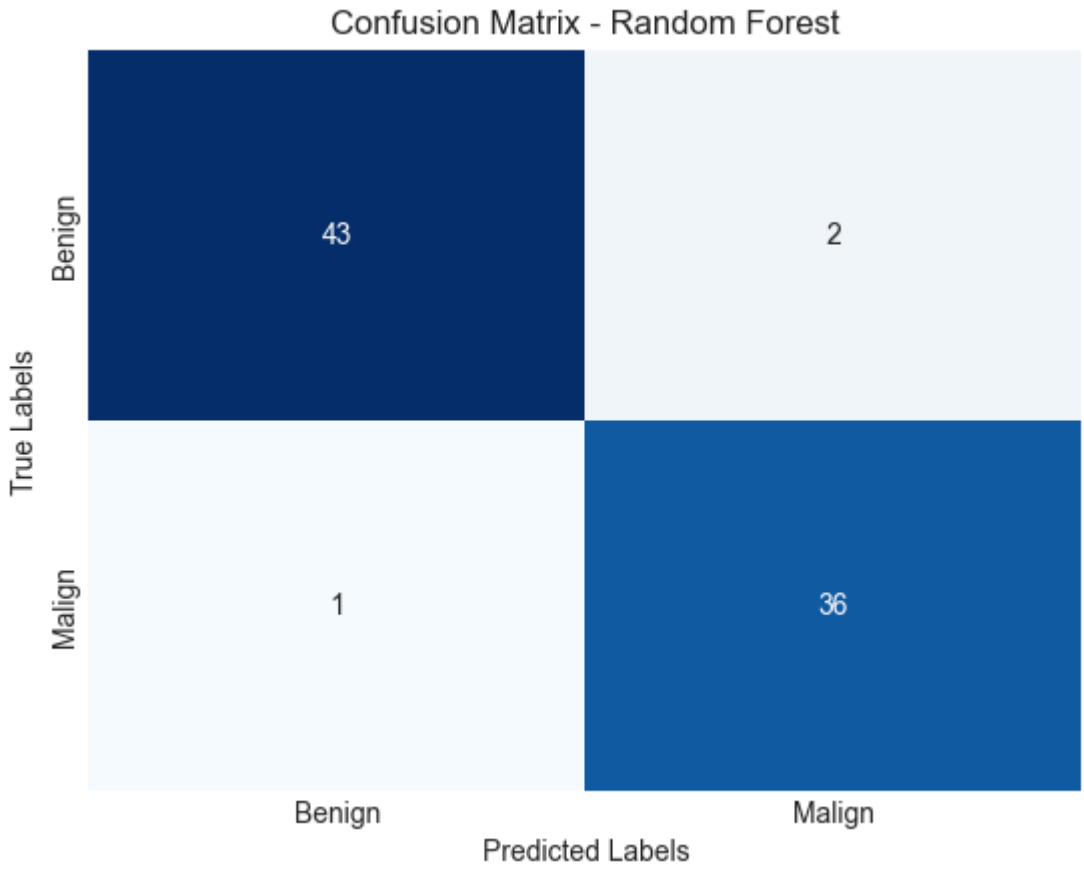
Out[70]:

	Calcification	Spiculation	Lobulation	Margin	Subtlety	original_glrlm_GrayLevelNonUniformit
0	6	5	3	4	5	757.88036
1	6	2	2	3	5	88.38461
2	3	1	1	5	2	34.51526
3	6	5	1	3	5	437.07015
4	3	1	1	5	3	33.84615
...	...	...	...	...	...	
405	6	1	1	4	3	41.76923
406	6	2	2	4	5	1239.76923
407	6	1	3	4	5	573.30769
408	6	1	1	3	5	444.76923
409	6	1	1	2	4	47.84615

410 rows × 58 columns

In [71]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = random_forest_kfold(fold_3d_ttest)
th_rf_3d.append([accuracies_3d,log_losses_3d,dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
results_rf_3d.append(average_metrics_3d)
```

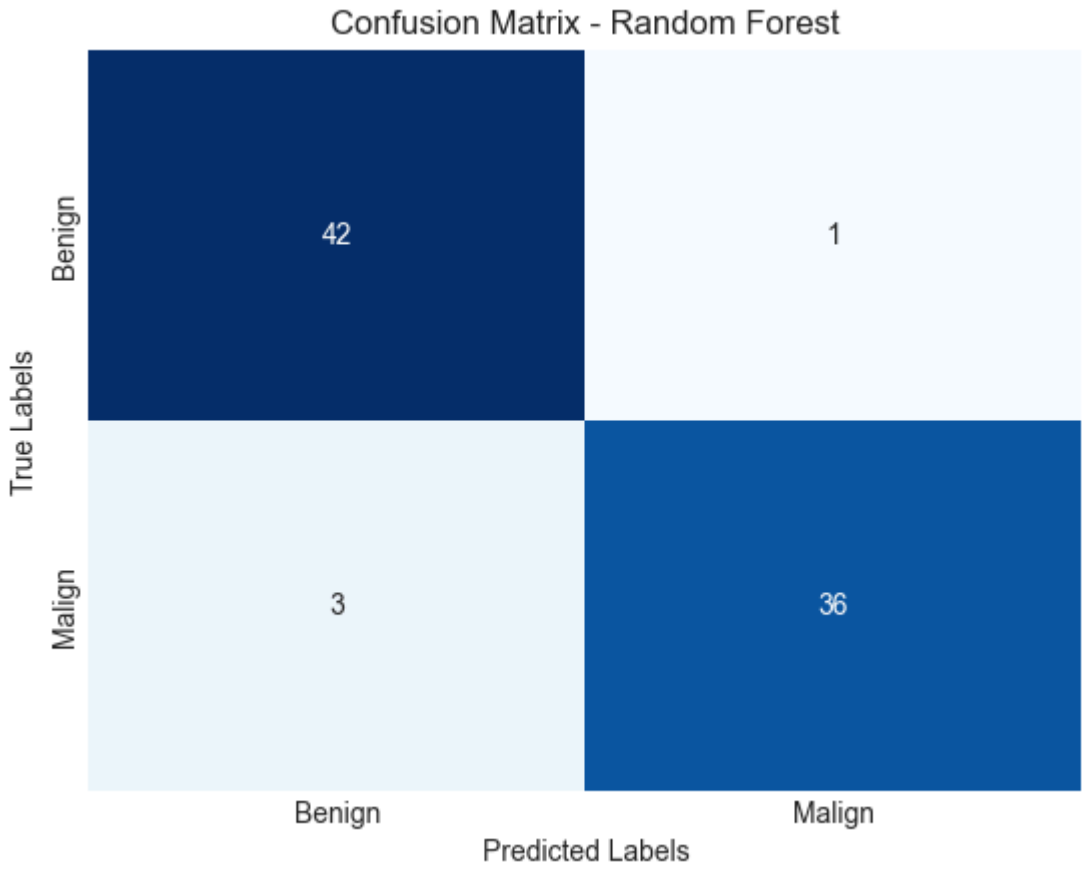
```
Número do Fold: 1
```



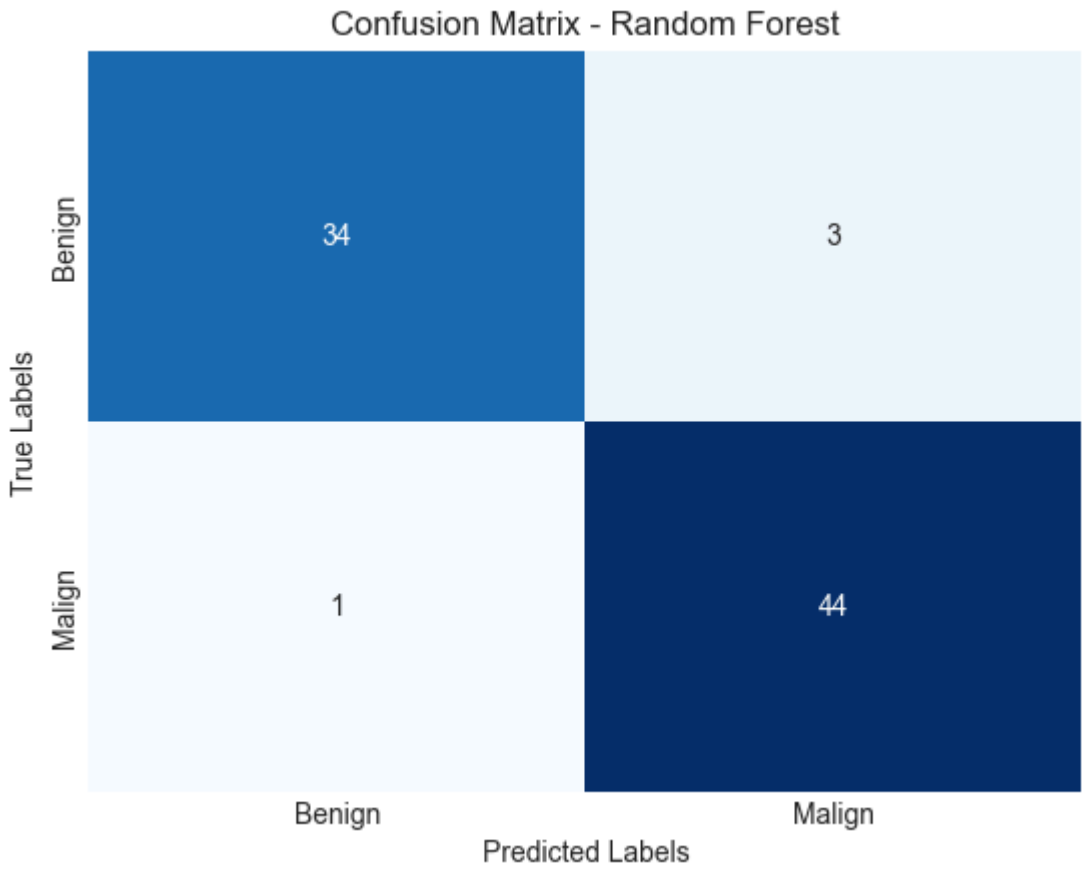

```
<Figure size 500x500 with 0 Axes>
```

Loading [MathJax]/extensions/Safe.js

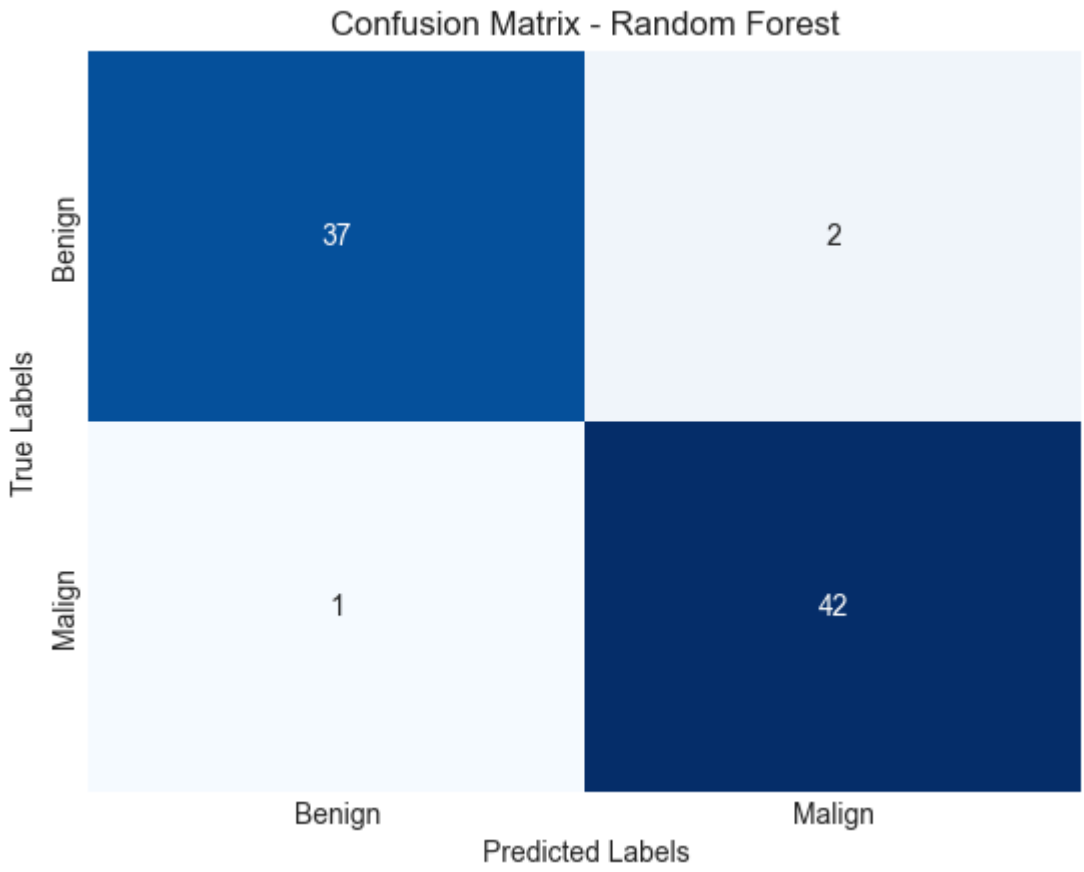
```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        45
           1       0.95      0.97      0.96        37

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Número do Fold: 2
```



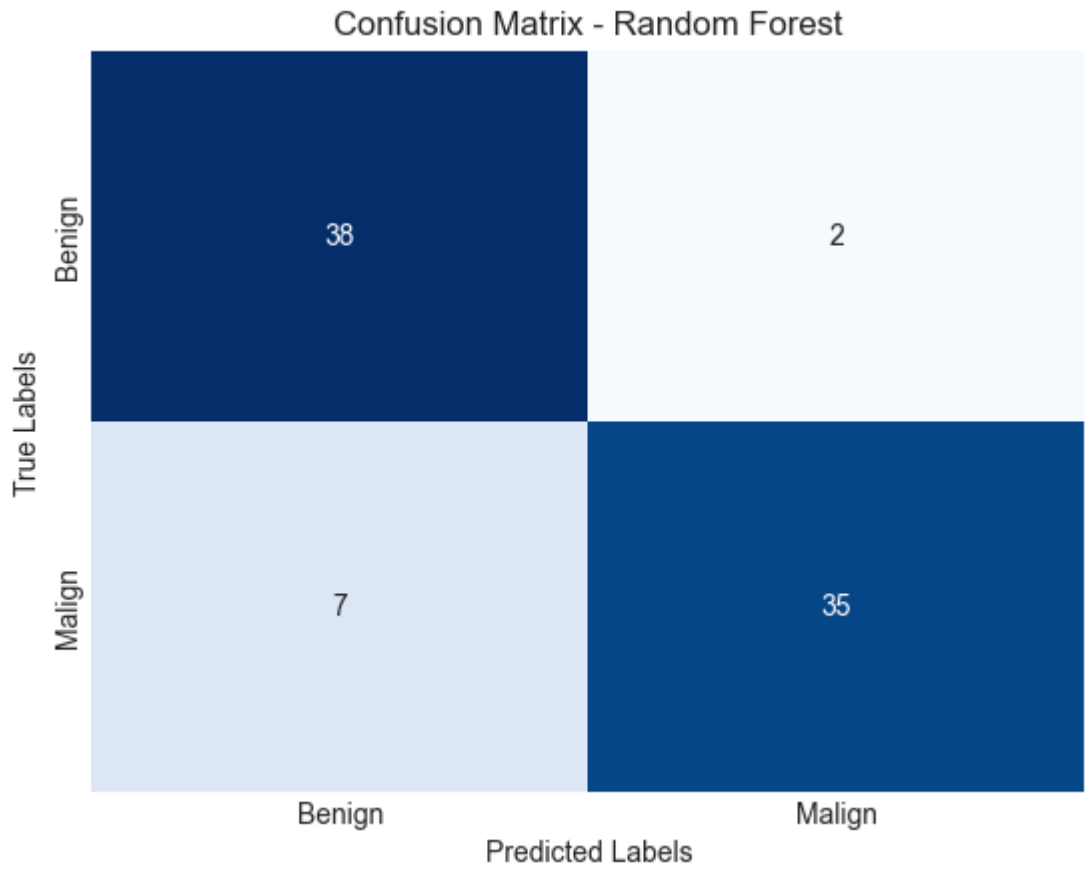

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        43
           1       0.97      0.92      0.95        39

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número do Fold: 3
```



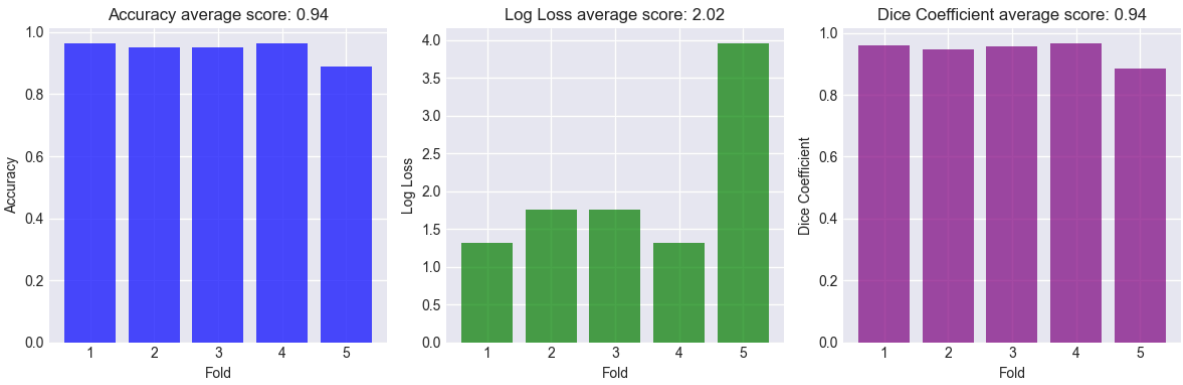

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.92      0.94        37
           1       0.94      0.98      0.96        45

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        39
           1       0.95      0.98      0.97        43

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Número do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.95      0.89        40
           1       0.95      0.83      0.89        42

    accuracy                           0.89        82
   macro avg       0.90      0.89      0.89        82
weighted avg       0.90      0.89      0.89        82
```



Random Forest + Random Forest (seleção de Features)

[Voltar a Random Forest]

Por fim, repetimos o processo para o dataset obtido pelo Random Forest, usado na seleção das features.

-> 2D

In [72]:
```
data_rf_2d
```

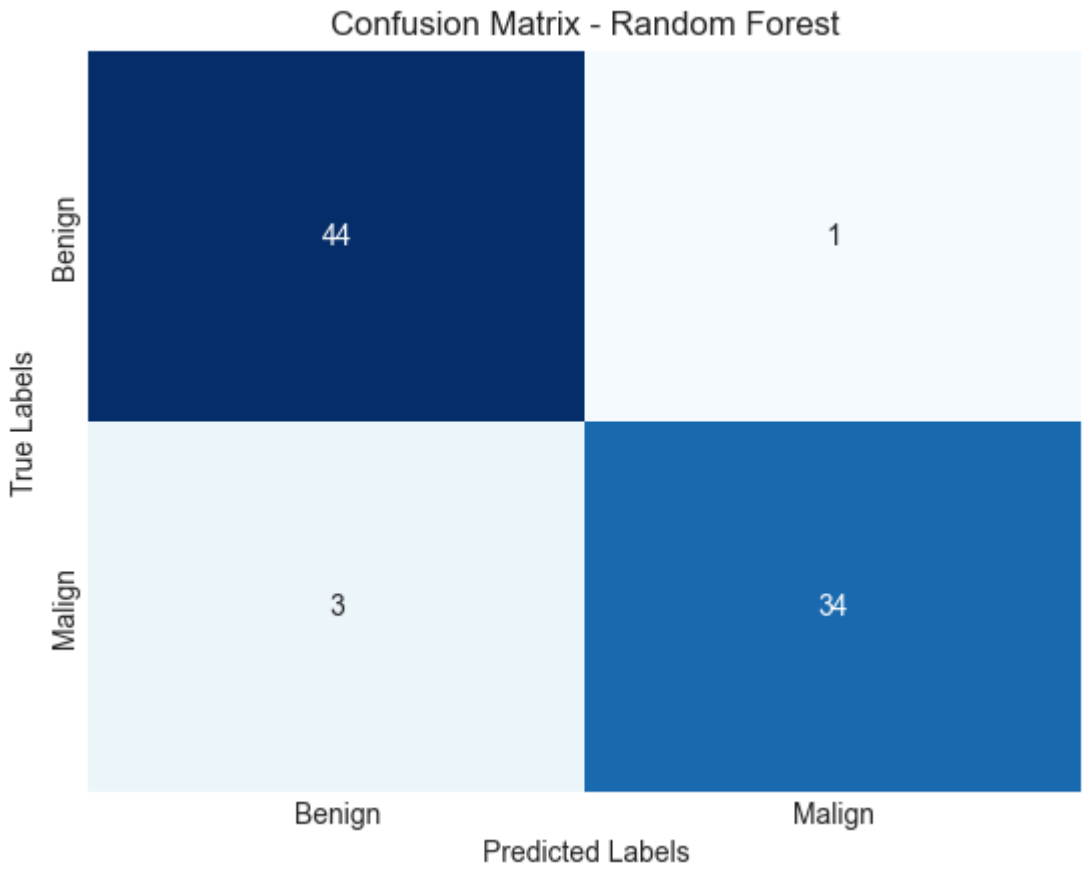
Out[72]:

	Calcification	Lobulation	Spiculation	original_gldm_GrayLevelNonUniformity	original_shape2D
0	6	3	5	160.036585	
1	6	2	2	33.000000	
2	3	1	1	12.142857	
3	6	1	5	43.079208	
4	3	1	1	18.000000	
...	...	...	...	...	
405	6	1	1	24.000000	
406	6	2	2	157.000000	
407	6	3	1	101.000000	
408	6	1	1	79.000000	
409	6	1	1	21.000000	

410 rows × 47 columns

In [73]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = random_forest_kfold(fold_2d_rf)
th_rf_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
results_rf_2d.append(average_metrics_2d)
```

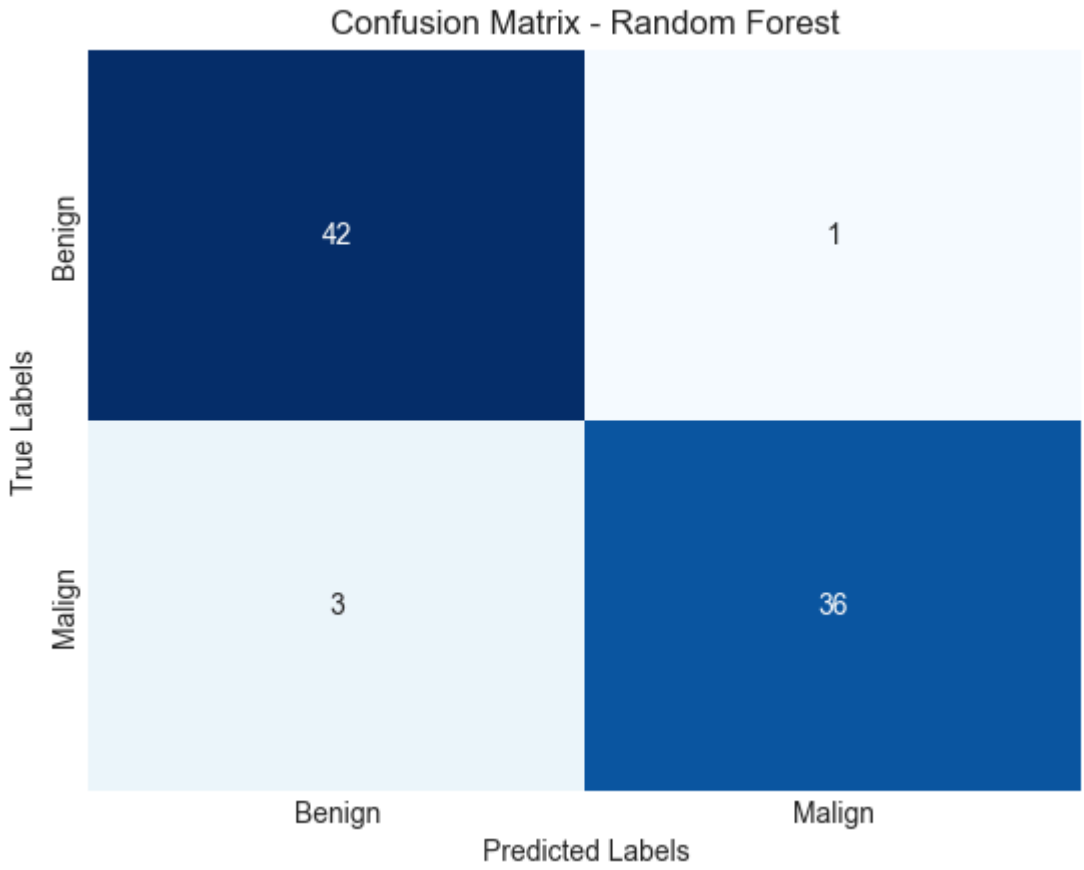
```
Número do Fold: 1
```



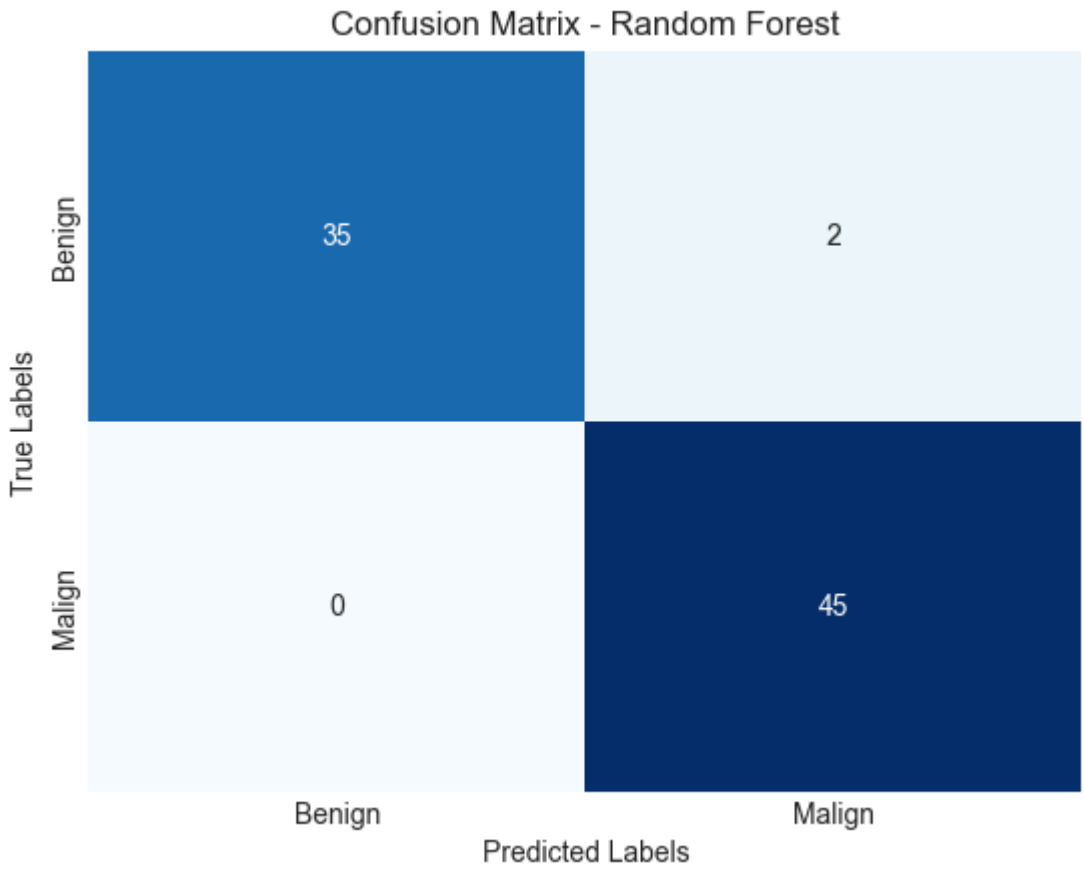
Loading [MathJax]/extensions/Safe.js

```
0x500 with 0 Axes>
```

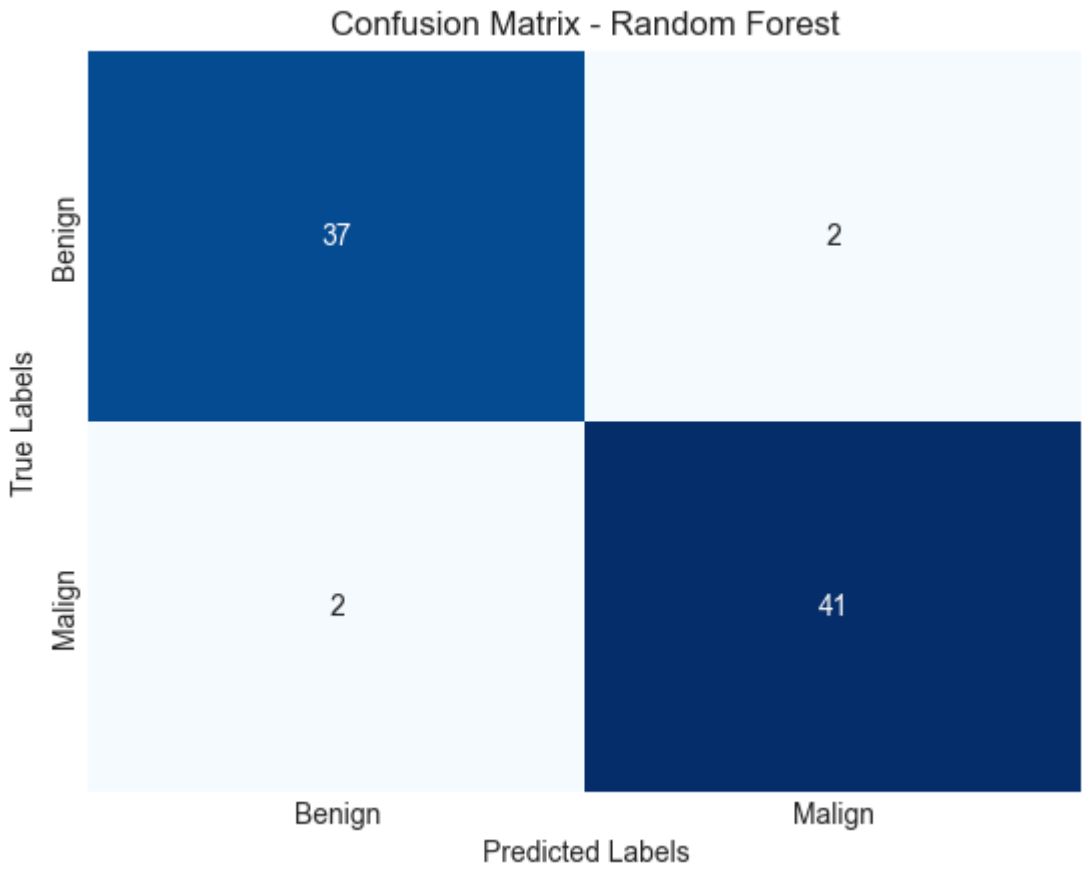
```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.98      0.96        45
           1       0.97      0.92      0.94        37

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número do Fold: 2
```



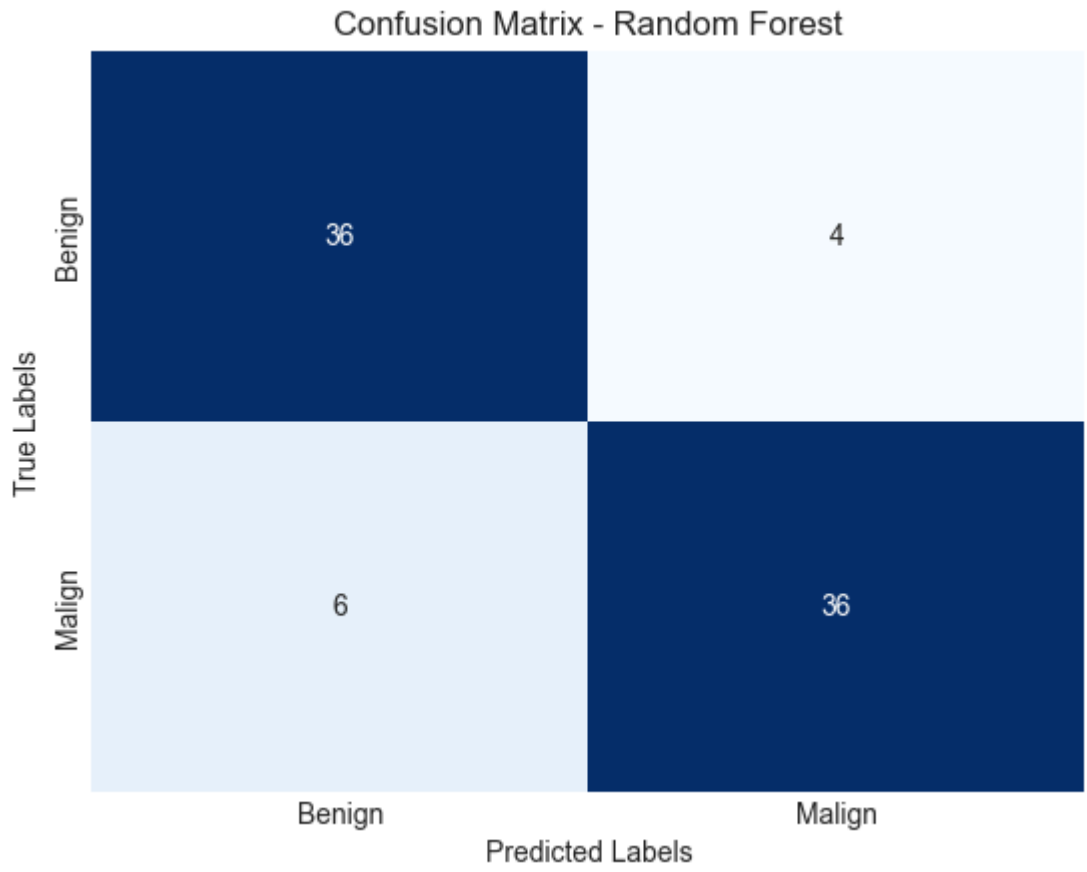

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        43
           1       0.97      0.92      0.95        39

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número do Fold: 3
```



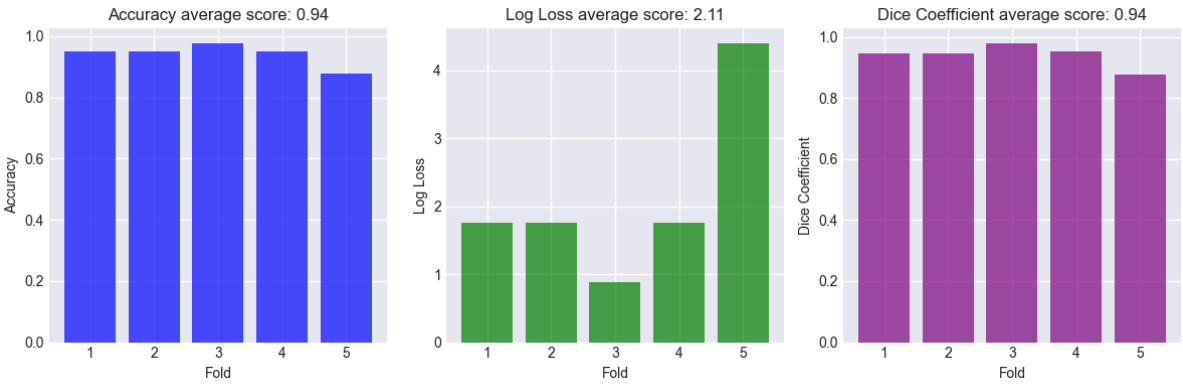

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        37
           1       0.96      1.00      0.98        45

    accuracy                           0.98        82
   macro avg       0.98      0.97      0.98        82
weighted avg       0.98      0.98      0.98        82

Número do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        39
           1       0.95      0.95      0.95        43

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88        40
           1       0.90      0.86      0.88        42

    accuracy                           0.88        82
   macro avg       0.88      0.88      0.88        82
weighted avg       0.88      0.88      0.88        82
```



-> 3D

In [74]:
```
data_rf_3d
```

Loading [MathJax]/extensions/Safe.js

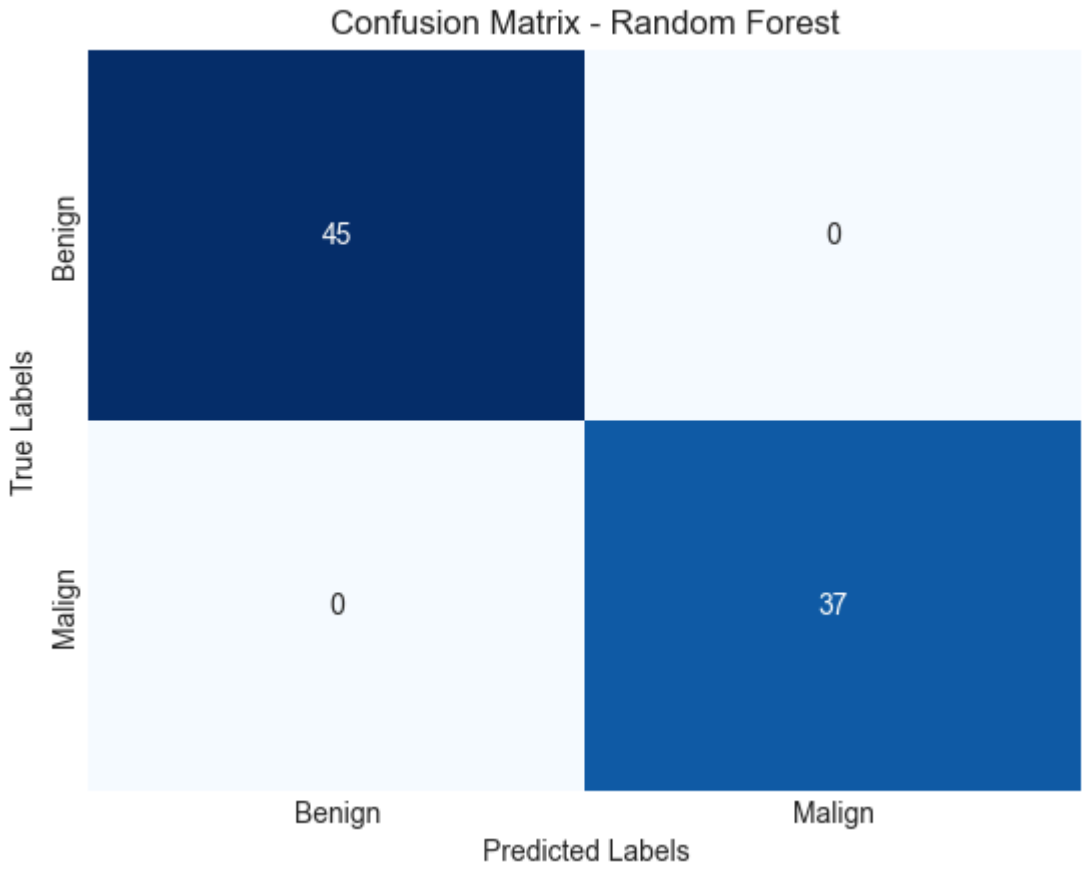
Out[74]:

	Calcification	Spiculation	Lobulation	Margin	original_glrlm_GrayLevelNonUniformity	origina
0	6	5	3	4	757.880362	
1	6	2	2	3	88.384615	
2	3	1	1	5	34.515260	
3	6	5	1	3	437.070150	
4	3	1	1	5	33.846154	
...	...	...	...	...	...	
405	6	1	1	4	41.769231	
406	6	2	2	4	1239.769231	
407	6	1	3	4	573.307692	
408	6	1	1	3	444.769231	
409	6	1	1	2	47.846154	

410 rows × 53 columns

In [75]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = random_forest_kfold(fold_3d_rf)
th_rf_3d.append([accuracies_3d,log_losses_3d,dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
results_rf_3d.append(average_metrics_3d)
```

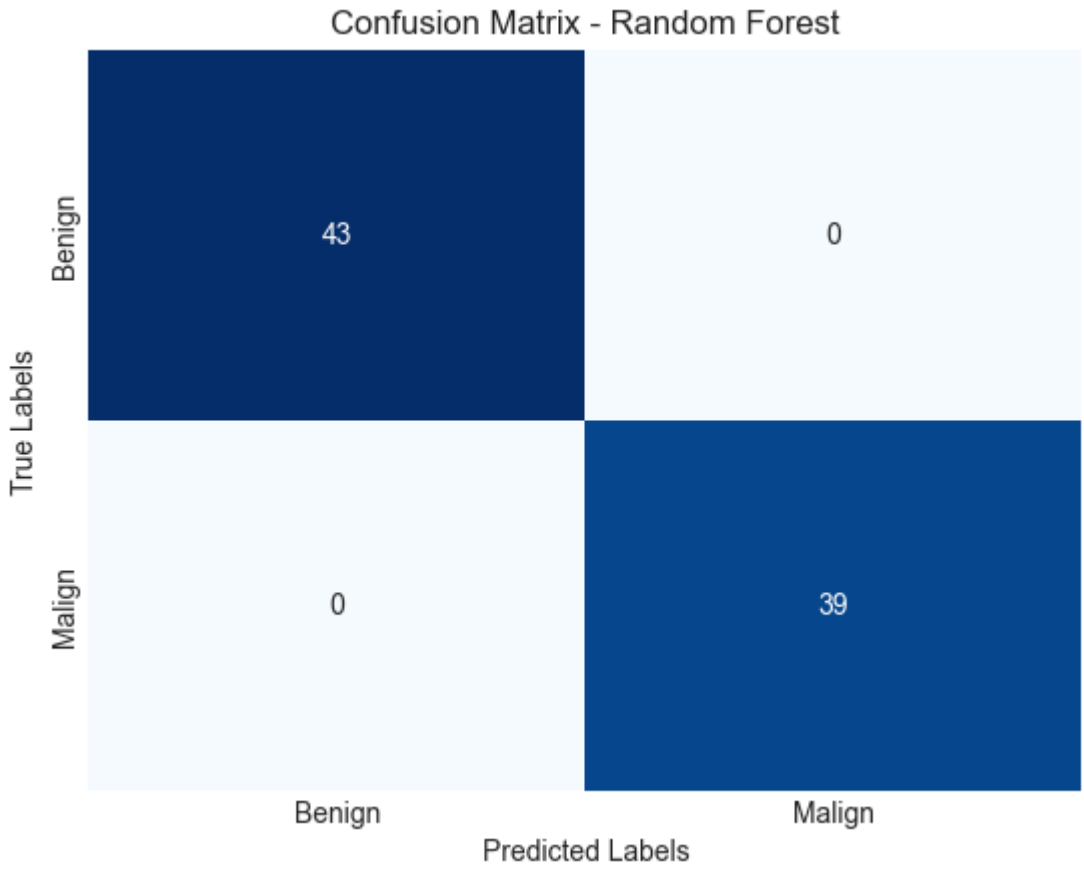
```
Número do Fold: 1
```



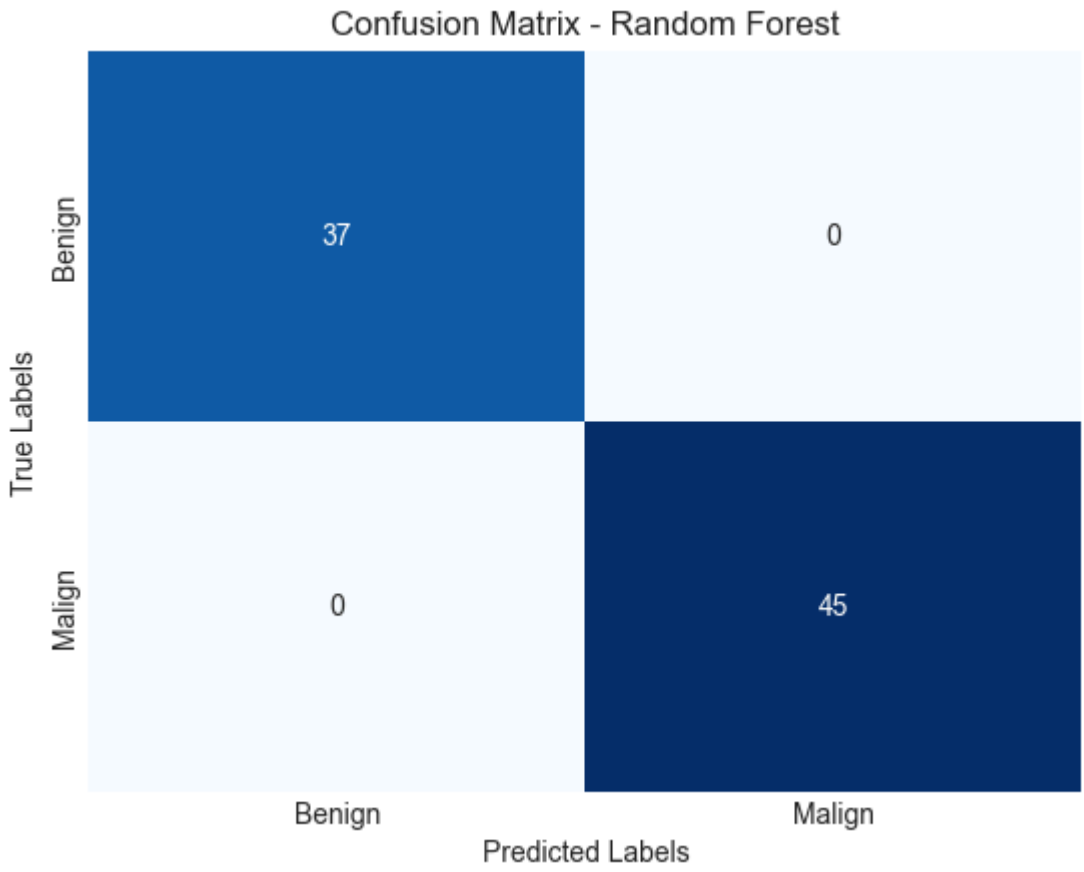

```
<Figure size 500x500 with 0 Axes>
```

Loading [MathJax]/extensions/Safe.js

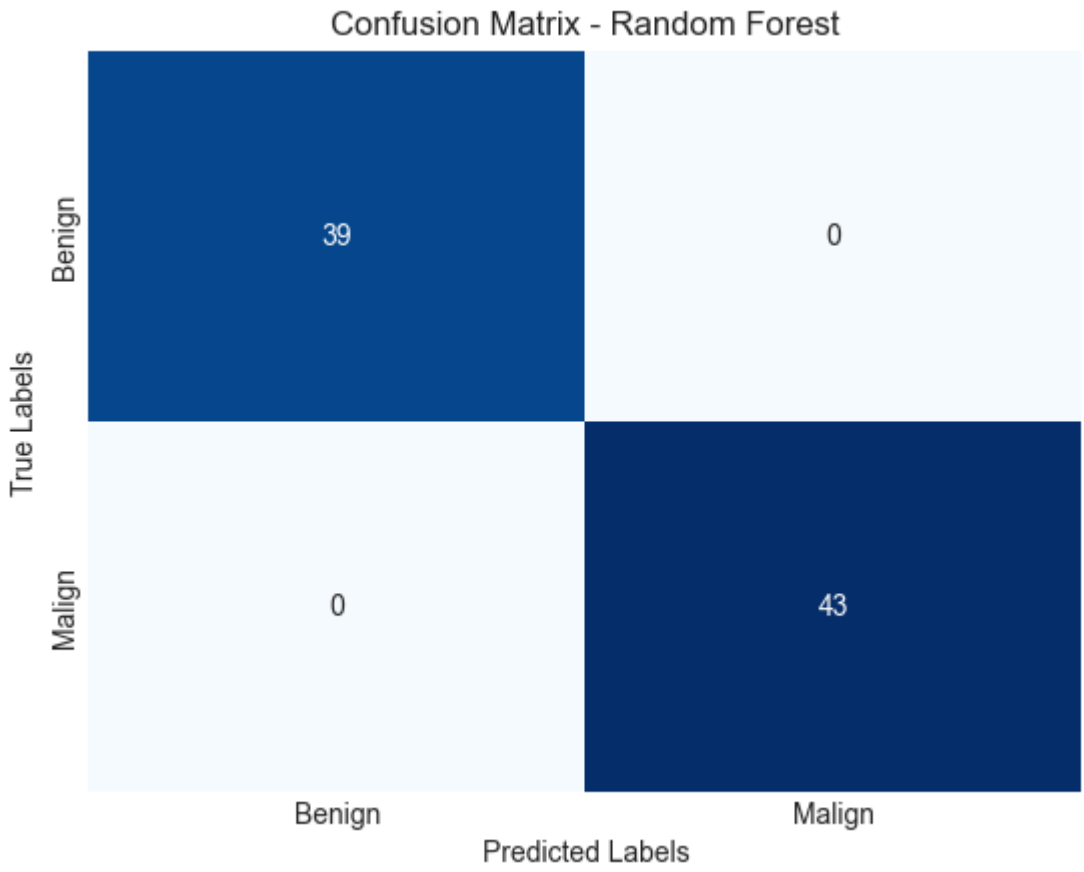
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       1.00      1.00      1.00        37

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número do Fold: 2
```



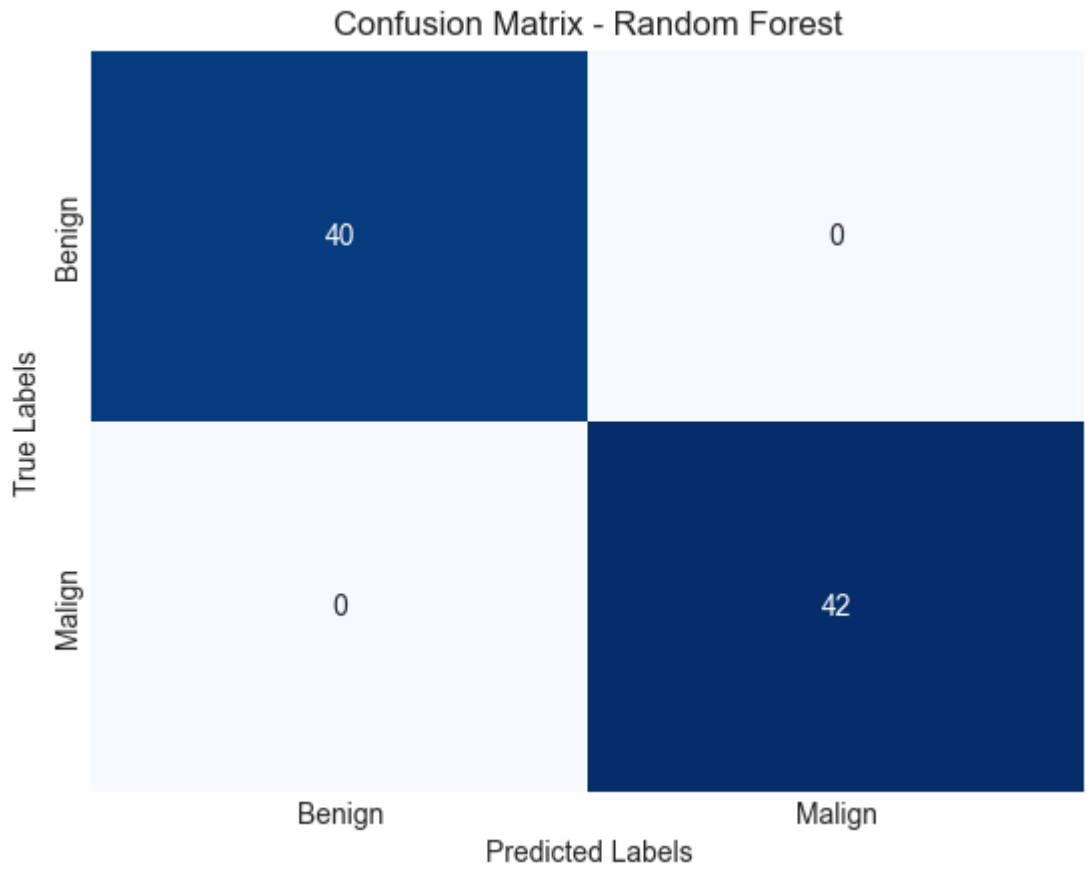

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        39

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número do Fold: 3
```



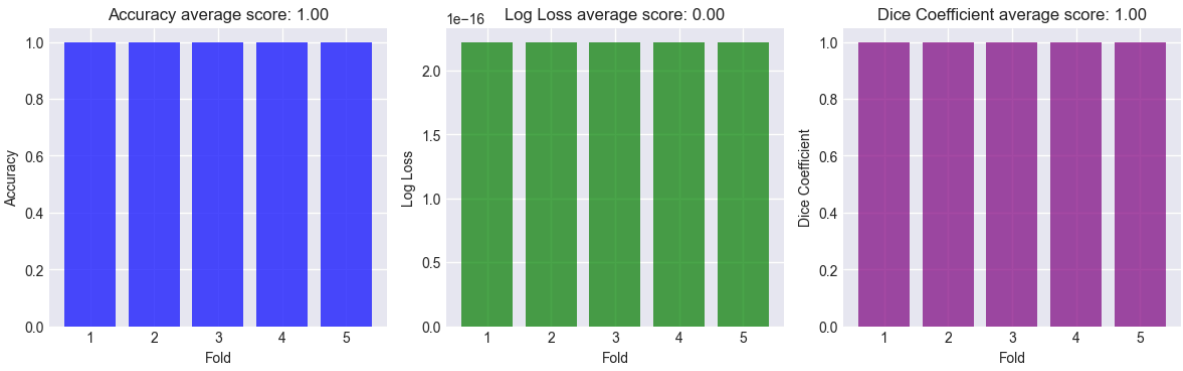

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        37
           1       1.00      1.00      1.00        45

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número do Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        39
           1       1.00      1.00      1.00        43

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número do Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       1.00      1.00      1.00        42

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82
```



Resultados Random Forest

[Voltar a Random Forest]

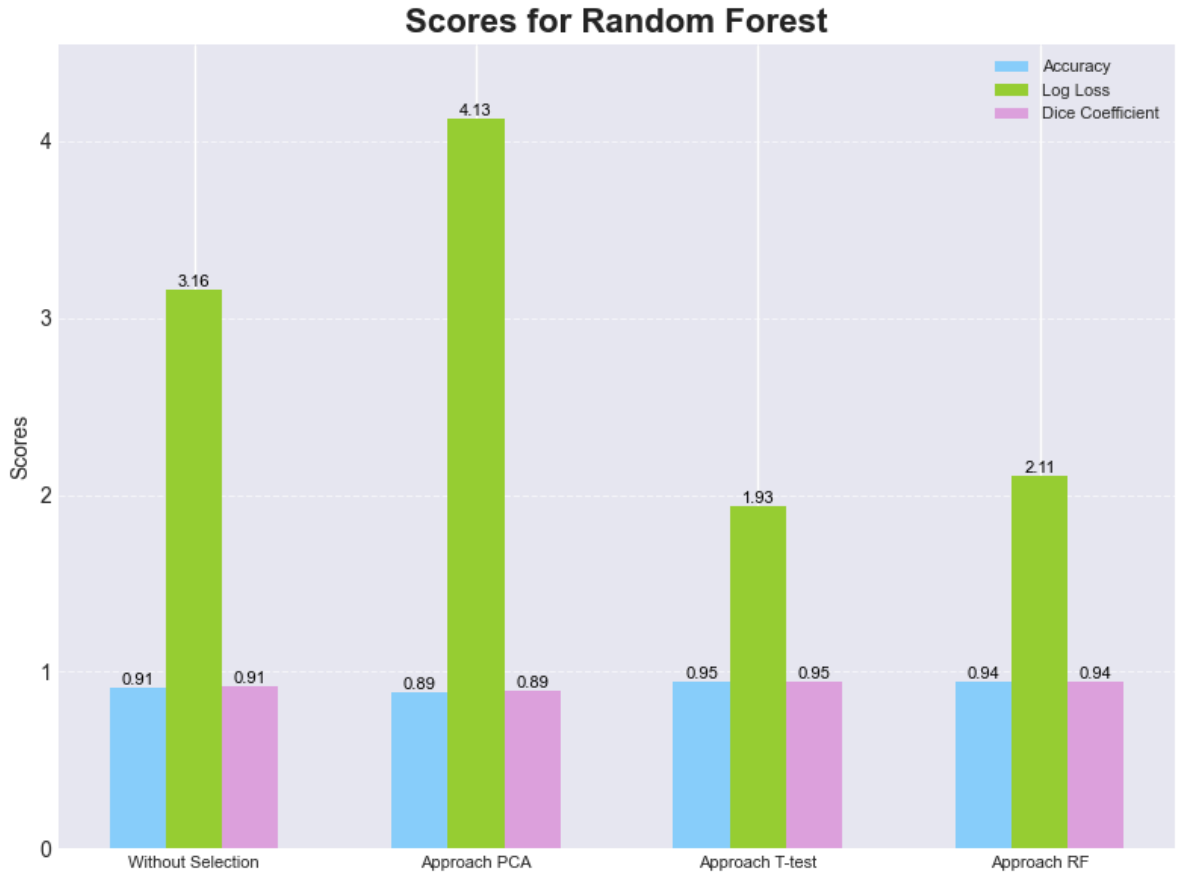
-> 2D

In [76]:
```
print(results_rf_2d)
```

```
[[0.9121951219512194, 3.164808590263945, 0.9127866022021893], [0.8853658536585366,
4.13183343728904, 0.8890888664025738], [0.9463414634146341, 1.9340496940501886, 0.
9458706597285762], [0.9414634146341463, 2.109872393509297, 0.9403221775286242]]
```

In [77]:
```
plot_scores(results_rf_2d, "Random Forest")
```

```
/var/folders/2t/mv0q1c7j2z97cgvtlhbfrhhm0000gn/T/ipykernel_67329/3843270632.py:9:
MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are depreca
ted since 3.6, as they no longer correspond to the styles shipped by seaborn. Howe
ver, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly
use the seaborn API instead.
  plt.style.use('seaborn-darkgrid')
```



-> 3D

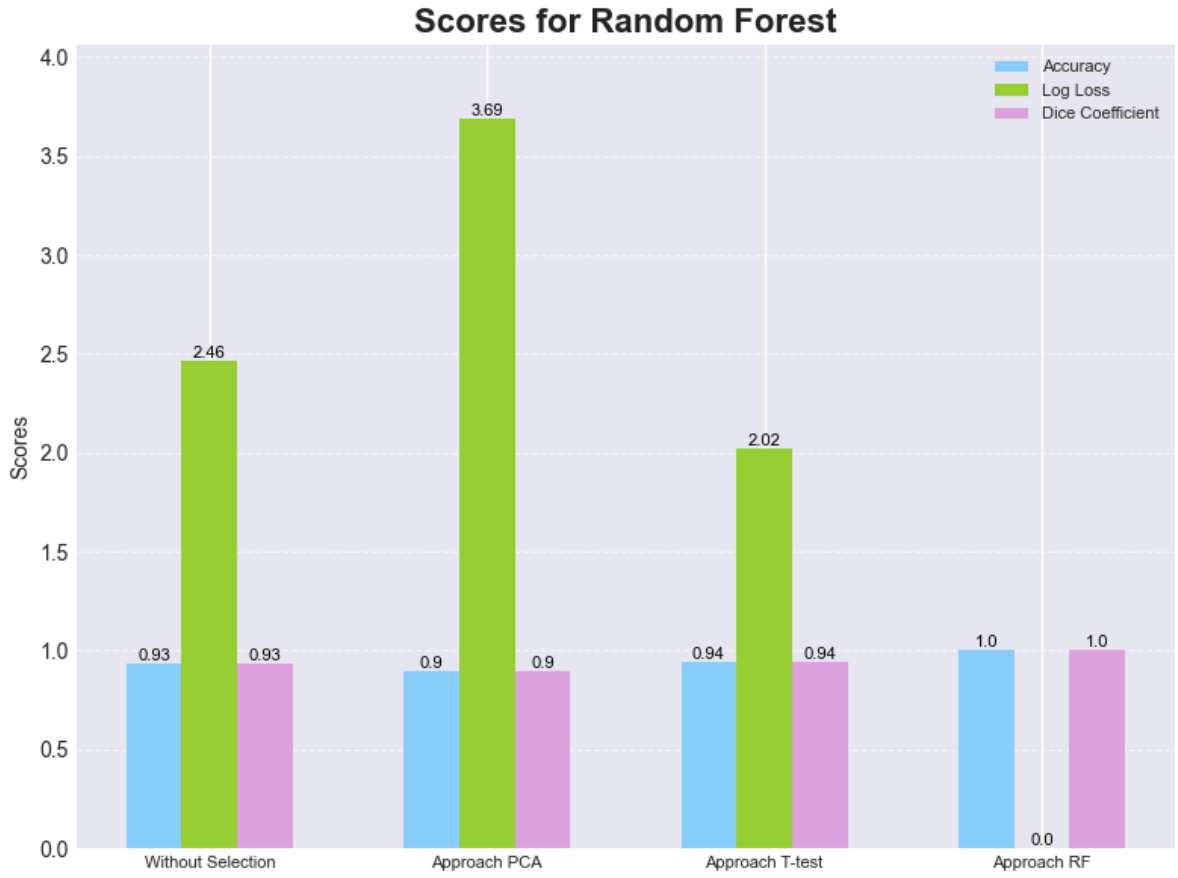
In [78]:
```
print(results_rf_3d)
```

```
[[0.9317073170731707, 2.461517792427513, 0.9320121025795863], [0.8975609756097562,
3.692276688641269, 0.8955214018715599], [0.9439024390243903, 2.0219610437797426,
0.9430966701858932], [1.0, 2.2204460492503136e-16, 1.0]]
```

In [79]:
```
plot_scores(results_rf_3d, "Random Forest")
```

```
/var/folders/2t/mv0q1c7j2z97cgvtlhbfrhhm0000gn/T/ipykernel_67329/3843270632.py:9:
MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are depreca
ted since 3.6, as they no longer correspond to the styles shipped by seaborn. Howe
ver, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly
use the seaborn API instead.
  plt.style.use('seaborn-darkgrid')
```

Loading [MathJax]/extensions/Safe.js



XGBoost

[Voltar a Training Models]

O XGBoost (Extreme Gradient Boosting) é utilizado como um modelo de classificação para prever labels de uma variável alvo y (neste caso a malignancy) a partir do input. O XGBoost é um algoritmo baseado em boosting, uma técnica de machine learning que combina vários modelos fracos (normalmente árvores de decisão) numa previsão mais robusta.

Código que iremos usar para treinar os modelos:

In [80]:
```
def xgboots_k_fold(fold, k=5, random_state=42):

    accuracies = []
    log_losses = []
    dice_coefs = []

    model = XGBClassifier(n_estimators=100, random_state=42, eval_metric='logloss')
    n_fold = 1
    for X_train, X_test, y_train, y_test in fold:
        print(f"Número de Fold: {n_fold}")
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)
        acc, logloss, dice = metrics_testing(y_test, predictions, "XGBoost")
        accuracies.append(acc)
        log_losses.append(logloss)
        dice_coefs.append(dice)
        n_fold+=1
```

```
    return accuracies, log_losses, dice_coefs
```

In [81]:
```
results_xgb_2d = []
results_xgb_3d = []
th_xgb_2d = []
th_xgb_3d =[]
```

XGBoost sem seleção de features

[Voltar a XGBoost]

-> 2D

In [82]:
```
data_2d
```

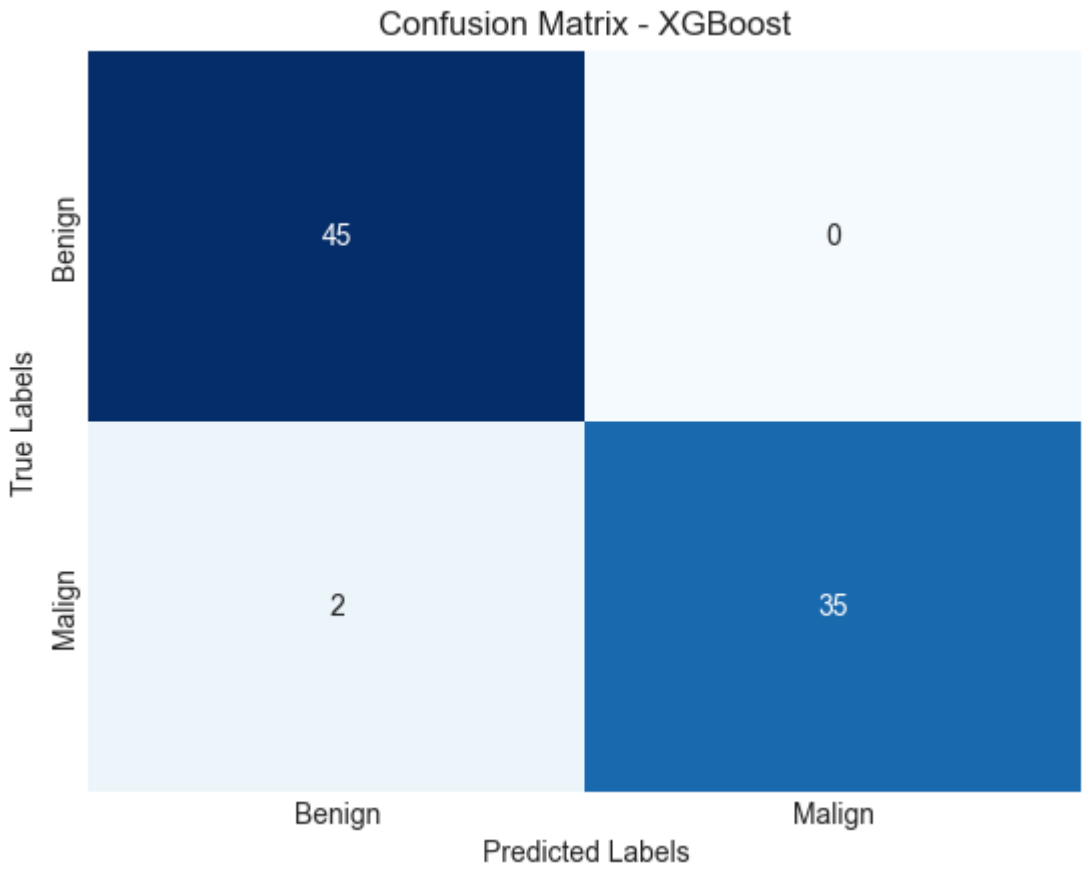
Out[82]:

	Spiculation	Lobulation	Sphericity	Margin	Subtlety	Texture	Calcification	Malignancy	dia ori
0	5	3	3	4	5	5	6	1	
1	2	2	4	3	5	4	6	1	
2	1	1	2	5	2	5	3	0	
3	5	1	4	3	5	5	6	1	
4	1	1	5	5	3	5	3	0	
...	...	...	...	...	...	...	...	...	
405	1	1	3	4	3	5	6	0	
406	2	2	4	4	5	5	6	1	
407	1	3	3	4	5	5	6	1	
408	1	1	3	3	5	5	6	1	
409	1	1	4	2	4	5	6	0	

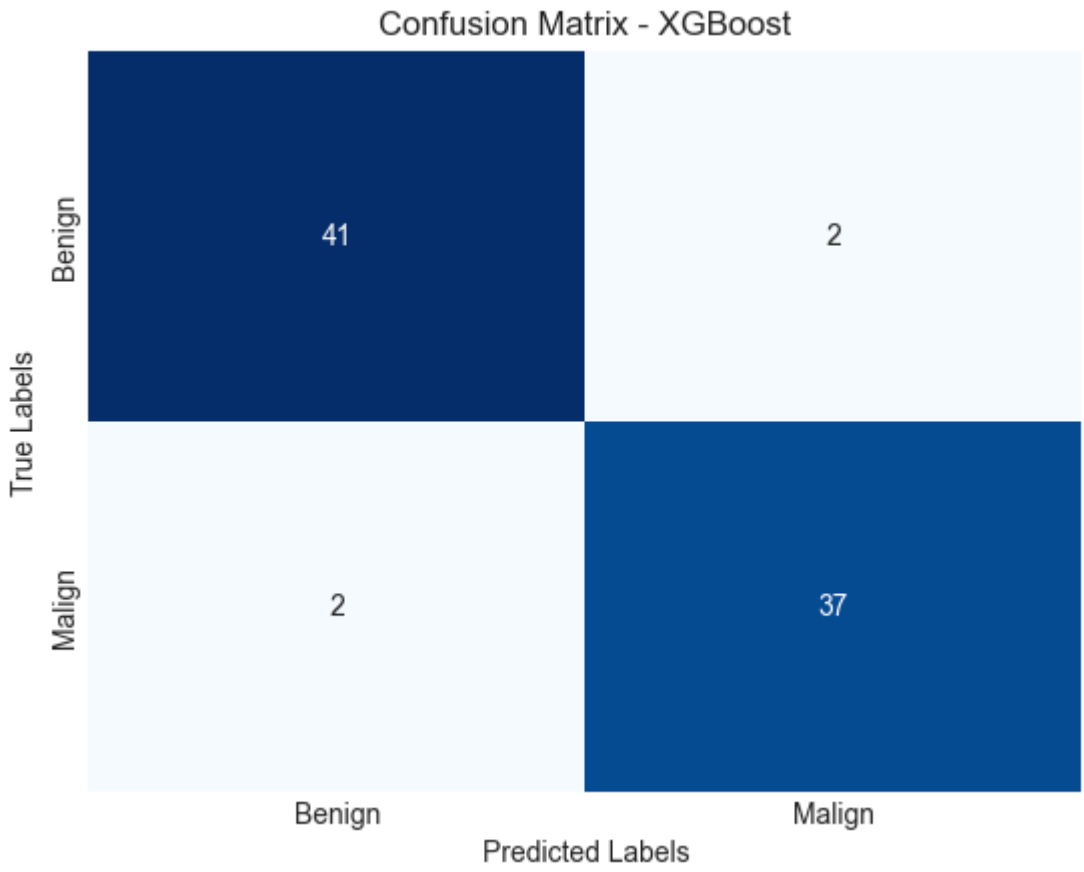
410 rows × 109 columns

In [83]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = xgboots_k_fold(fold_2d)
th_xgb_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
print(average_metrics_2d)
results_xgb_2d.append(average_metrics_2d)
```

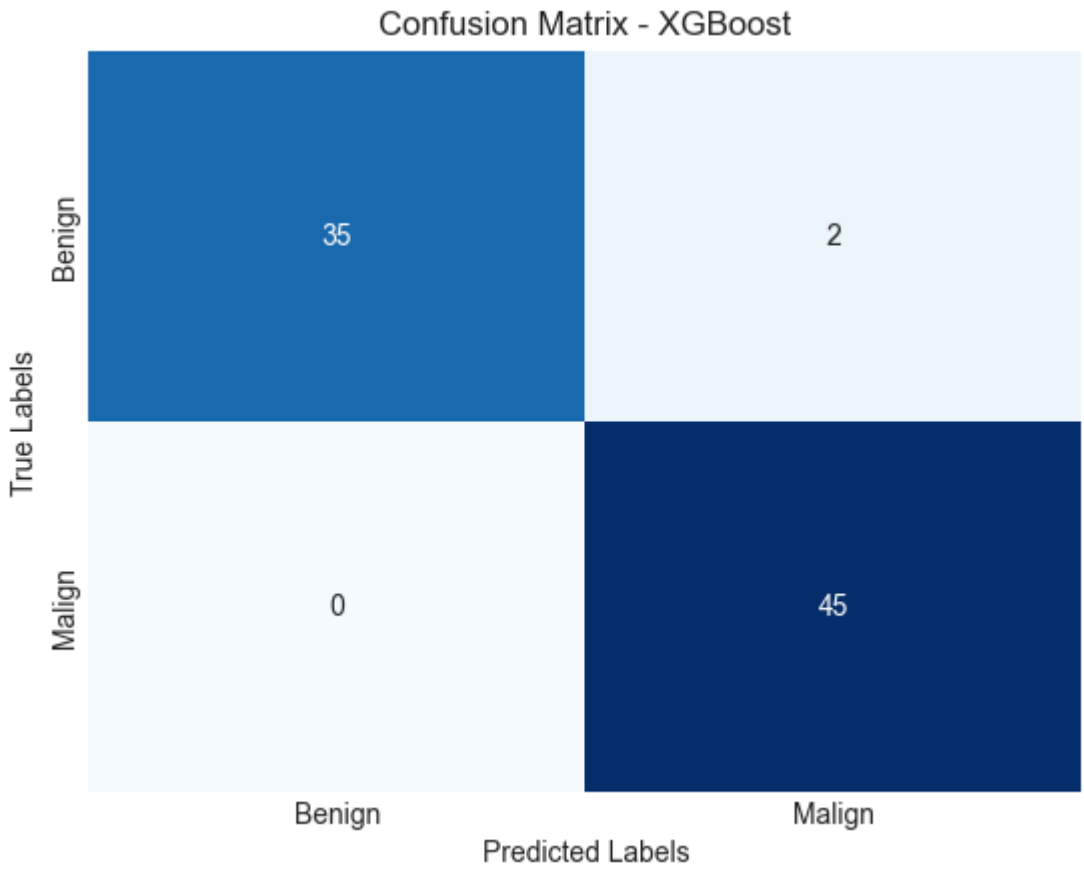
```
Número de Fold: 1
```



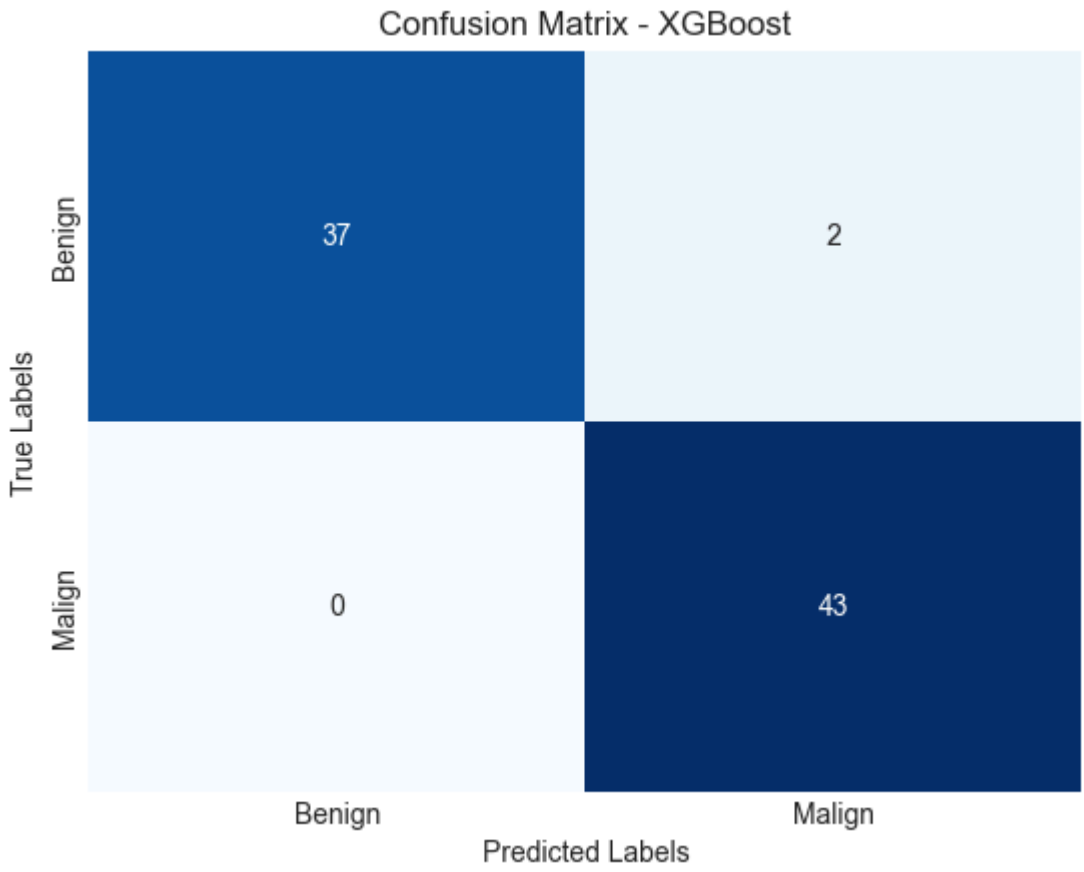

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        45
           1       1.00      0.95      0.97        37

    accuracy                           0.98        82
   macro avg       0.98      0.97      0.98        82
weighted avg       0.98      0.98      0.98        82

Número de Fold: 2
```



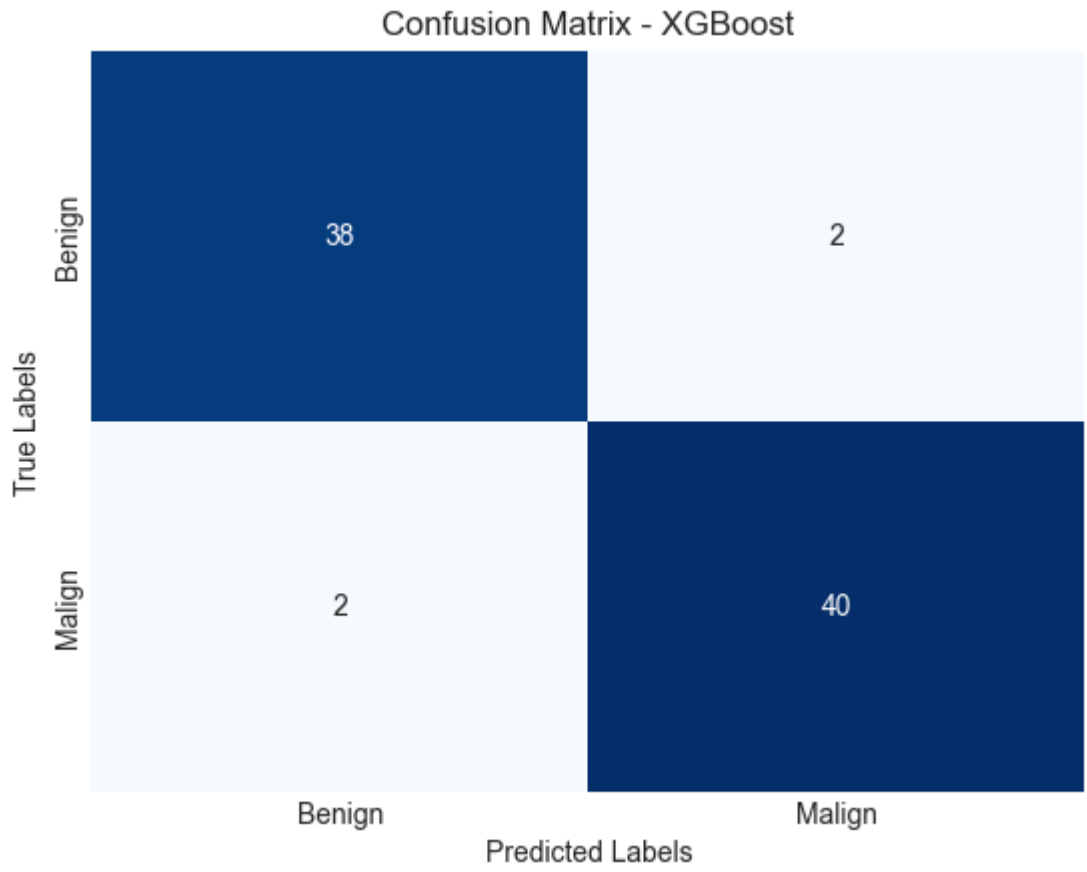

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        43
           1       0.95      0.95      0.95        39

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número de Fold: 3
```



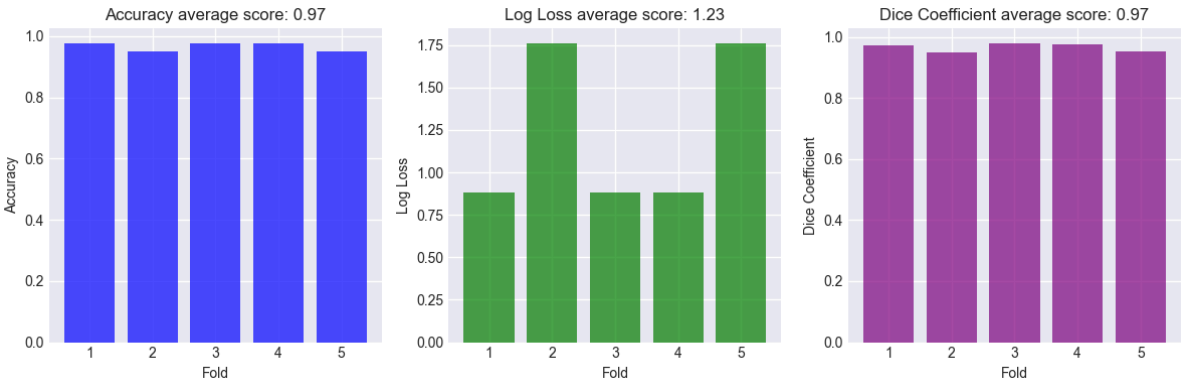

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        37
           1       0.96      1.00      0.98        45

    accuracy                           0.98        82
   macro avg       0.98      0.97      0.98        82
weighted avg       0.98      0.98      0.98        82

Número de Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        39
           1       0.96      1.00      0.98        43

    accuracy                           0.98        82
   macro avg       0.98      0.97      0.98        82
weighted avg       0.98      0.98      0.98        82

Número de Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        40
           1       0.95      0.95      0.95        42

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82
```




```
[0.9658536585365853, 1.2307588962137566, 0.9657709440318136]
```

-> 3D

In [84]:
```
data_3d
```

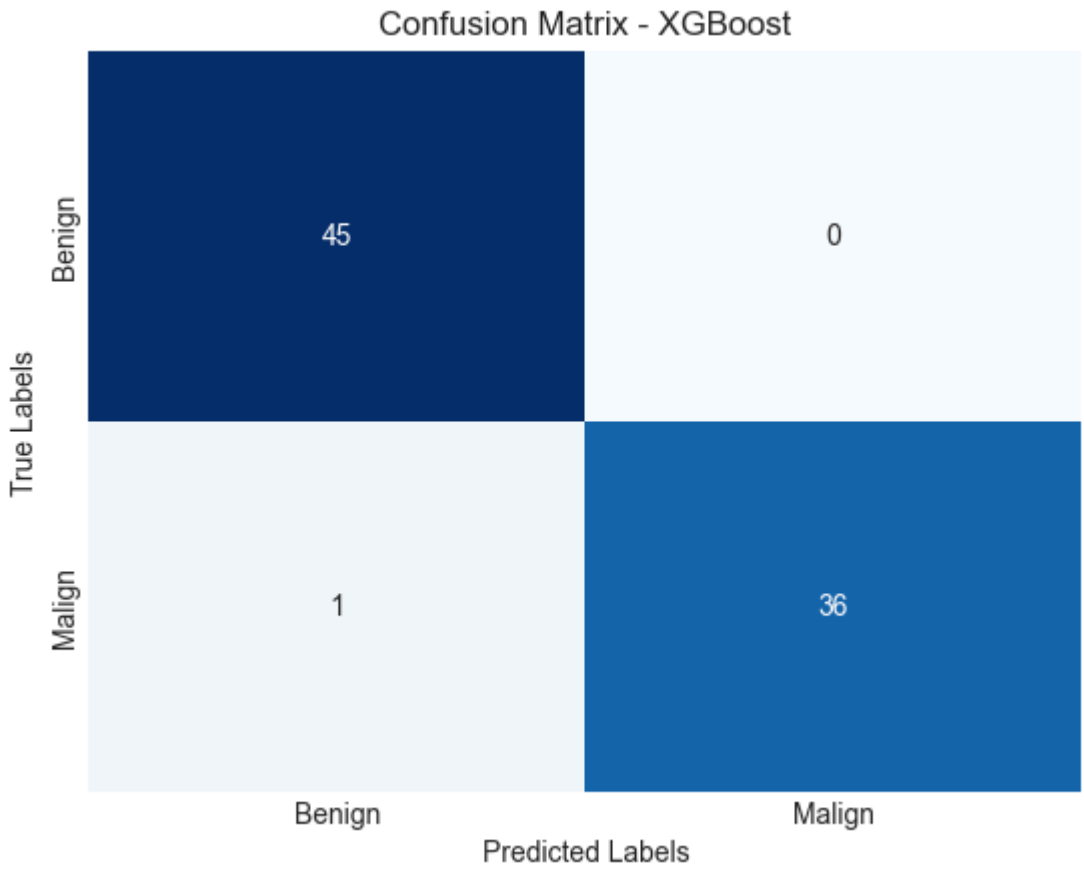
Out[84]:

	Spiculation	Lobulation	Sphericity	Margin	Subtlety	Texture	Calcification	Malignancy	dia ori
0	5	3	3	4	5	5	6	1	
1	2	2	4	3	5	4	6	1	
2	1	1	2	5	2	5	3	0	
3	5	1	4	3	5	5	6	1	
4	1	1	5	5	3	5	3	0	
...	...	...	...	...	...	...	...	...	
405	1	1	3	4	3	5	6	0	
406	2	2	4	4	5	5	6	1	
407	1	3	3	4	5	5	6	1	
408	1	1	3	3	5	5	6	1	
409	1	1	4	2	4	5	6	0	

410 rows × 105 columns

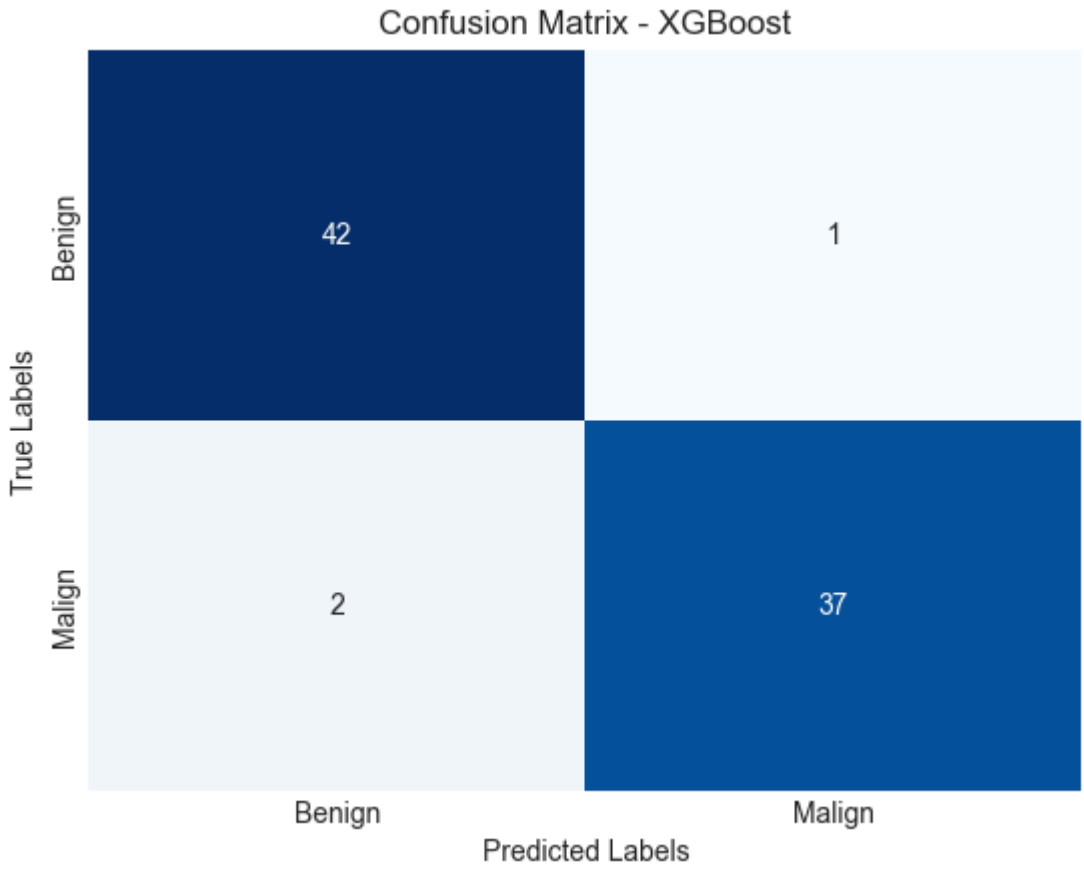
In [85]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = xgboots_k_fold(fold_3d)
th_xgb_3d.append([accuracies_3d,log_losses_3d,dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
print(average_metrics_3d)
results_xgb_3d.append(average_metrics_3d)
```

```
Número de Fold: 1
```

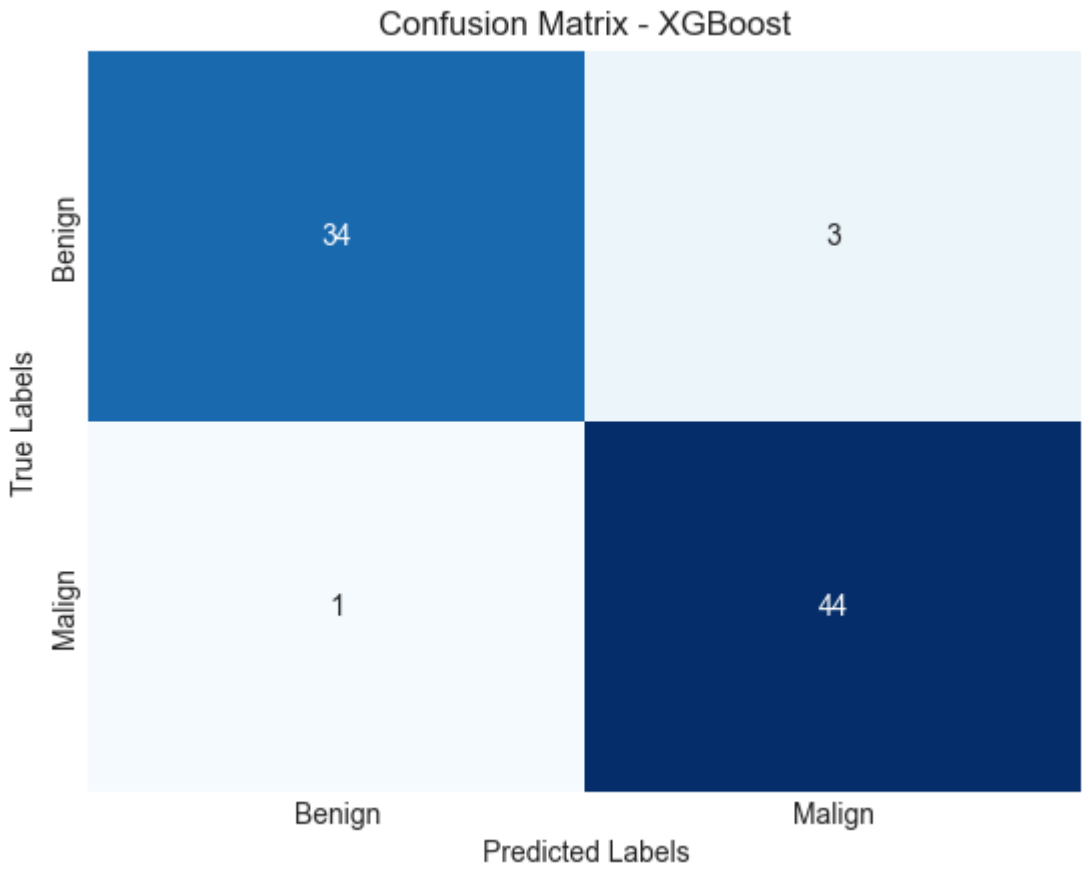



```
<Figure size 500x500 with 0 Axes>
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       1.00      0.97      0.99        37

    accuracy                           0.99        82
   macro avg       0.99      0.99      0.99        82
weighted avg       0.99      0.99      0.99        82

Número de Fold: 2
```

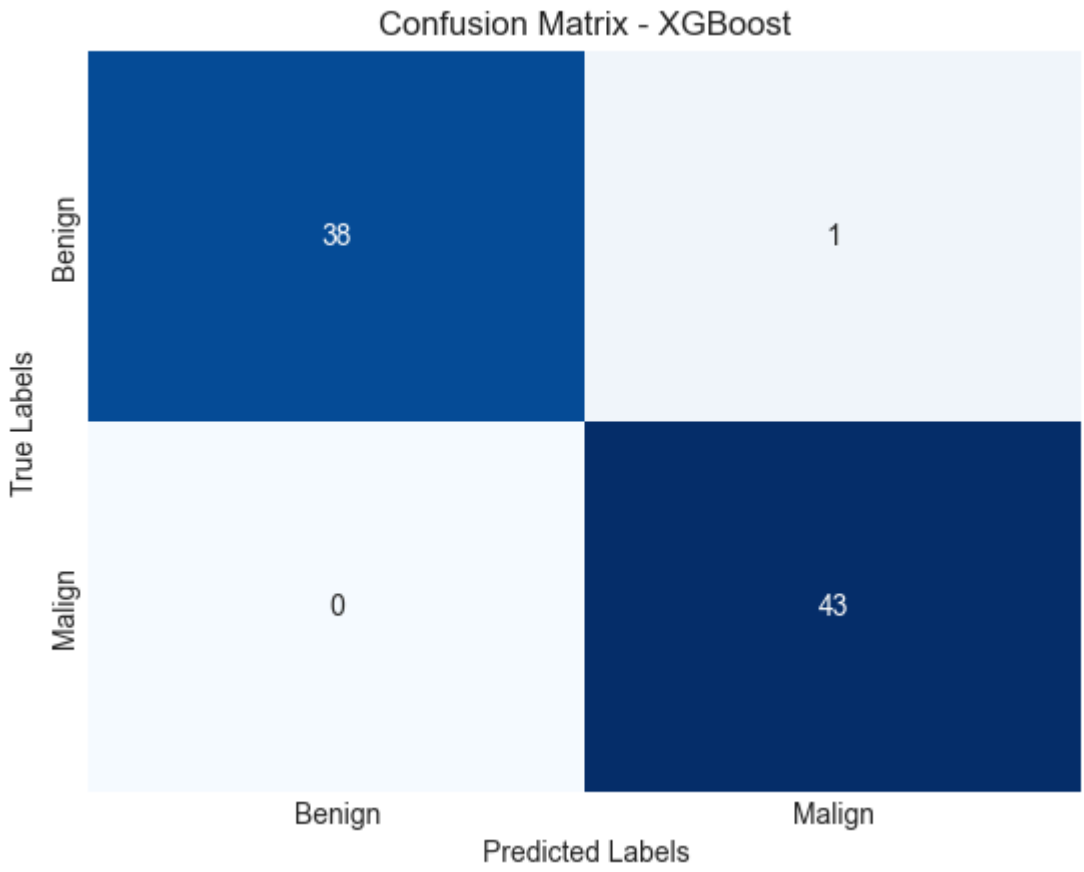



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.97        43
           1       0.97      0.95      0.96        39

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Número de Fold: 3
```

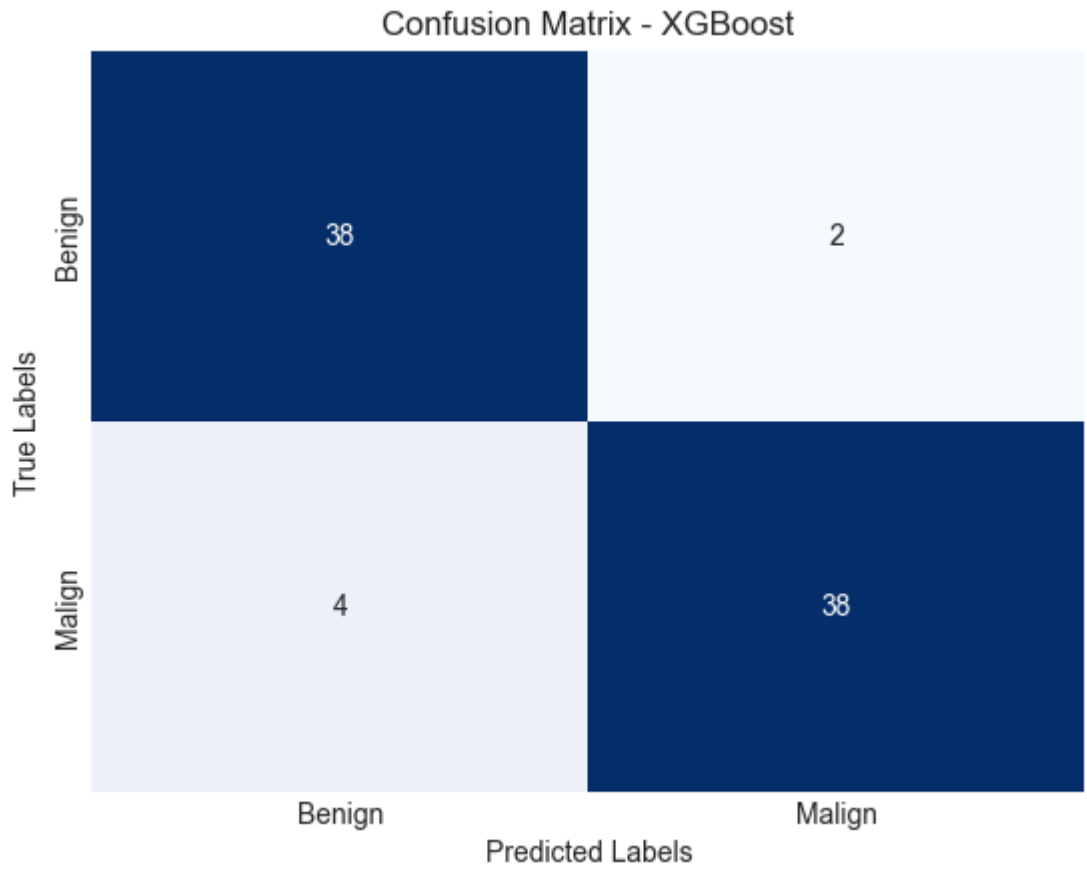



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.92      0.94        37
           1       0.94      0.98      0.96        45

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número de Fold: 4
```

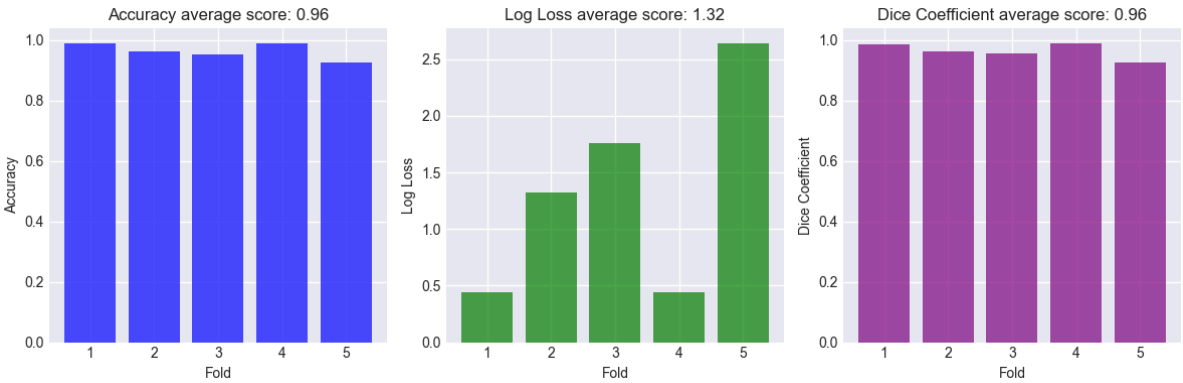
Loading [MathJax]/extensions/Safe.js




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        39
           1       0.98      1.00      0.99        43

    accuracy                           0.99        82
   macro avg       0.99      0.99      0.99        82
weighted avg       0.99      0.99      0.99        82

Número de Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.95      0.93        40
           1       0.95      0.90      0.93        42

    accuracy                           0.93        82
   macro avg       0.93      0.93      0.93        82
weighted avg       0.93      0.93      0.93        82
```




```
[0.9634146341463415, 1.3186702459433106, 0.9638394170903058]
```

XGBoost + PCA

[Voltar a XGBoost]

Iremos agora passar à implementação do modelo XGBoost, usando o dataset preparado a partir do método de Principal Component Analysis, data_pca.

-> 2D

In [86]:
```
data_pca_2d
```

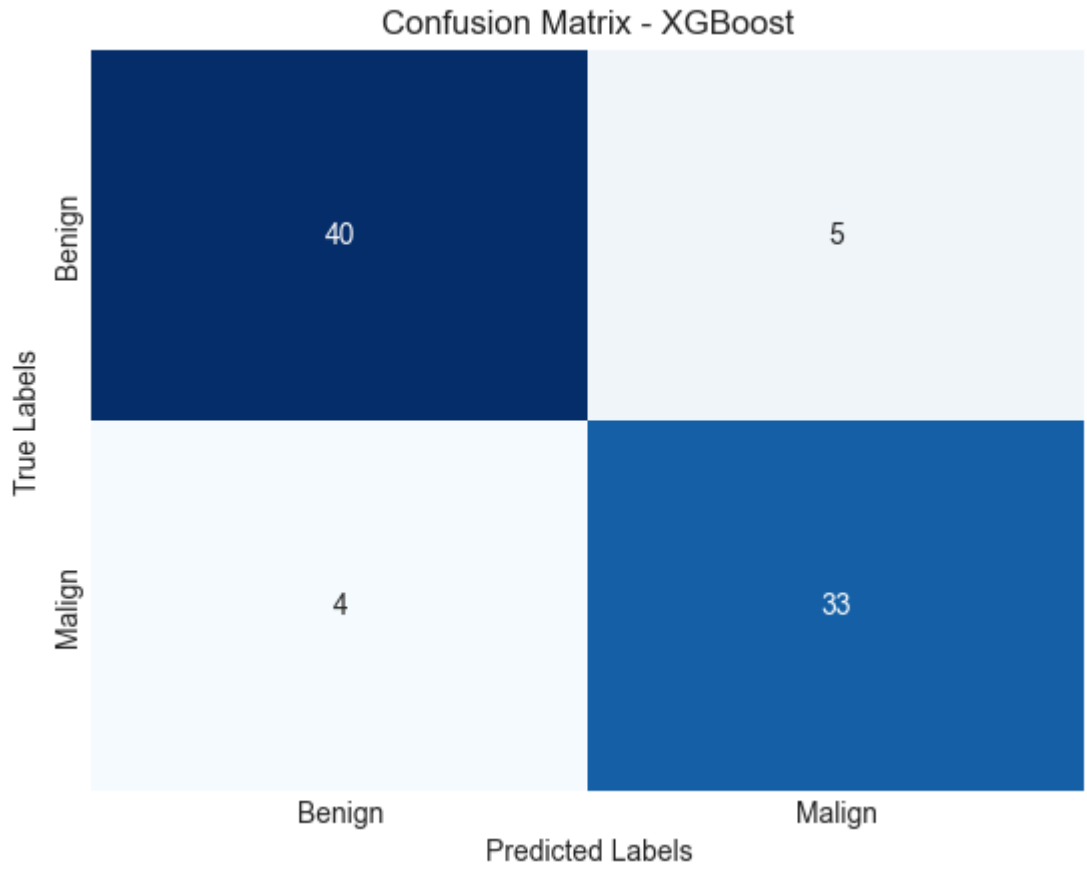
Out[86]:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	P
0	0.605527	14.312994	2.618862	-5.010918	-1.873619	0.486000	3.780943	4.509357	-3.0005
1	-2.267379	-1.819947	0.040090	0.846899	-1.514118	-0.992908	-0.811545	0.548039	-0.1913
2	5.058170	-2.480426	-3.601613	-4.459463	-1.977084	-3.833077	1.266602	3.487037	0.1742
3	26.065083	6.727777	-6.985184	4.449976	-1.804223	1.655932	-2.786482	-3.047505	1.8274
4	-2.158470	-3.958437	-0.673922	0.104410	1.383675	-1.097692	0.958196	-0.653246	0.2457
...	...	...	...	...	...	...	...	...	
405	-2.233398	-2.998710	-0.378282	0.497056	-0.144041	-1.423559	-0.648699	0.289676	0.1224
406	-2.923210	11.711044	5.856770	4.536946	0.801008	-0.007470	1.331167	-1.196007	0.6149
407	-2.716925	5.627624	3.129216	2.034017	-1.944248	1.328403	0.538597	-0.491240	0.2763
408	-2.511517	3.152825	2.056172	1.597862	-1.588072	0.221263	0.574015	-0.519261	0.3861
409	-2.493756	-2.801108	-0.163597	-0.006073	0.120255	1.392496	-1.301845	0.218497	-0.2664

410 rows × 18 columns

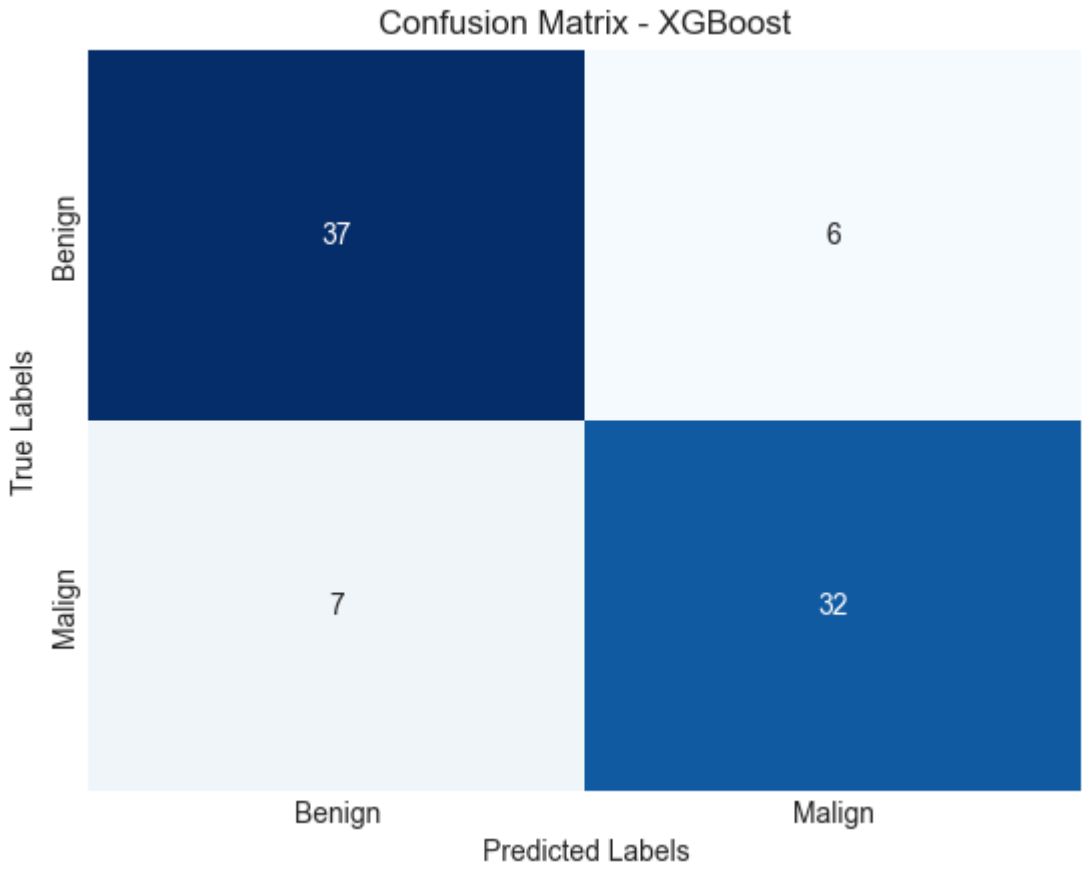
In [87]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = xgboots_k_fold(fold_2d_pca)
th_xgb_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
results_xgb_2d.append(average_metrics_2d)
```

```
Número de Fold: 1
```

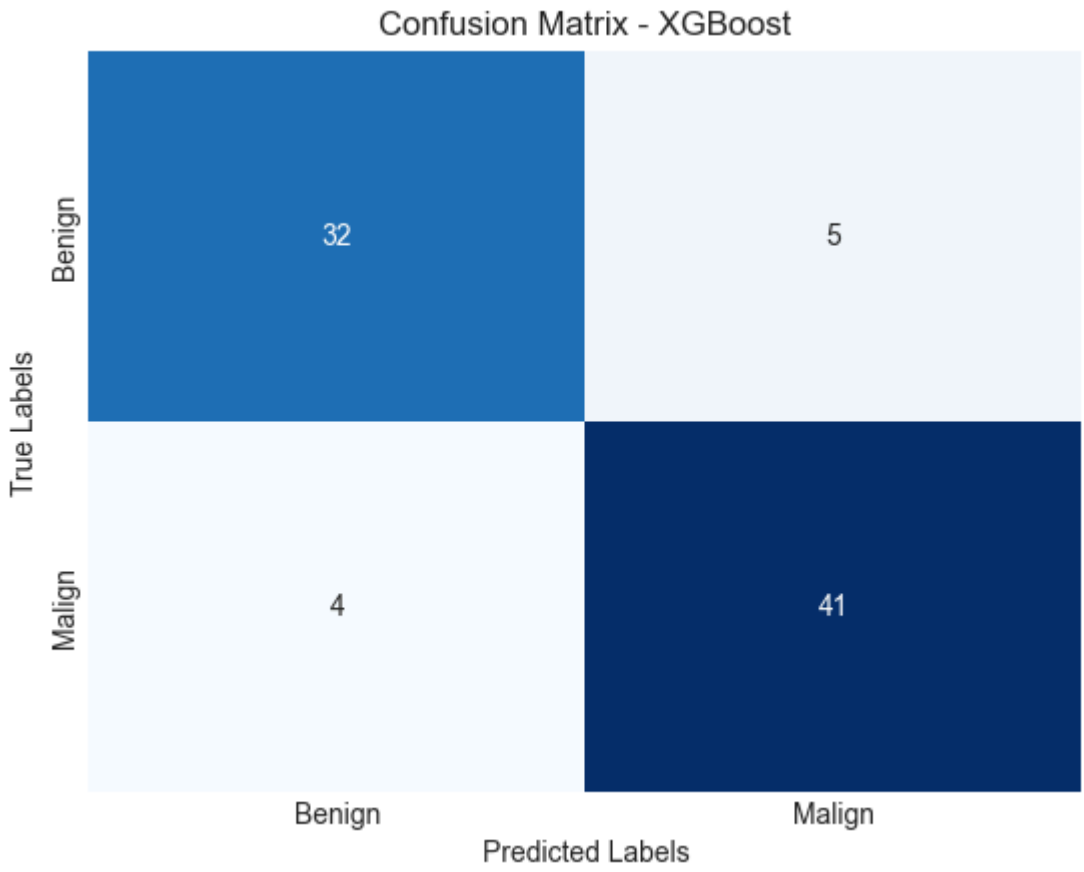



```
0x500 with 0 Axes>
```

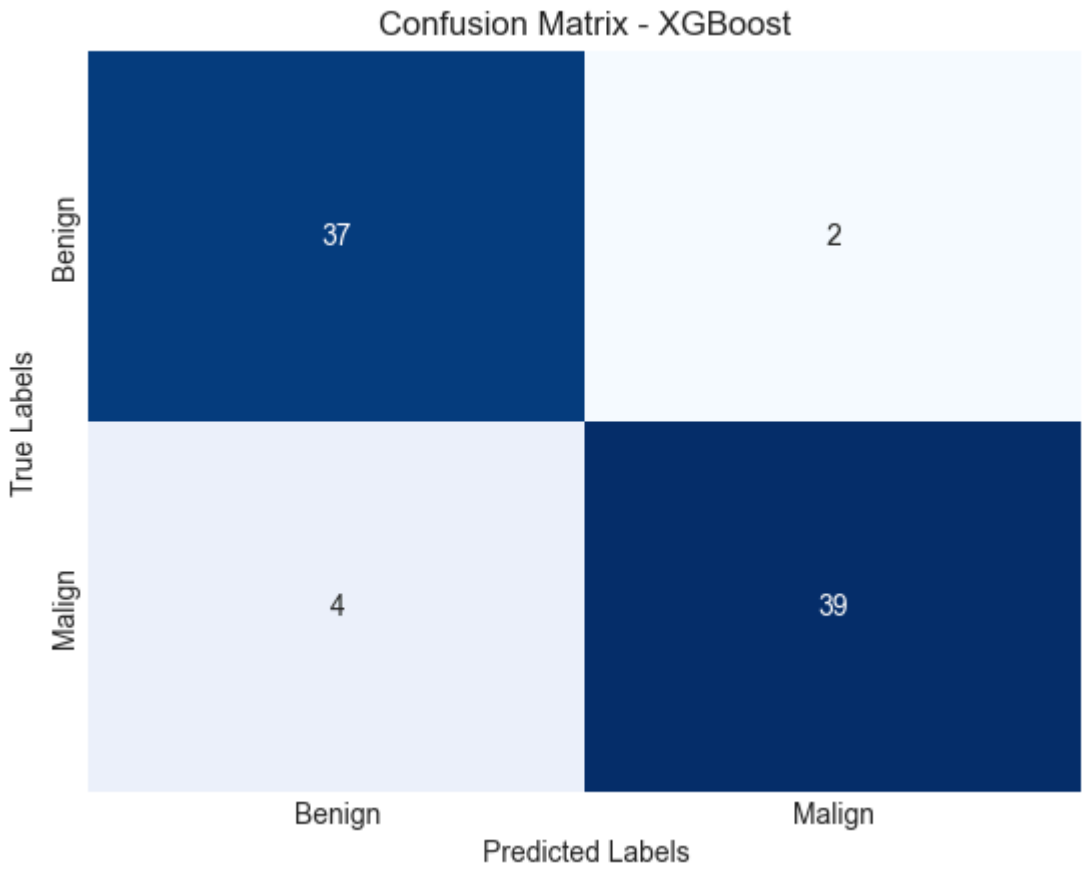
```
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.89      0.90        45
           1       0.87      0.89      0.88        37

    accuracy                           0.89        82
   macro avg       0.89      0.89      0.89        82
weighted avg       0.89      0.89      0.89        82

Número de Fold: 2
```



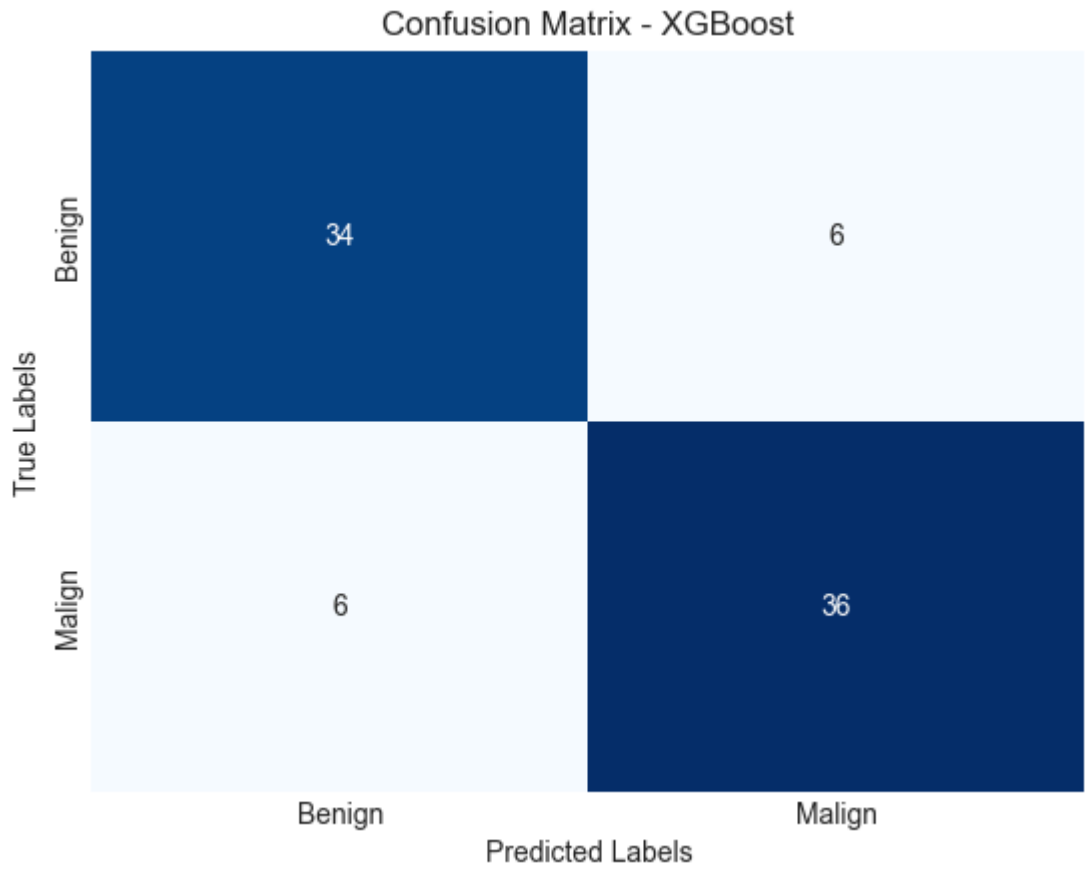

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.86      0.85        43
           1       0.84      0.82      0.83        39

    accuracy                           0.84        82
   macro avg       0.84      0.84      0.84        82
weighted avg       0.84      0.84      0.84        82

Número de Fold: 3
```



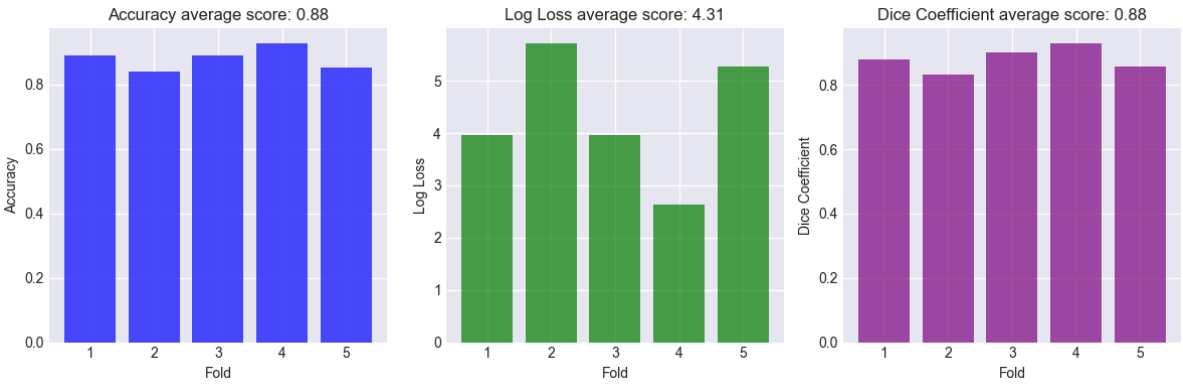

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.88        37
           1       0.89      0.91      0.90        45

    accuracy                           0.89        82
   macro avg       0.89      0.89      0.89        82
weighted avg       0.89      0.89      0.89        82

Número de Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.95      0.92        39
           1       0.95      0.91      0.93        43

    accuracy                           0.93        82
   macro avg       0.93      0.93      0.93        82
weighted avg       0.93      0.93      0.93        82

Número de Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.85      0.85        40
           1       0.86      0.86      0.86        42

    accuracy                           0.85        82
   macro avg       0.85      0.85      0.85        82
weighted avg       0.85      0.85      0.85        82
```



-> 3D

In [88]:
```
data_pca_3d
```

Loading [MathJax]/extensions/Safe.js

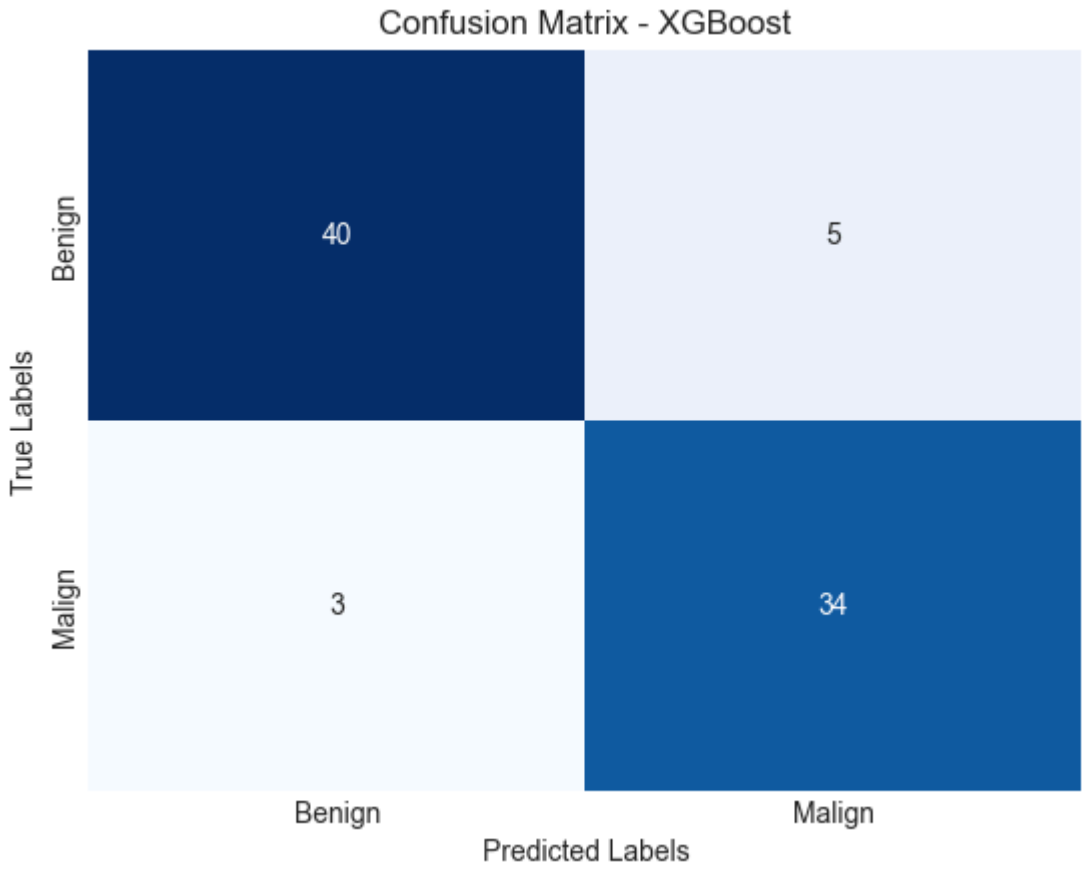
Out[88]:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	P
0	-0.782122	-4.117717	1.807559	3.238631	-2.126003	0.470964	0.041700	-1.212518	-0.2280
1	-2.459356	2.218242	-1.818623	2.089689	0.356737	1.117322	-0.424158	-0.750689	-0.4551
2	13.960910	0.629641	-1.754254	-1.768815	-1.563777	-2.868636	-2.049989	-1.101771	-1.3600
3	18.196420	-6.671389	-6.882279	1.530733	2.834788	2.809031	-2.197428	2.412911	1.9639
4	-2.639728	3.325768	-2.169896	0.994315	-0.182417	-0.915270	1.220950	1.471481	0.1514
...	...	...	...	...	...	...	...	...	
405	-2.682271	3.198533	-2.295293	1.549300	0.057934	-0.301314	0.212454	-0.055443	0.9605
406	-2.269498	-5.455861	4.720836	7.225682	2.834643	-3.934610	3.135748	-0.186134	0.6010
407	-2.505729	-0.167273	0.442583	2.612708	2.009912	0.850195	-0.439097	0.373260	-1.7192
408	-2.686433	0.246159	0.531057	1.261710	1.665685	0.842926	-0.394356	0.287286	-1.0636
409	-2.196771	3.355261	-3.609834	4.514218	-0.464621	0.370936	0.322757	-0.490927	1.1526

410 rows × 18 columns

In [89]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = xgboots_k_fold(fold_3d_pca)
th_xgb_3d.append([accuracies_3d,log_losses_3d, dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
results_xgb_3d.append(average_metrics_3d)
```

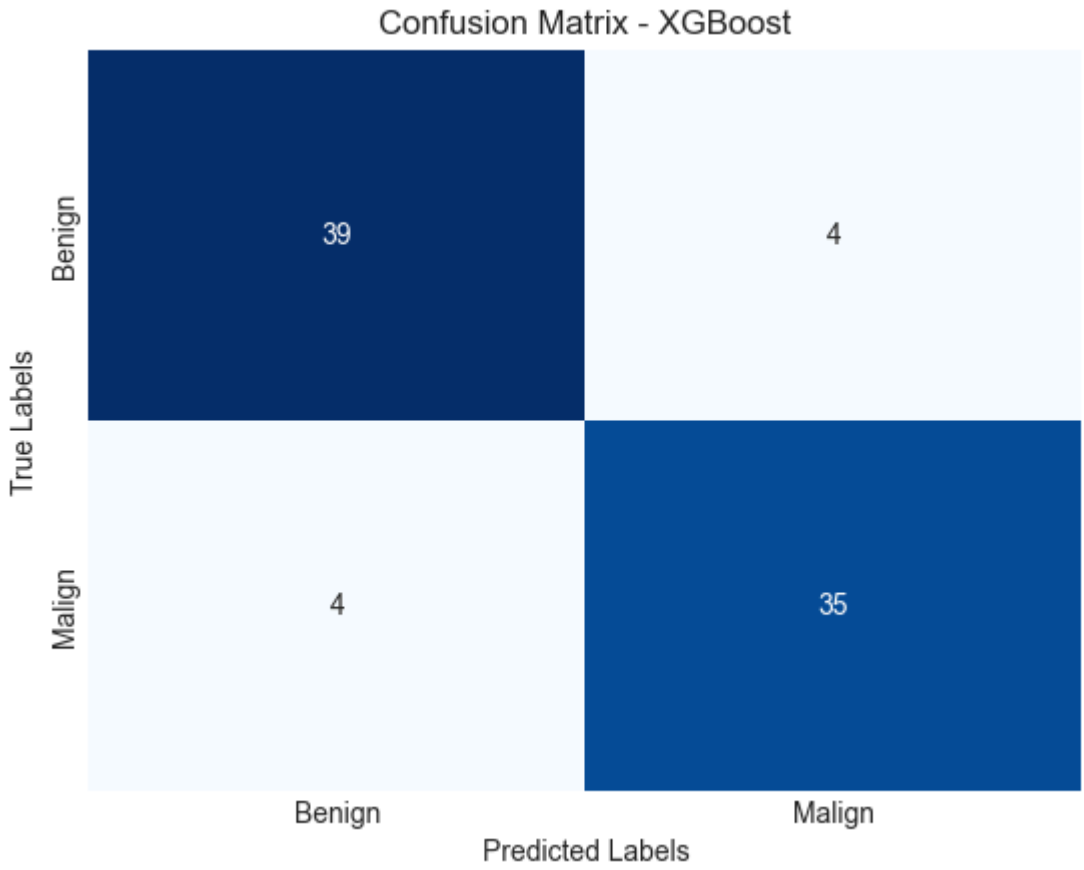
```
Número de Fold: 1
```



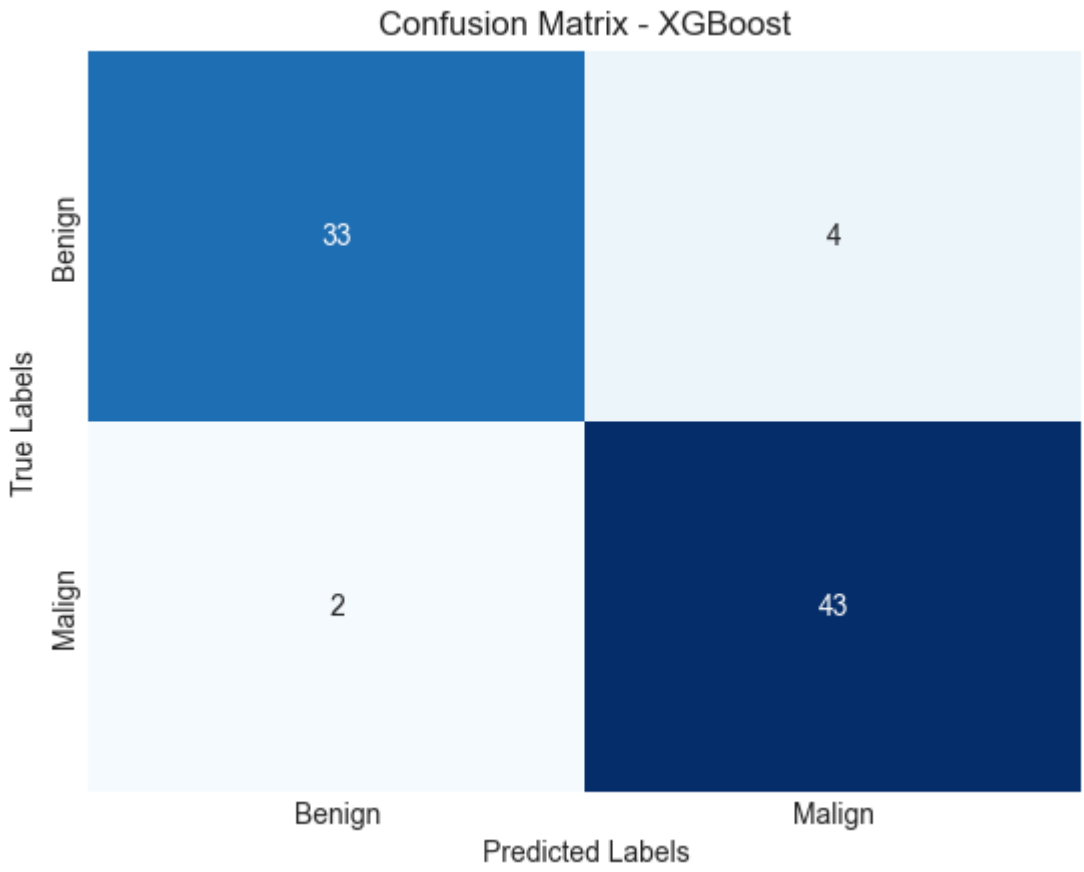

```
<Figure size 500x500 with 0 Axes>
```

Loading [MathJax]/extensions/Safe.js

```
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.89      0.91        45
           1       0.87      0.92      0.89        37

    accuracy                           0.90        82
   macro avg       0.90      0.90      0.90        82
weighted avg       0.90      0.90      0.90        82

Número de Fold: 2
```

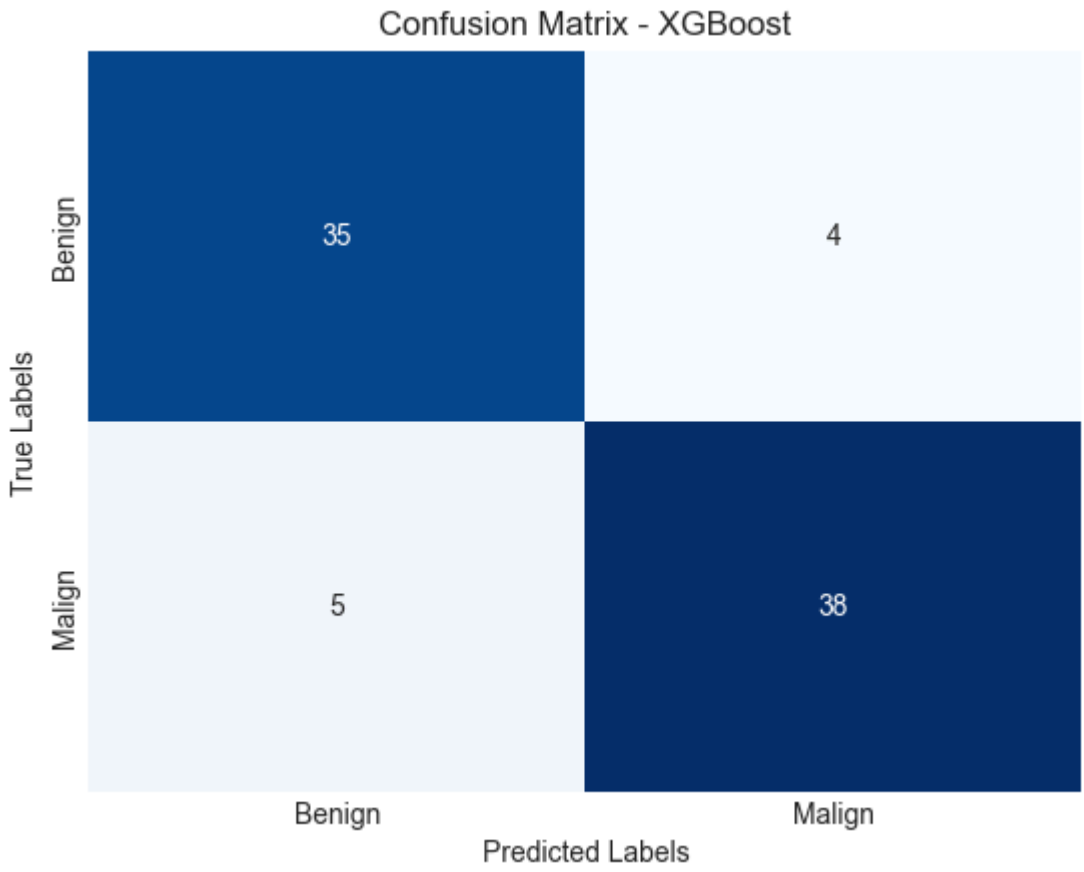



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.91      0.91        43
           1       0.90      0.90      0.90        39

    accuracy                           0.90        82
   macro avg       0.90      0.90      0.90        82
weighted avg       0.90      0.90      0.90        82

Número de Fold: 3
```

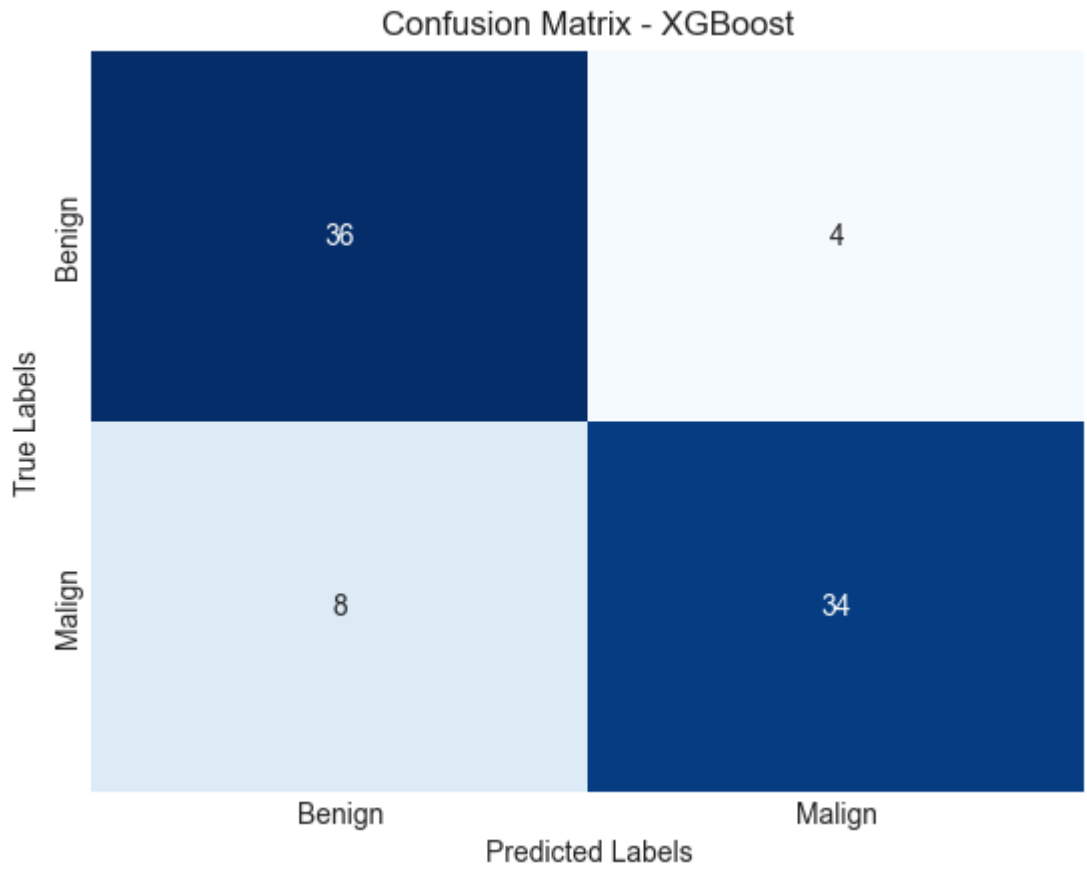



```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.89      0.92        37
           1       0.91      0.96      0.93        45

    accuracy                           0.93        82
   macro avg       0.93      0.92      0.93        82
weighted avg       0.93      0.93      0.93        82

Número de Fold: 4
```

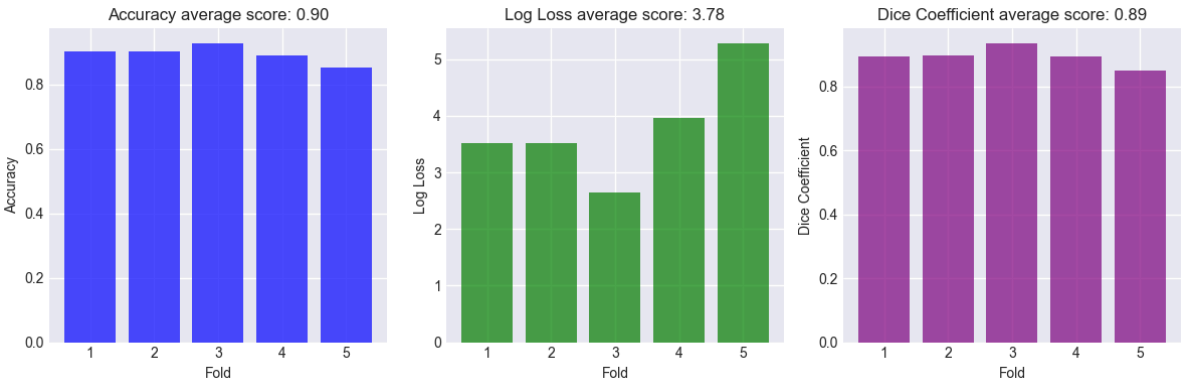
Loading [MathJax]/extensions/Safe.js




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.90      0.89        39
           1       0.90      0.88      0.89        43

    accuracy                           0.89        82
   macro avg       0.89      0.89      0.89        82
weighted avg       0.89      0.89      0.89        82

Número de Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.90      0.86        40
           1       0.89      0.81      0.85        42

    accuracy                           0.85        82
   macro avg       0.86      0.85      0.85        82
weighted avg       0.86      0.85      0.85        82
```



XGBoost + t-test

[Voltar a XGBoost]

Repetimos agora o processo para o dataset obtido pelo t-test.

In [90]:
```
data_ttest_2d
```

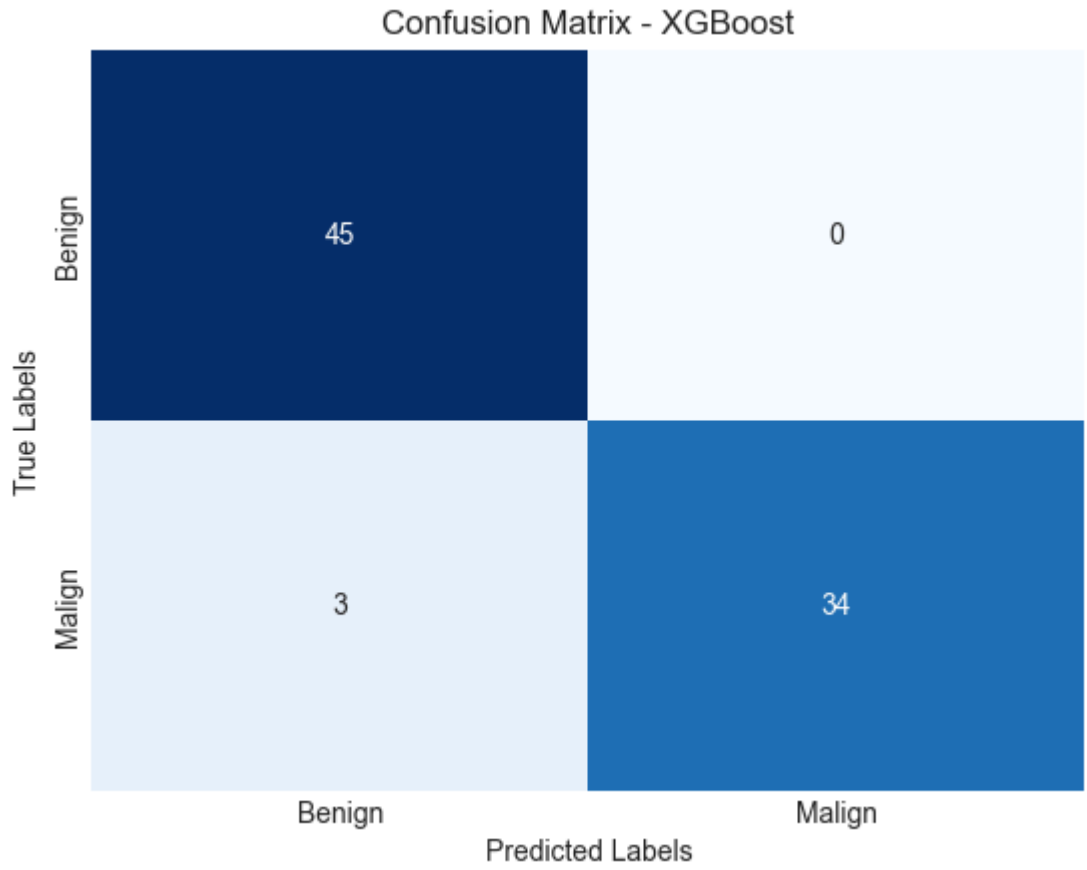
Out[90]:

	Calcification	original_shape2D_Sphericity	original_shape2D_MajorAxisLength	original_shape2D
0	6	0.116476	57.542760	
1	6	0.237705	17.090541	
2	3	0.293095	9.244896	
3	6	0.145259	40.718270	
4	3	0.288620	10.327956	
...	...	...	...	
405	6	0.245244	13.852837	
406	6	0.112044	61.575217	
407	6	0.138573	51.164214	
408	6	0.154955	41.452617	
409	6	0.276215	12.494295	

410 rows × 49 columns

In [91]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = xgboots_k_fold(fold_2d_ttest)
th_xgb_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
results_xgb_2d.append(average_metrics_2d)
```

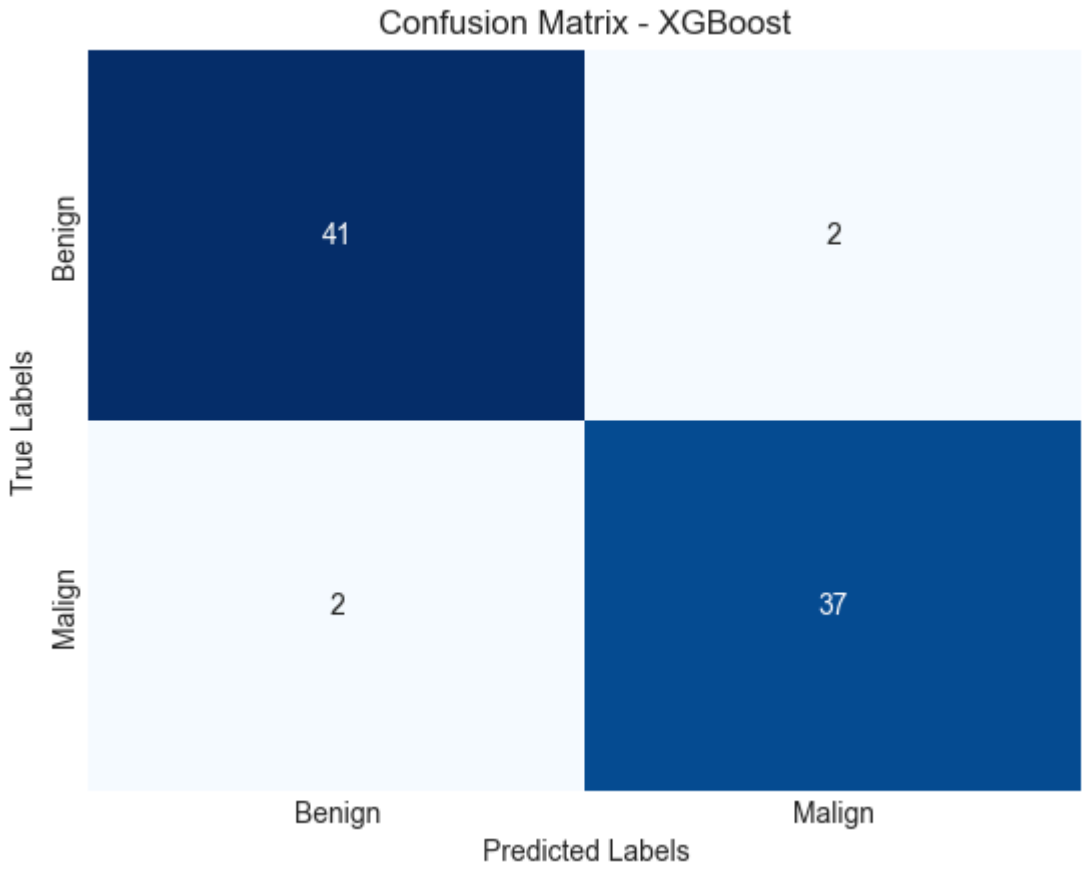
```
Número de Fold: 1
```



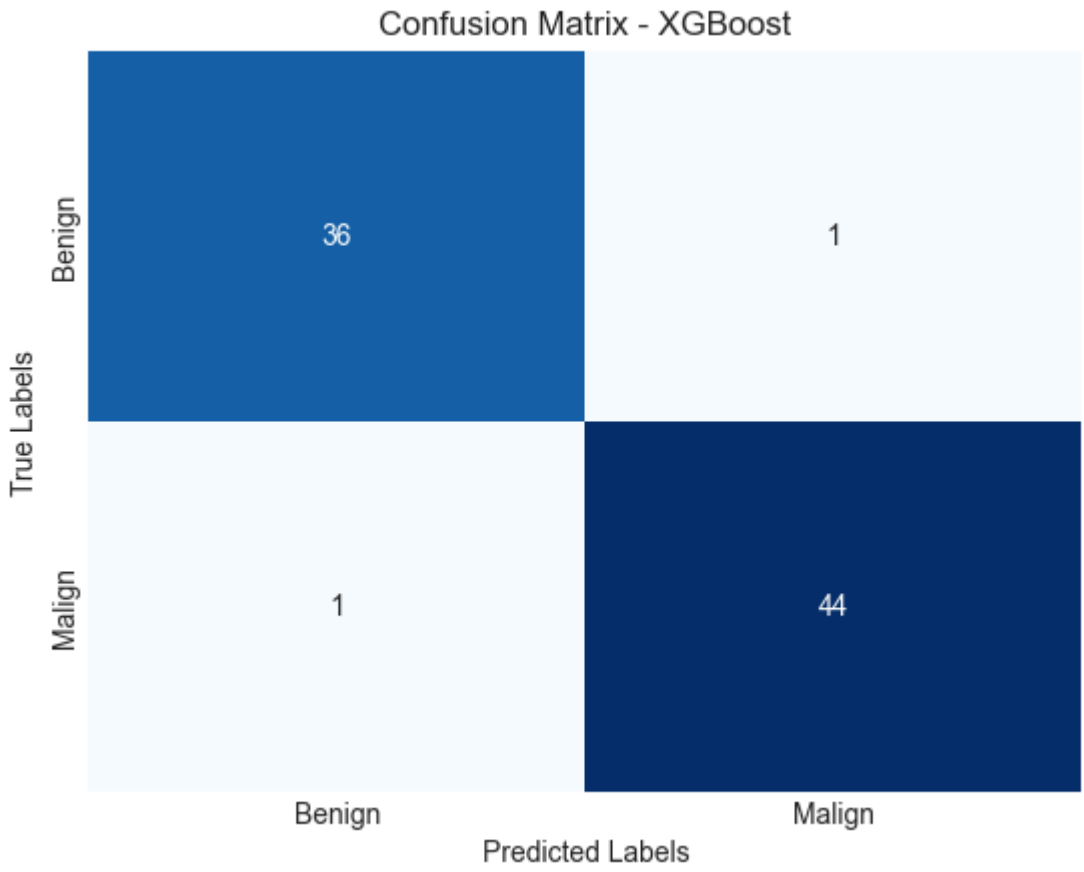

```
0x500 with 0 Axes>
```

Loading [MathJax]/extensions/Safe.js

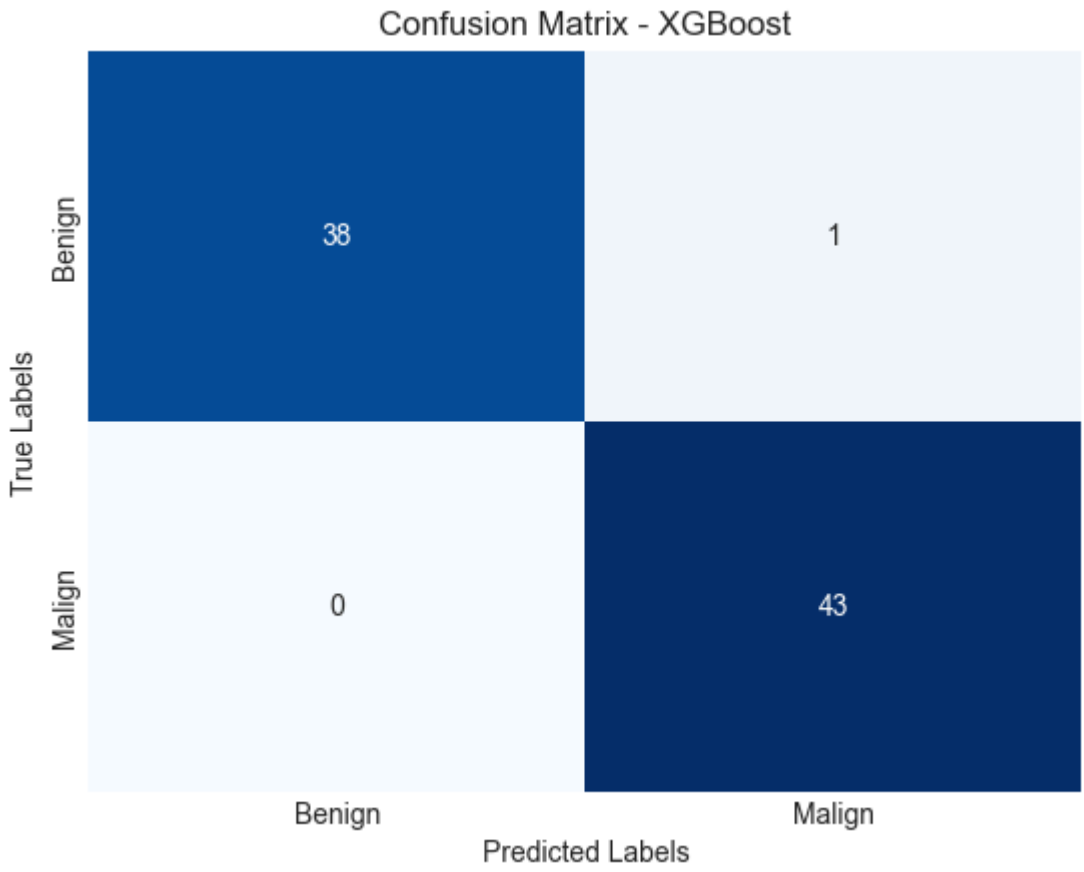
```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        45
           1       1.00      0.92      0.96        37

    accuracy                           0.96        82
   macro avg       0.97      0.96      0.96        82
weighted avg       0.97      0.96      0.96        82

Número de Fold: 2
```



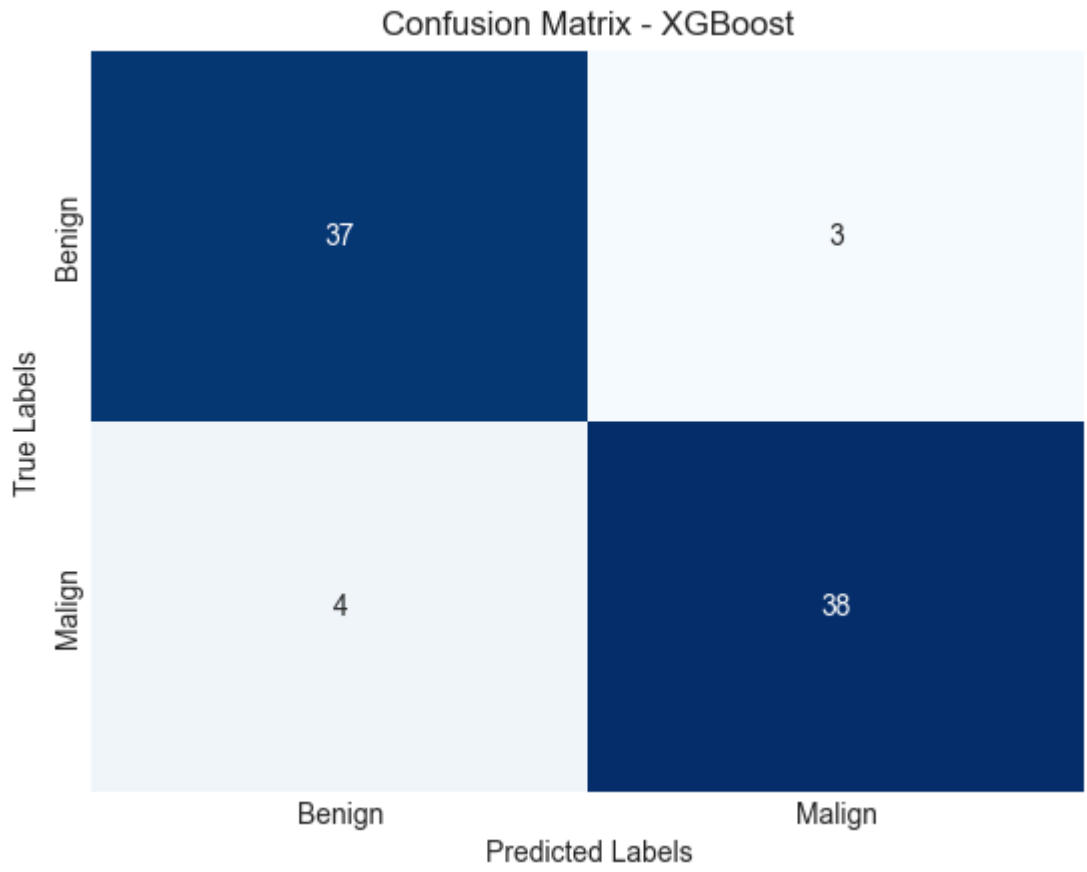

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        43
           1       0.95      0.95      0.95        39

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

Número de Fold: 3
```



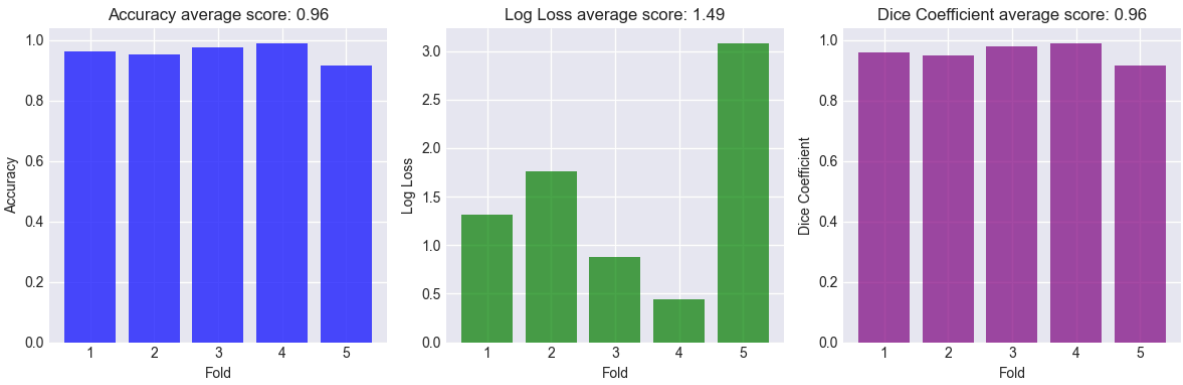

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        37
           1       0.98      0.98      0.98        45

    accuracy                           0.98        82
   macro avg       0.98      0.98      0.98        82
weighted avg       0.98      0.98      0.98        82

Número de Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        39
           1       0.98      1.00      0.99        43

    accuracy                           0.99        82
   macro avg       0.99      0.99      0.99        82
weighted avg       0.99      0.99      0.99        82

Número de Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.93      0.91        40
           1       0.93      0.90      0.92        42

    accuracy                           0.91        82
   macro avg       0.91      0.91      0.91        82
weighted avg       0.91      0.91      0.91        82
```



-> 3D

In [92]:
```
data_ttest_3d
```

Loading [MathJax]/extensions/Safe.js

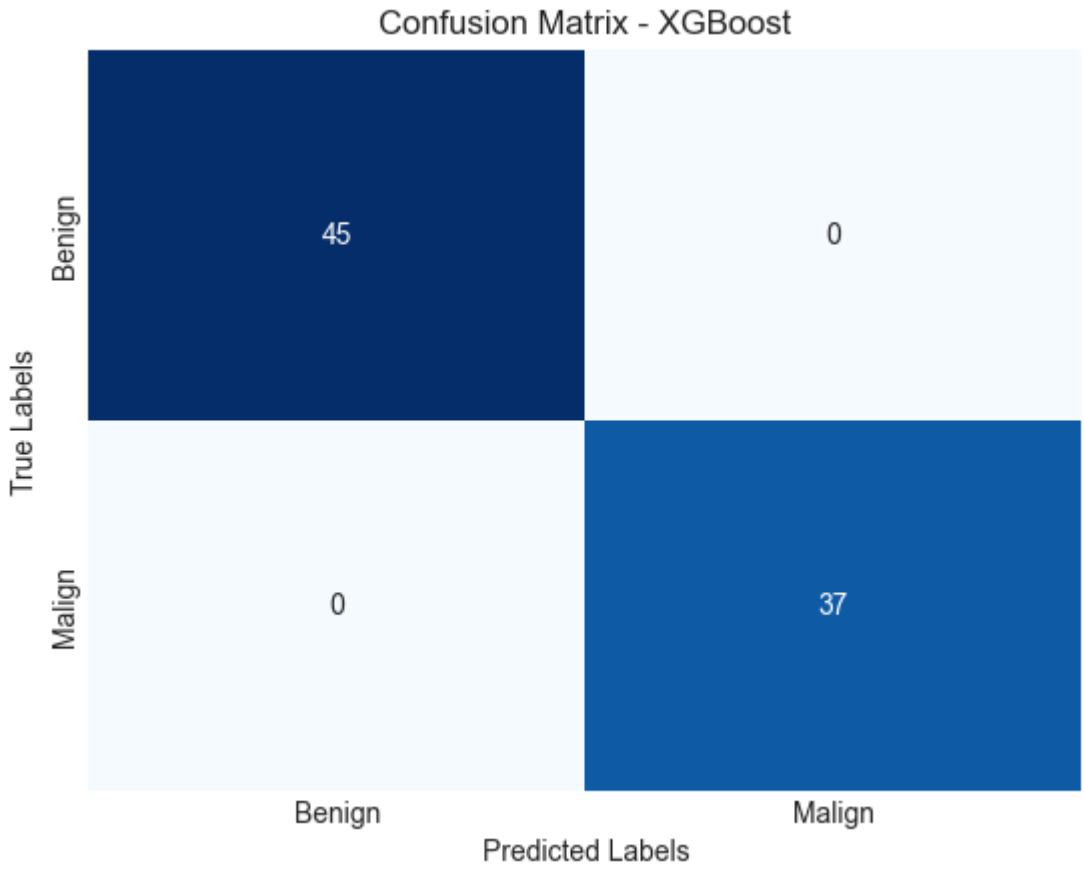
Out[92]:

	Calcification	Spiculation	Lobulation	Margin	Subtlety	original_glrlm_GrayLevelNonUniformit
0	6	5	3	4	5	757.88036
1	6	2	2	3	5	88.38461
2	3	1	1	5	2	34.51526
3	6	5	1	3	5	437.07015
4	3	1	1	5	3	33.84615
...	...	...	...	...	...	
405	6	1	1	4	3	41.76923
406	6	2	2	4	5	1239.76923
407	6	1	3	4	5	573.30769
408	6	1	1	3	5	444.76923
409	6	1	1	2	4	47.84615

410 rows × 58 columns

In [93]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = xgboots_k_fold(fold_3d_rf)
th_xgb_3d.append([accuracies_3d,log_losses_3d, dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
results_xgb_3d.append(average_metrics_3d)
```

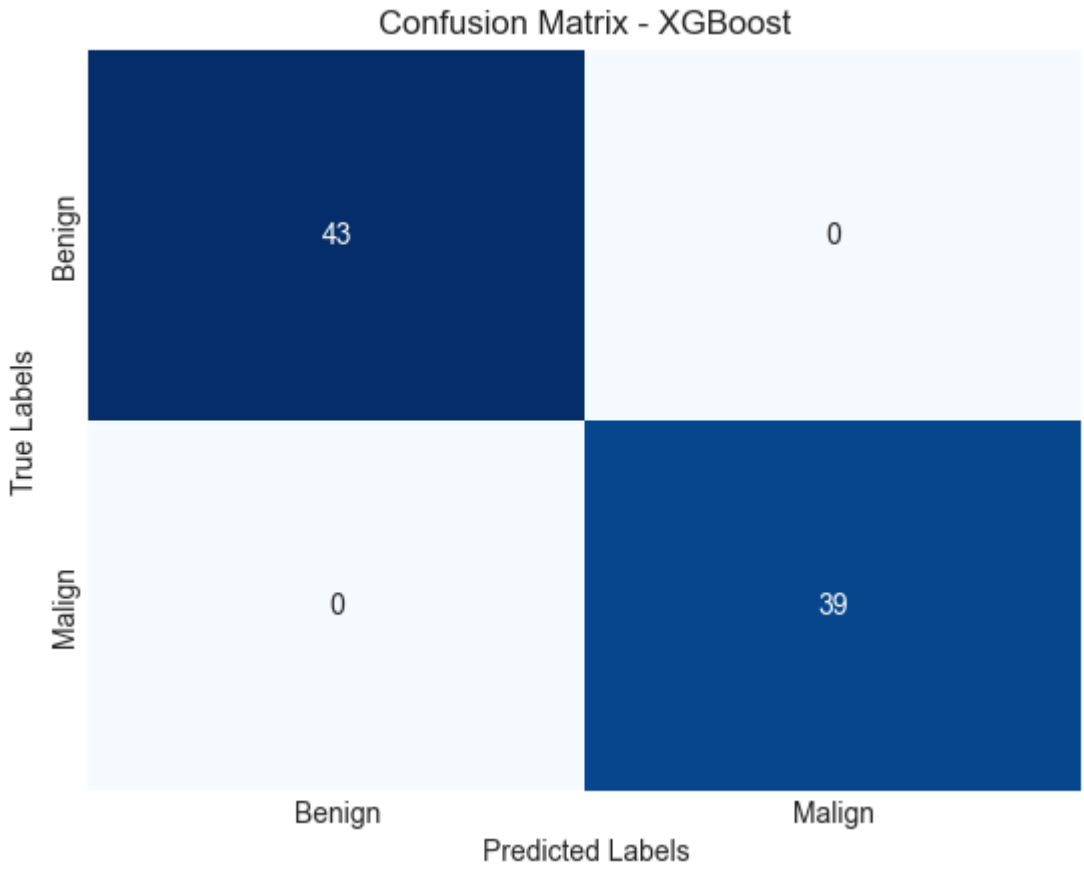
```
Número de Fold: 1
```



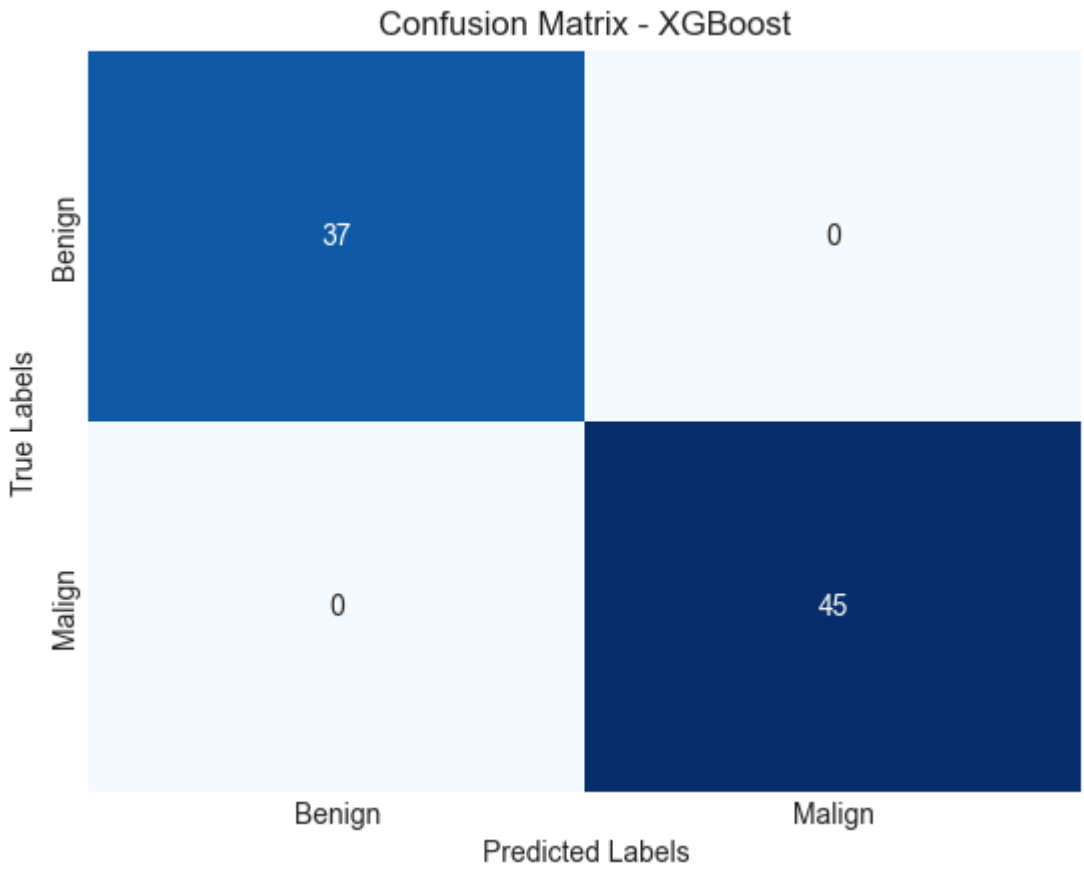

```
<Figure size 500x500 with 0 Axes>
```

Loading [MathJax]/extensions/Safe.js

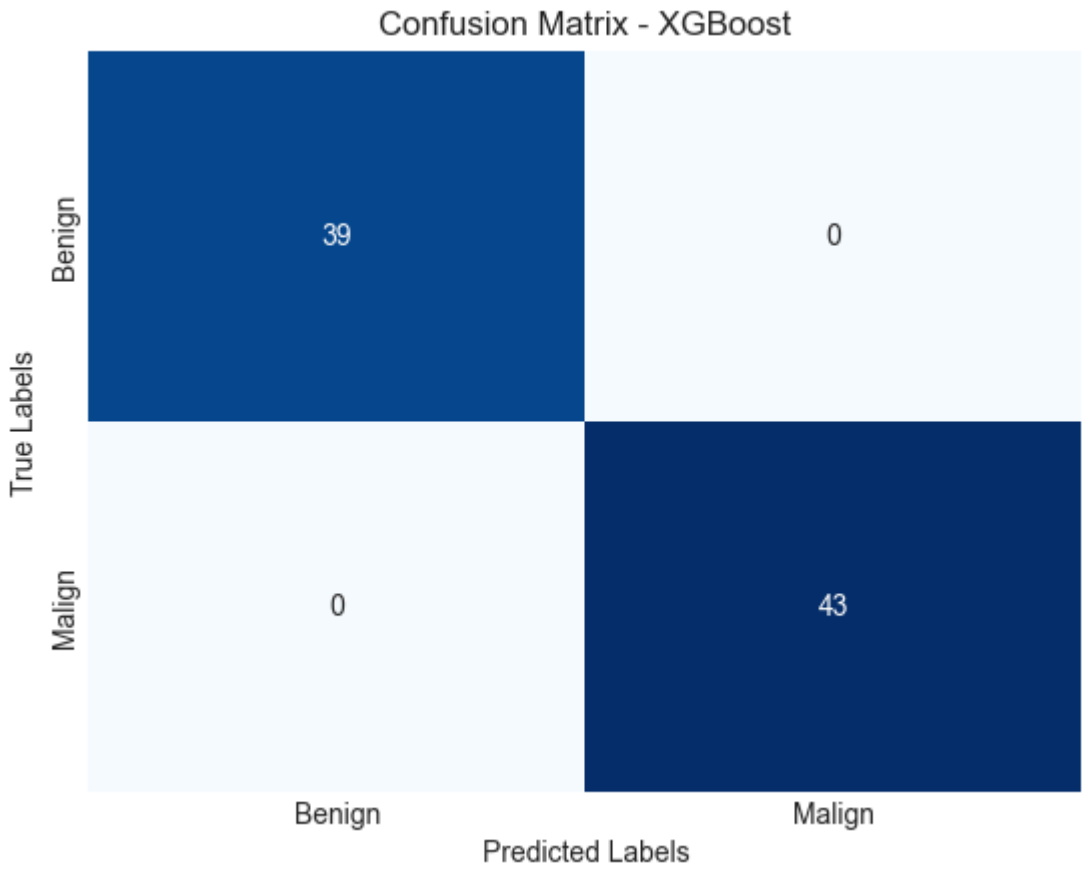
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       1.00      1.00      1.00        37

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número de Fold: 2
```



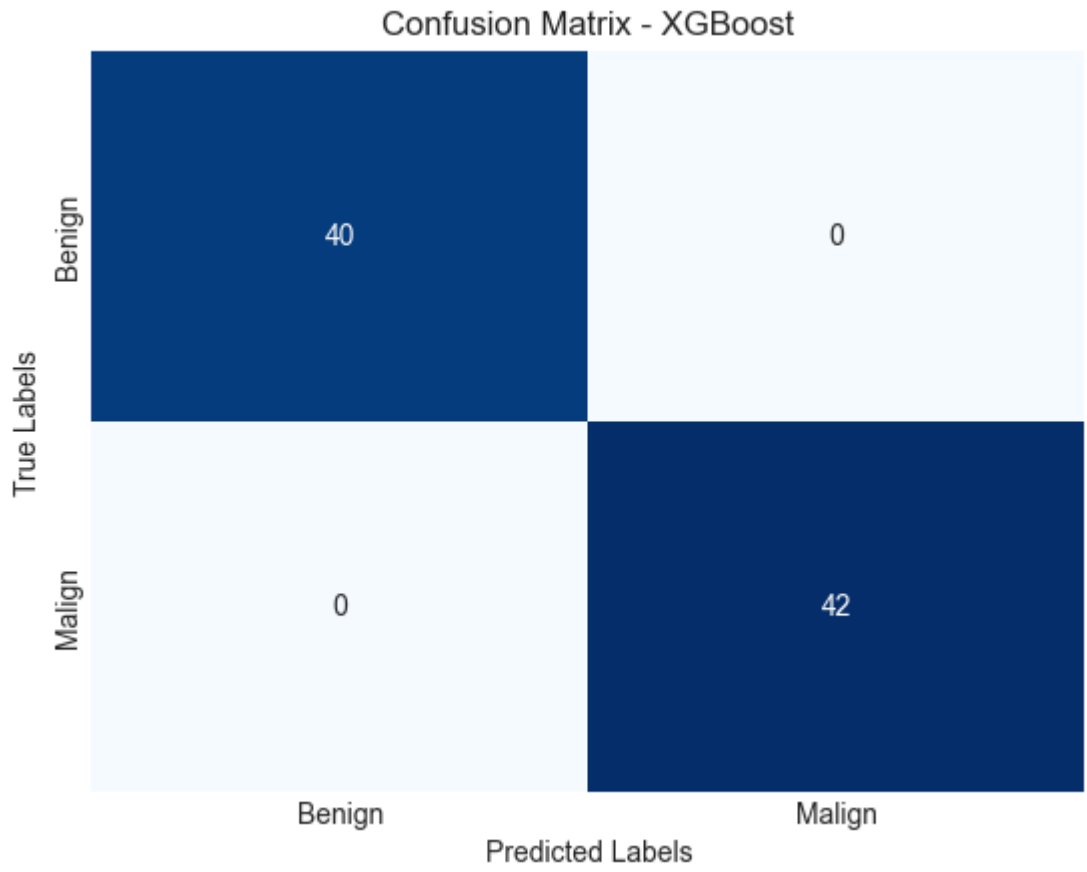

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        39

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número de Fold: 3
```



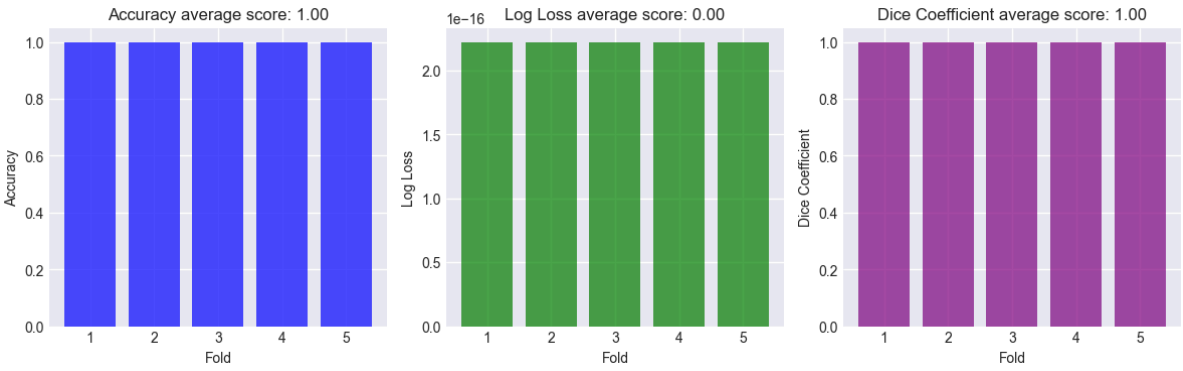

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        37
           1       1.00      1.00      1.00        45

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número de Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        39
           1       1.00      1.00      1.00        43

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número de Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       1.00      1.00      1.00        42

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82
```



XGBoost + Random Forest (seleção de features)

[Voltar a XGBoost]

Por fim, utilizamos o dataset obtido pelo Random Forest (na seleção das features)

-> 2D

In [94]:
```
data_rf_2d
```

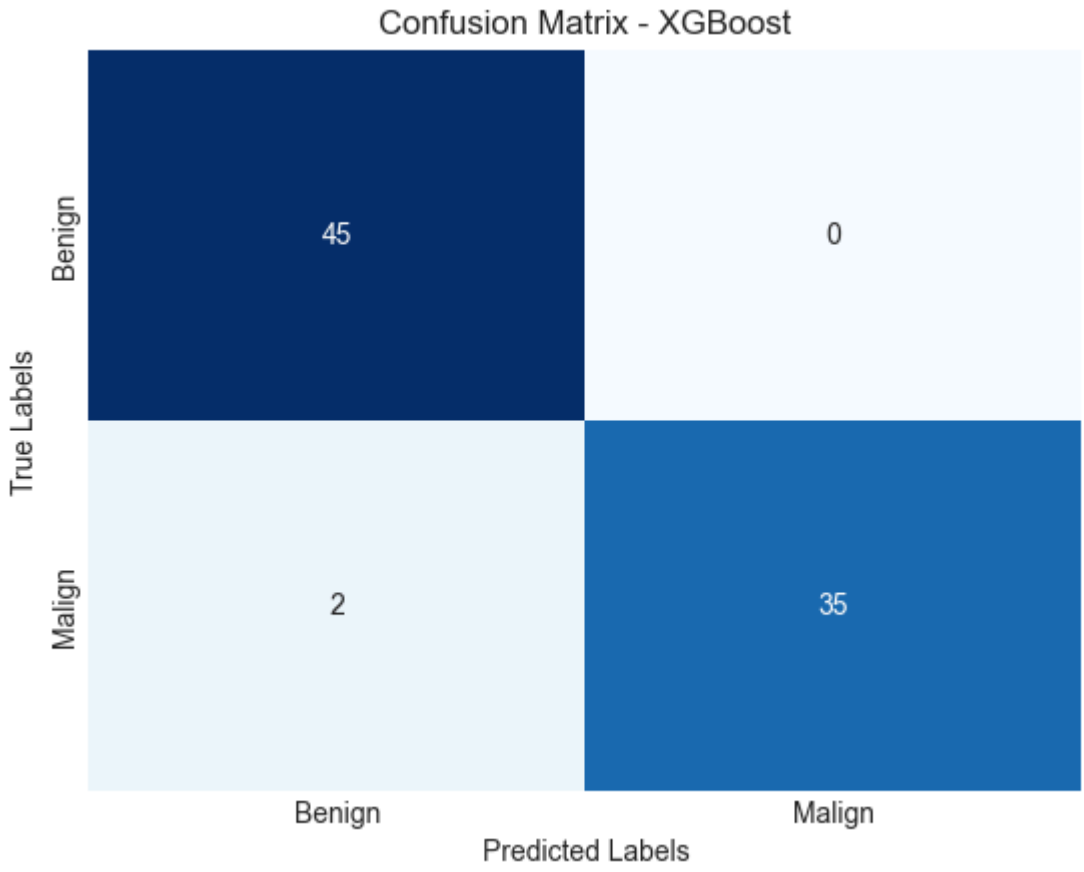
Out[94]:

	Calcification	Lobulation	Spiculation	original_gldm_GrayLevelNonUniformity	original_shape2D
0	6	3	5	160.036585	
1	6	2	2	33.000000	
2	3	1	1	12.142857	
3	6	1	5	43.079208	
4	3	1	1	18.000000	
...	...	...	...	...	
405	6	1	1	24.000000	
406	6	2	2	157.000000	
407	6	3	1	101.000000	
408	6	1	1	79.000000	
409	6	1	1	21.000000	

410 rows × 47 columns

In [95]:
```
accuracies_2d, log_losses_2d, dice_coefs_2d = xgboots_k_fold(fold_2d_rf)
th_xgb_2d.append([accuracies_2d,log_losses_2d,dice_coefs_2d])
average_metrics_2d = plot_metrics(accuracies_2d, log_losses_2d, dice_coefs_2d)
results_xgb_2d.append(average_metrics_2d)
```

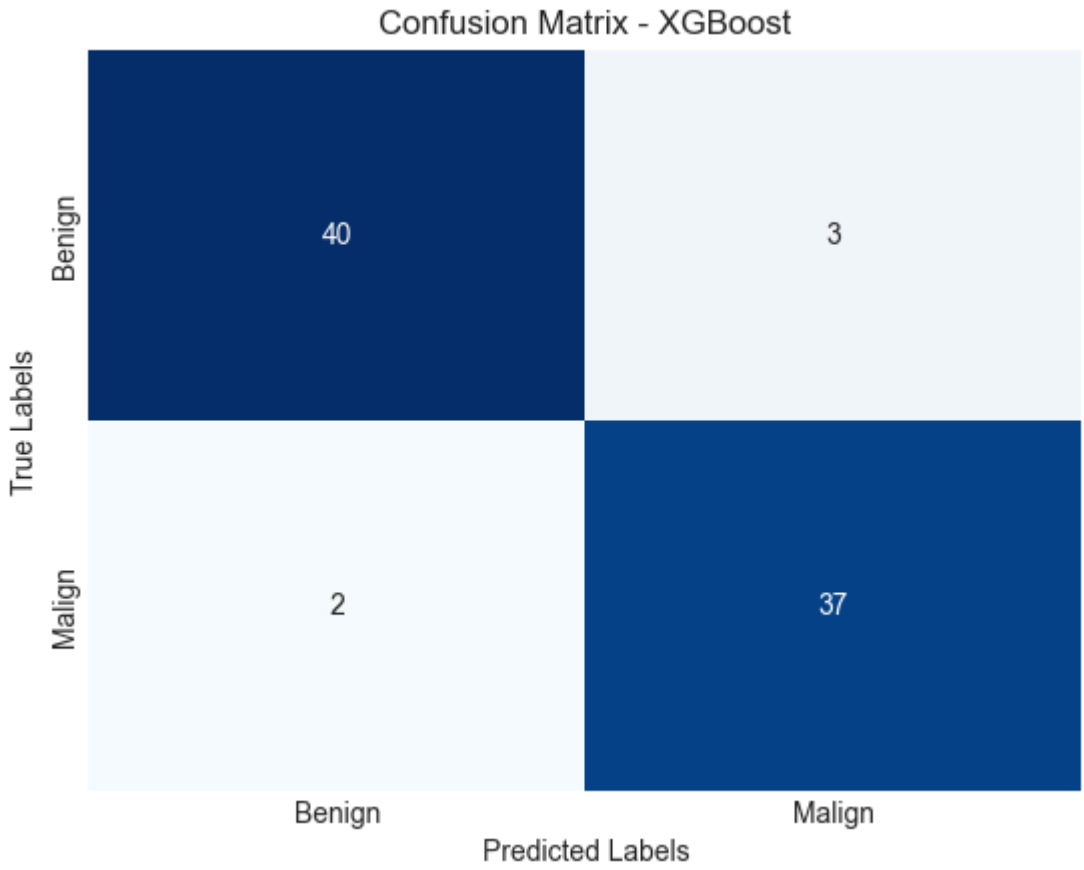
```
Número de Fold: 1
```



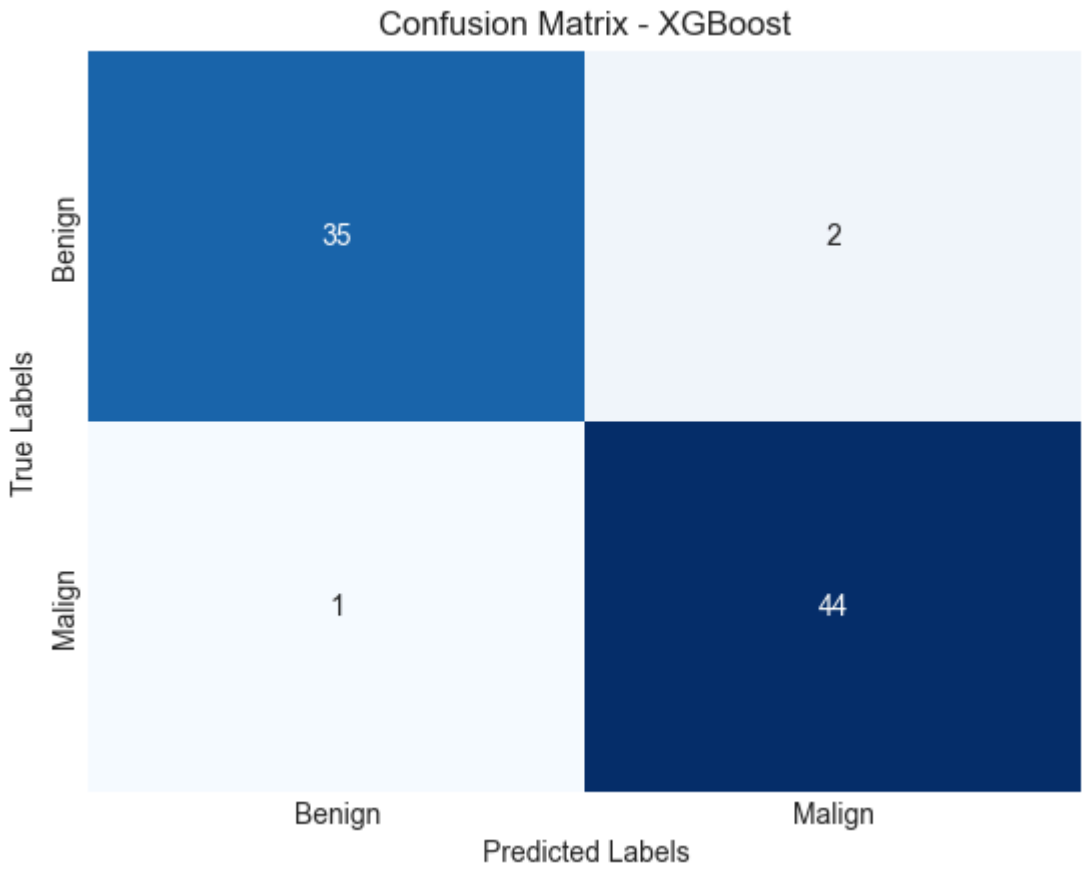

```
<Figure size 500x500 with 0 Axes>
```

Loading [MathJax]/extensions/Safe.js

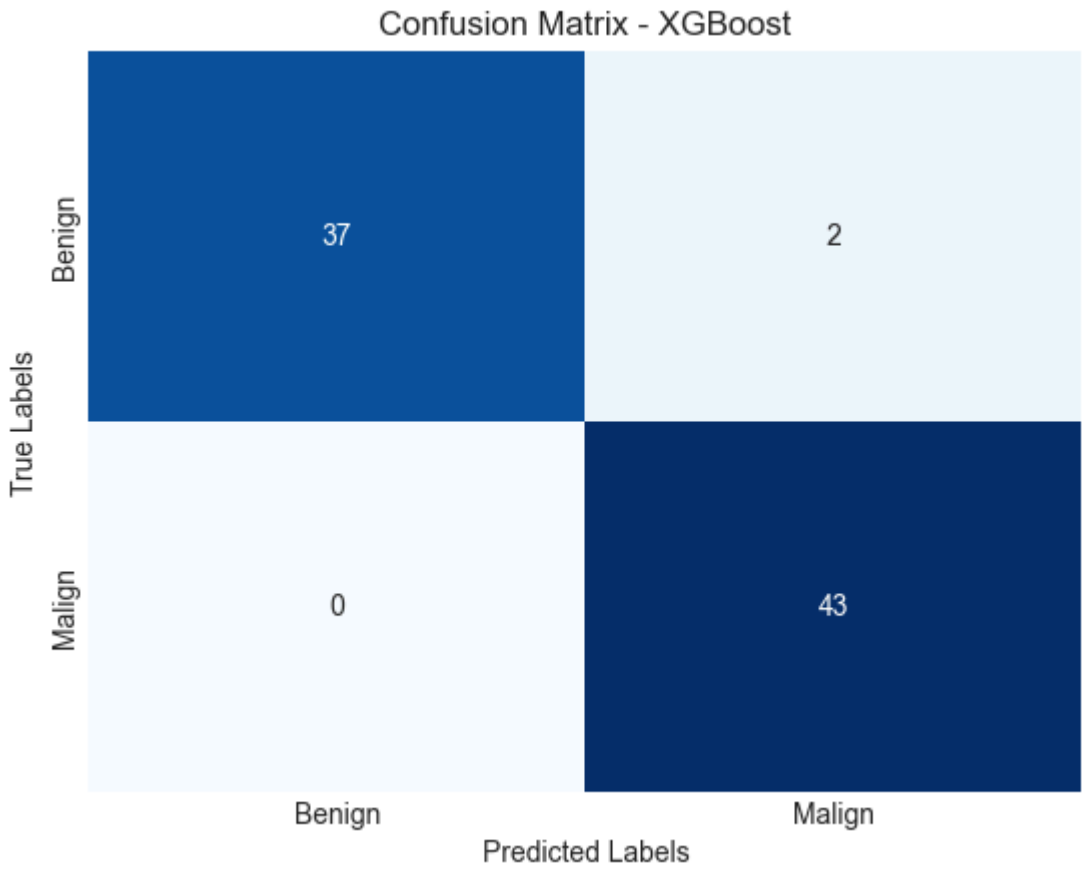
```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        45
           1       1.00      0.95      0.97        37

    accuracy                           0.98        82
   macro avg       0.98      0.97      0.98        82
weighted avg       0.98      0.98      0.98        82

Número de Fold: 2
```



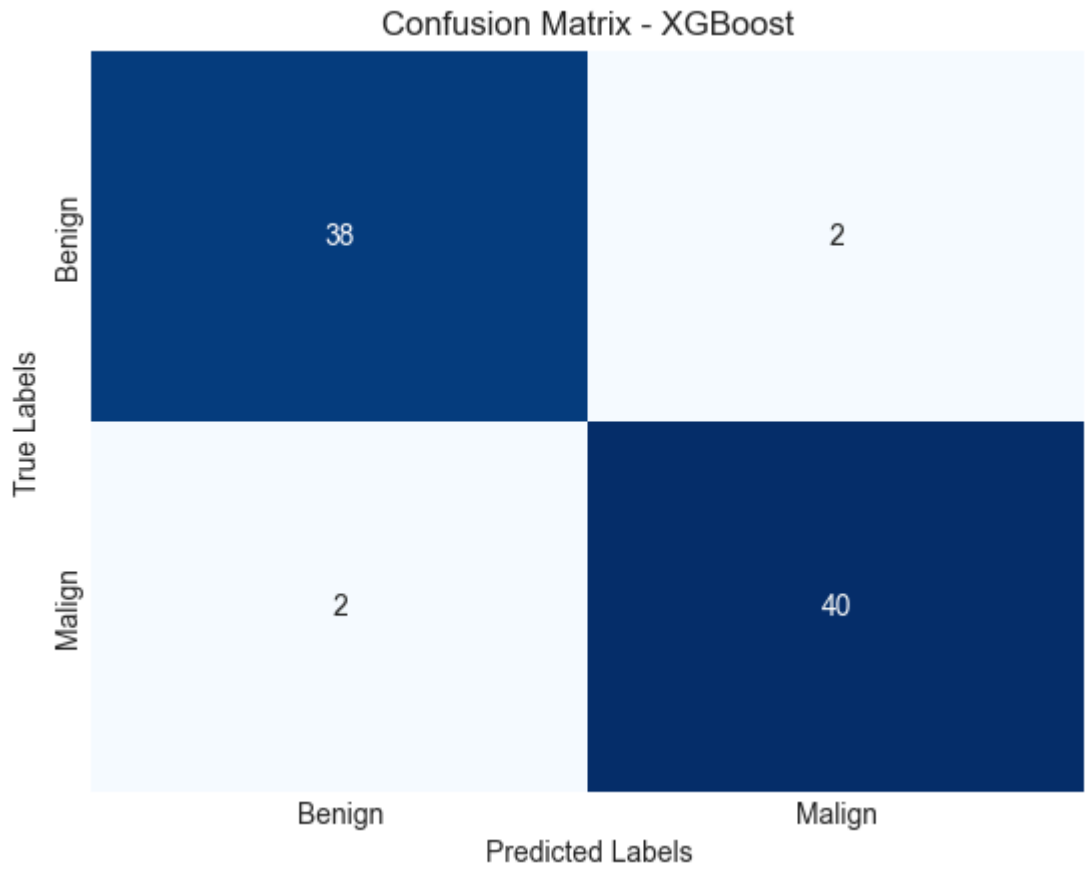

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        43
           1       0.93      0.95      0.94        39

    accuracy                           0.94        82
   macro avg       0.94      0.94      0.94        82
weighted avg       0.94      0.94      0.94        82

Número de Fold: 3
```



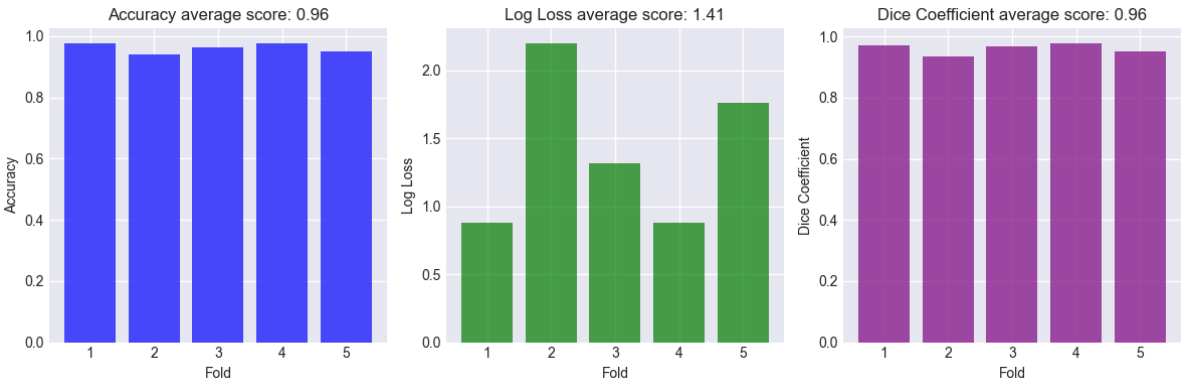

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        37
           1       0.96      0.98      0.97        45

    accuracy                           0.96        82
   macro avg       0.96      0.96      0.96        82
weighted avg       0.96      0.96      0.96        82

Número de Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        39
           1       0.96      1.00      0.98        43

    accuracy                           0.98        82
   macro avg       0.98      0.97      0.98        82
weighted avg       0.98      0.98      0.98        82

Número de Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        40
           1       0.95      0.95      0.95        42

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82
```



-> 3D

In [96]:
```
data_rf_3d
```

Loading [MathJax]/extensions/Safe.js

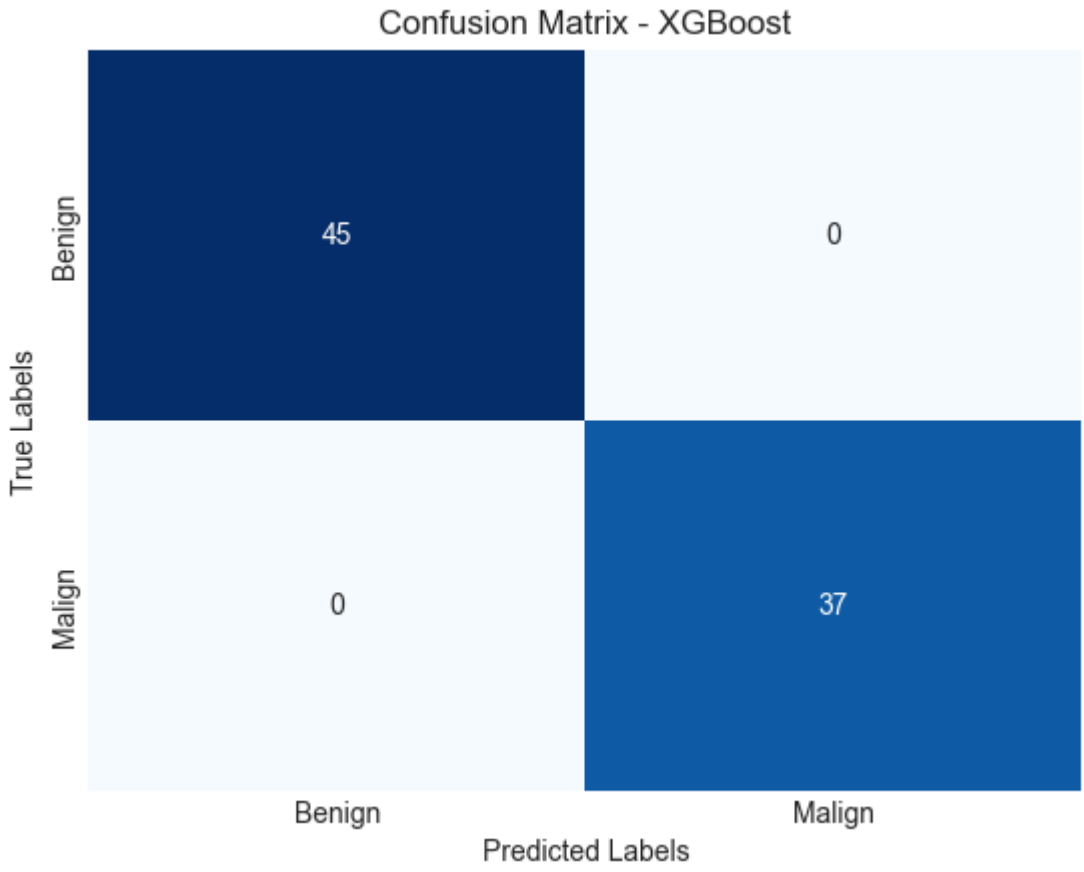
Out[96]:

	Calcification	Spiculation	Lobulation	Margin	original_glrlm_GrayLevelNonUniformity	origina
0	6	5	3	4	757.880362	
1	6	2	2	3	88.384615	
2	3	1	1	5	34.515260	
3	6	5	1	3	437.070150	
4	3	1	1	5	33.846154	
...	...	...	...	...	...	
405	6	1	1	4	41.769231	
406	6	2	2	4	1239.769231	
407	6	1	3	4	573.307692	
408	6	1	1	3	444.769231	
409	6	1	1	2	47.846154	

410 rows × 53 columns

In [97]:
```
accuracies_3d, log_losses_3d, dice_coefs_3d = xgboots_k_fold(fold_3d_rf)
th_xgb_3d.append([accuracies_3d, log_losses_3d, dice_coefs_3d])
average_metrics_3d = plot_metrics(accuracies_3d, log_losses_3d, dice_coefs_3d)
results_xgb_3d.append(average_metrics_3d)
```

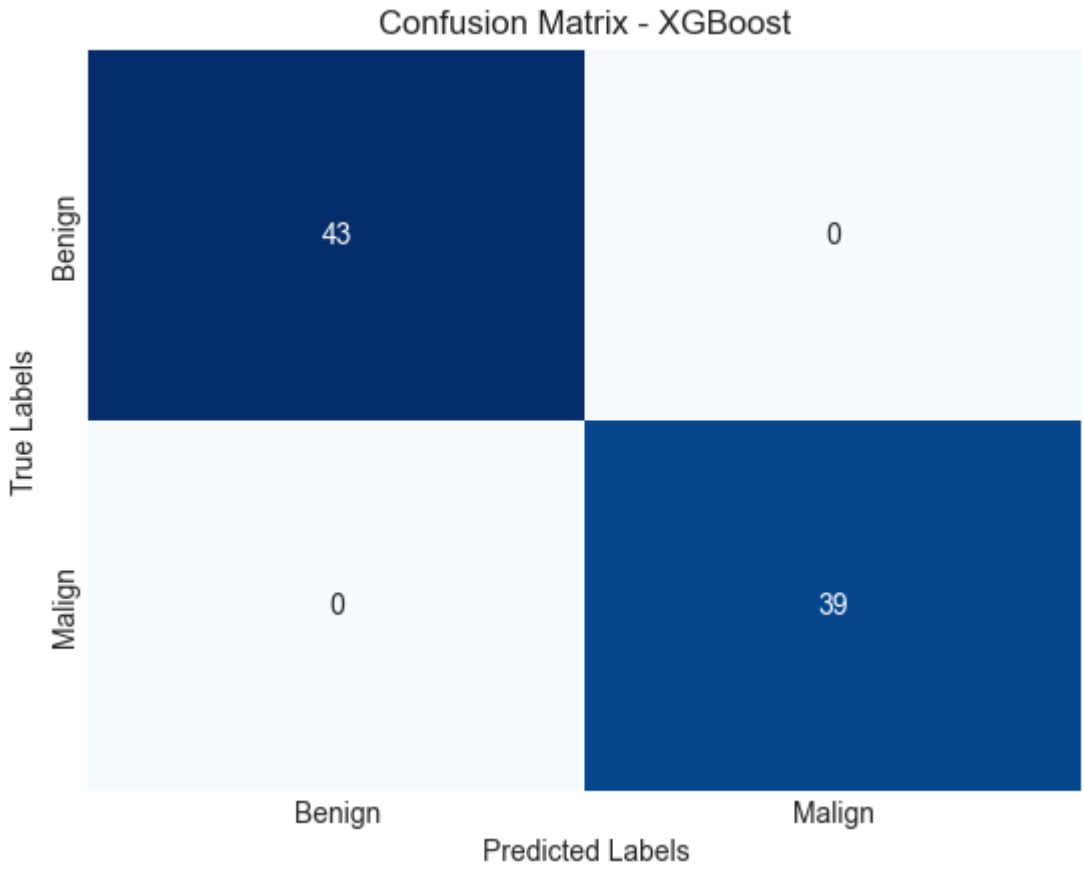
```
Número de Fold: 1
```



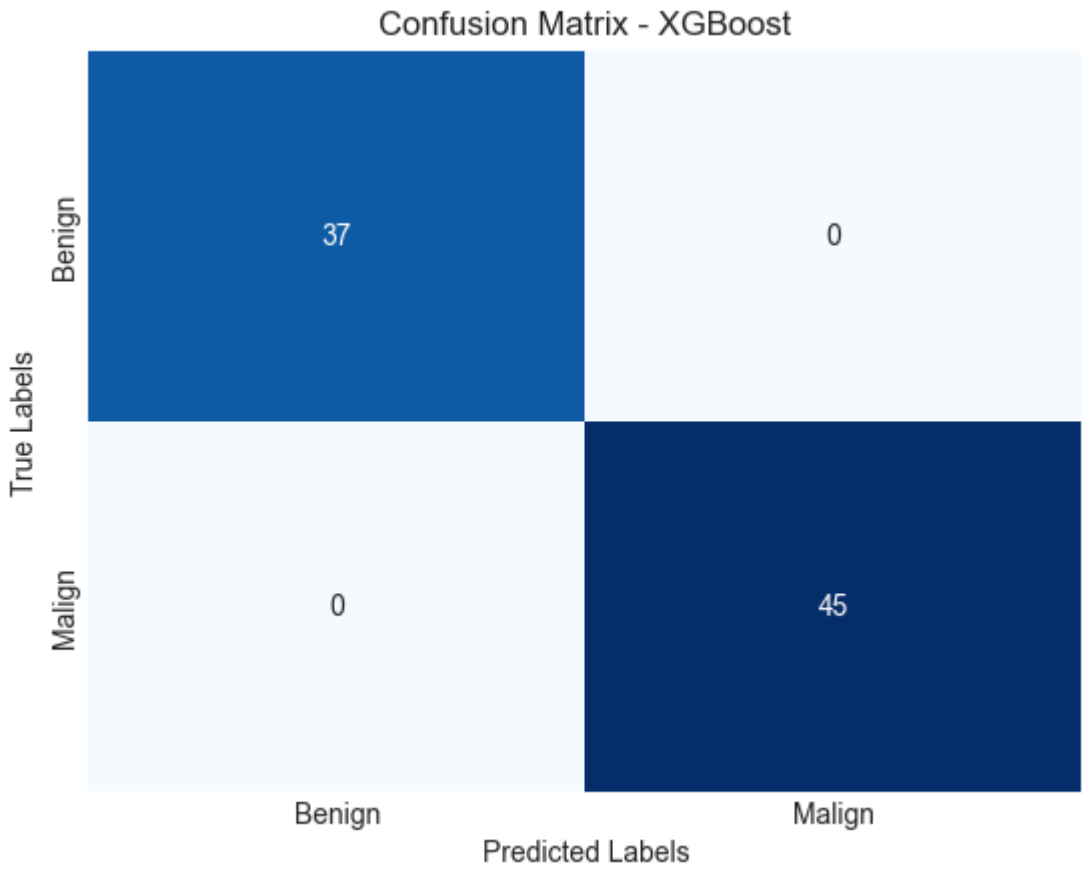

```
<Figure size 500x500 with 0 Axes>
```

Loading [MathJax]/extensions/Safe.js

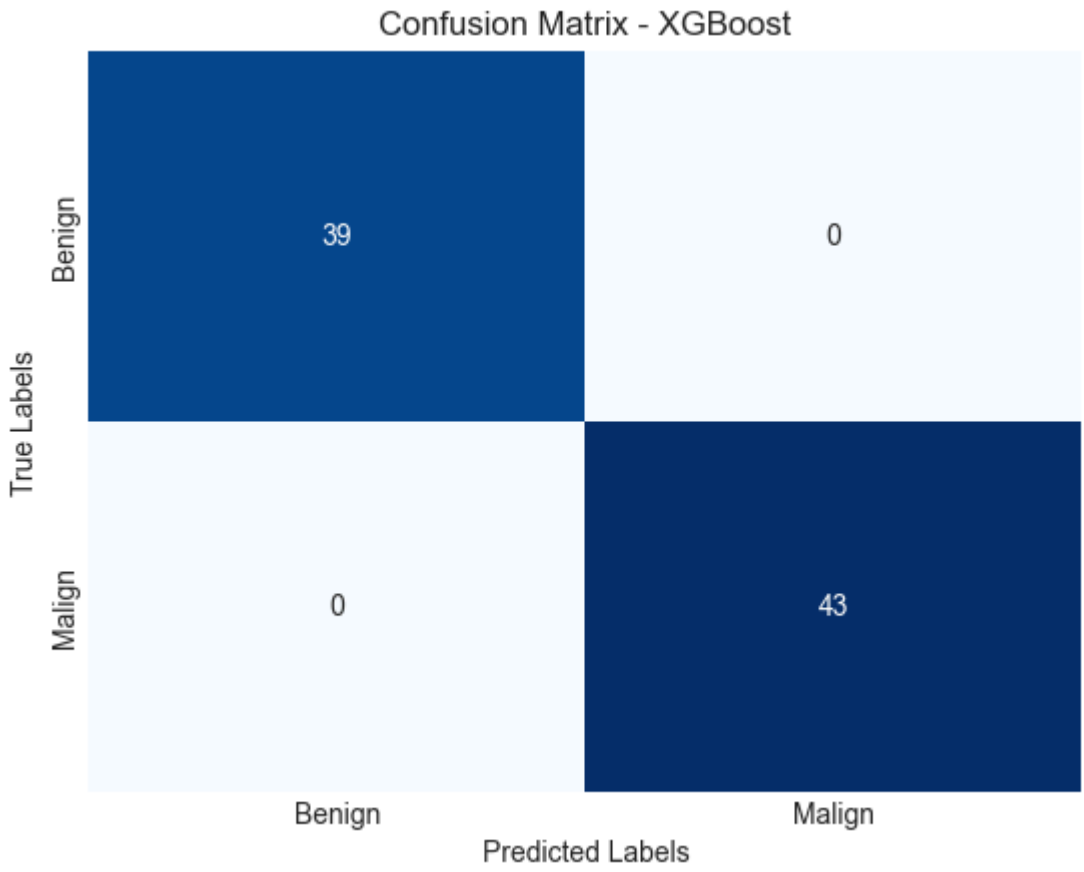
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       1.00      1.00      1.00        37

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número de Fold: 2
```



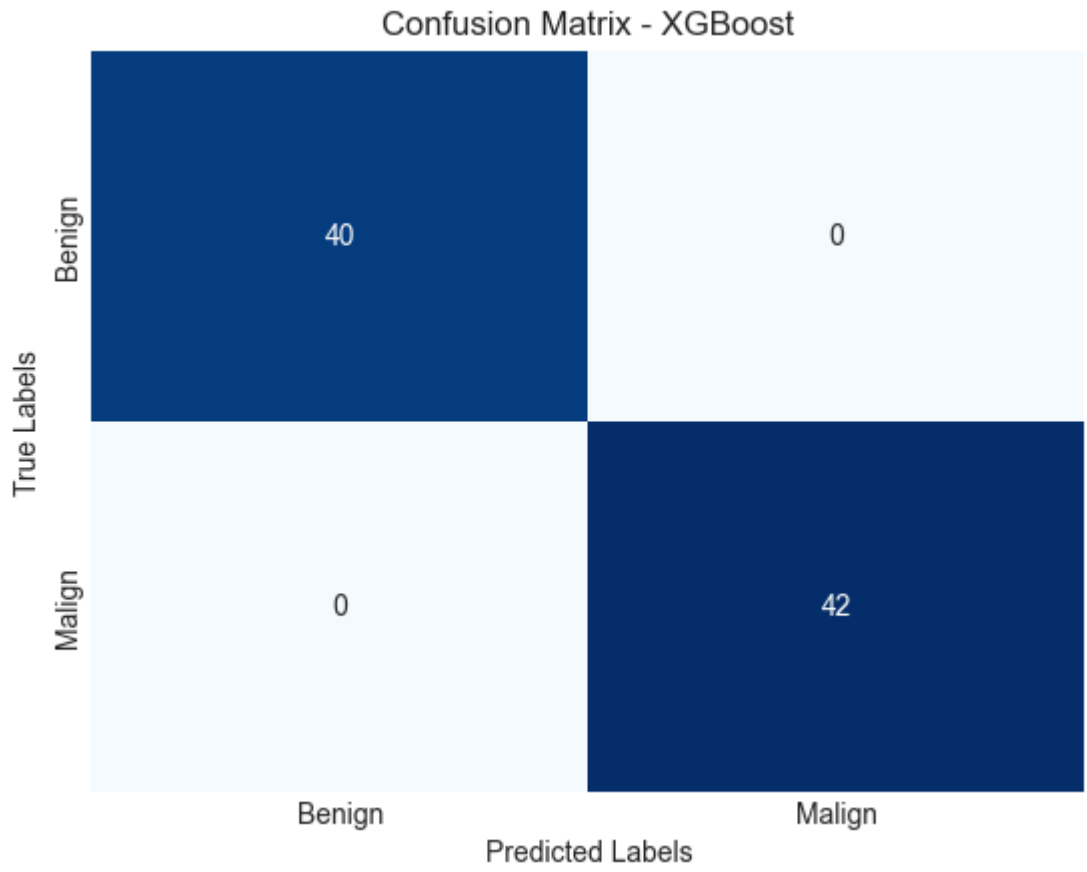

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        39

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número de Fold: 3
```



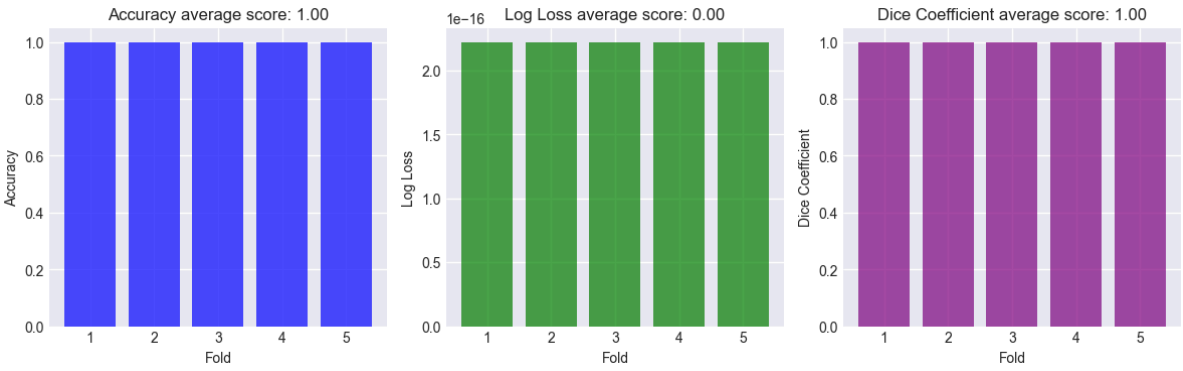

```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        37
           1       1.00      1.00      1.00        45

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número de Fold: 4
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        39
           1       1.00      1.00      1.00        43

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82

Número de Fold: 5
```




```
<Figure size 500x500 with 0 Axes>
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       1.00      1.00      1.00        42

    accuracy                           1.00        82
   macro avg       1.00      1.00      1.00        82
weighted avg       1.00      1.00      1.00        82
```



Resultados XGBoost

[Voltar a XGBoost]

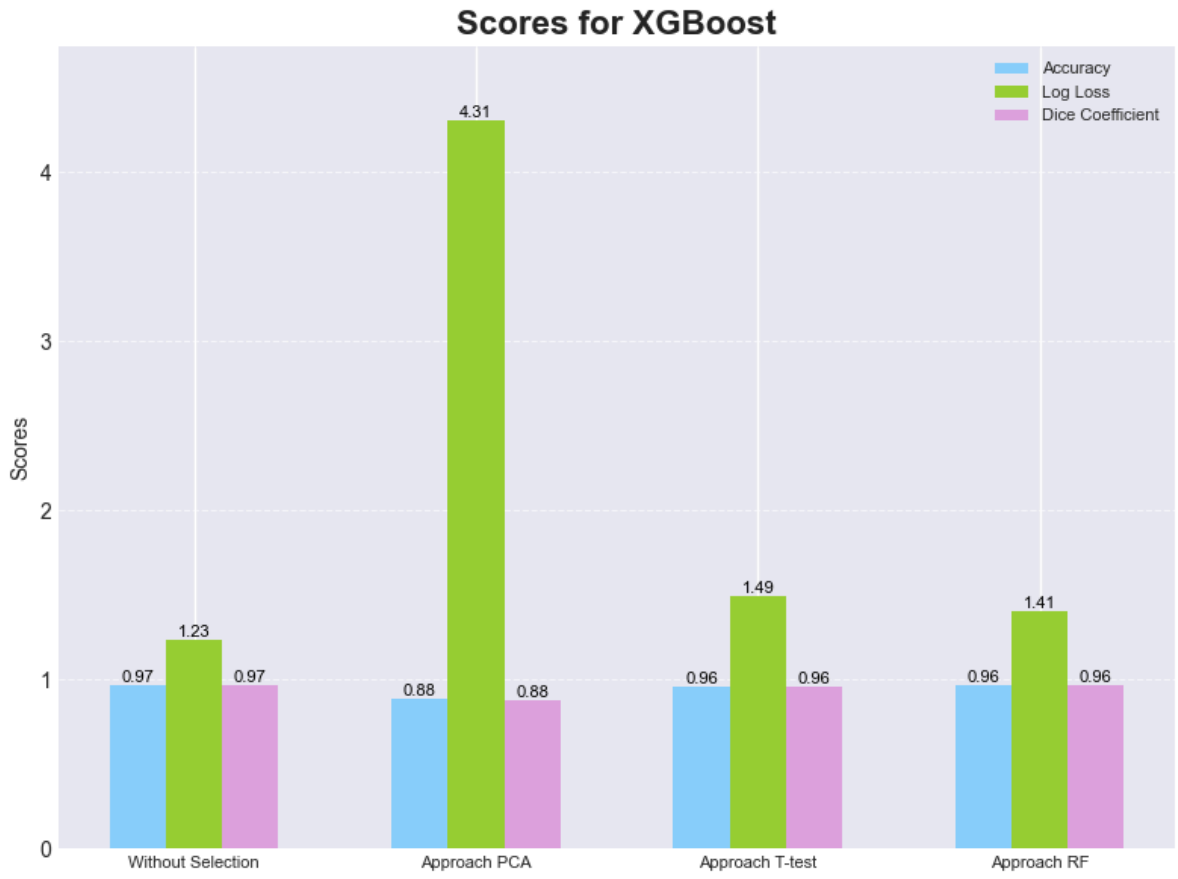
-> 2D

In [98]:
```
print(results_xgb_2d)
```

```
[[0.9658536585365853, 1.2307588962137566, 0.9657709440318136], [0.880487804878048, 4.307656136748148, 0.8795964035964035], [0.9585365853658537, 1.494492945402418, 0.9576821206195625], [0.9609756097560975, 1.4065815956728644, 0.9611235459336726]]
```

In [99]:
```
plot_scores(results_xgb_2d, "XGBoost")
```

```
/var/folders/2t/mv0q1c7j2z97cgvtlhbfrhhm0000gn/T/ipykernel_67329/3843270632.py:9: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn-darkgrid')
```



-> 3D

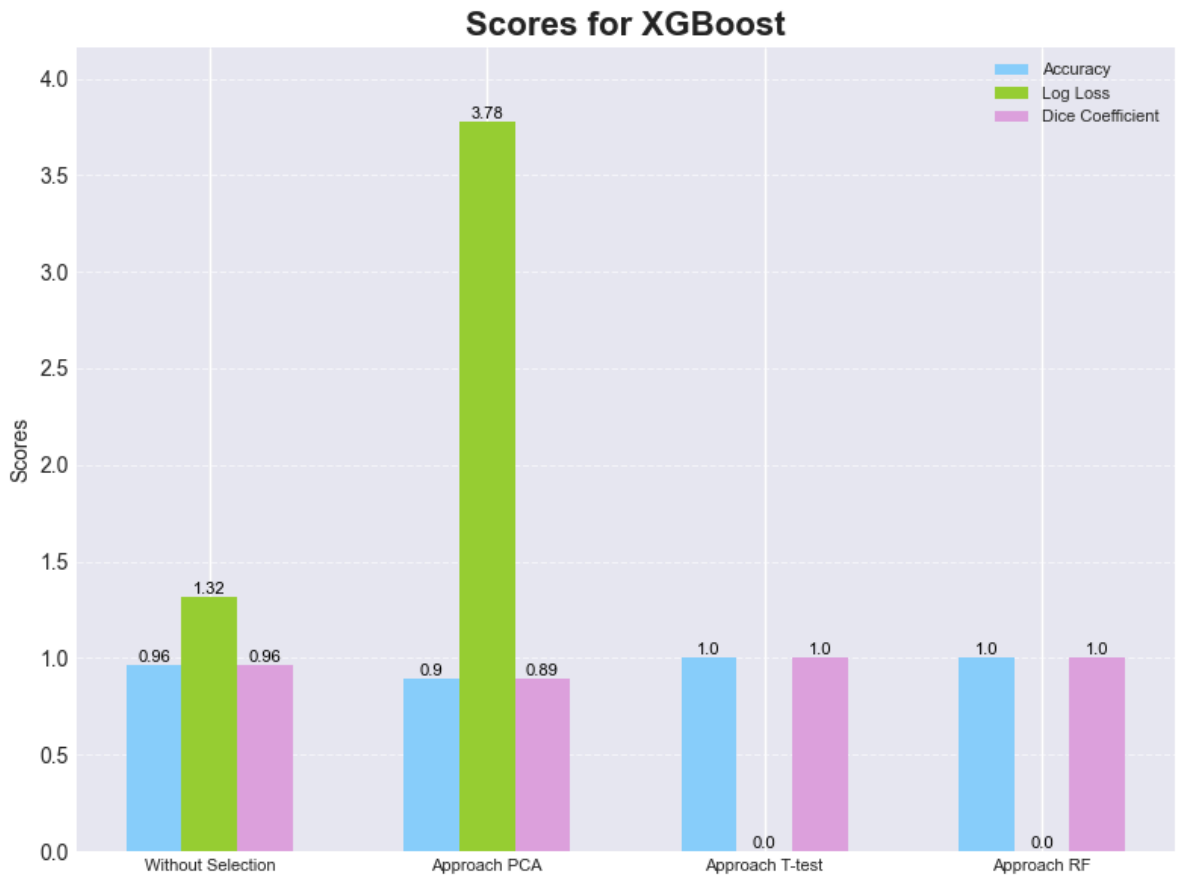
In [100…
```
print(results_xgb_3d)
```

```
[[0.9634146341463415, 1.3186702459433106, 0.9638394170903058], [0.8951219512195122, 3.780188038370823, 0.8942145990591273], [1.0, 2.2204460492503136e-16, 1.0], [1.0, 2.2204460492503136e-16, 1.0]]
```

In [101…
```
plot_scores(results_xgb_3d, "XGBoost")
```

```
/var/folders/2t/mv0q1c7j2z97cgvtlhbfrhhm0000gn/T/ipykernel_67329/3843270632.py:9: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn-darkgrid')
```

Loading [MathJax]/extensions/Safe.js



Comparação de resultados

[Voltar ao Indice]

De modo a termos uma visão do geral dos resultados obtidos, criamos a seguinte função que nos permite ter em conta todos os resultados:

In [102…

```
def plot_all_models_combined(results_svm, results_rf, results_xgb):
    models = ['SVM', 'Random Forest', 'XGBoost']
    approaches = ['Without Selection', 'Approach PCA', 'Approach T-test', 'Approach
    metrics = ['Accuracy', 'Log Loss', 'Dice Coefficient']

    # Preparar dados em uma estrutura única
    all_results = [results_svm, results_rf, results_xgb]
    bar_width = 0.2
    x = np.arange(len(approaches))

    # Verificação de estrutura
    for model, results in zip(models, all_results):
        if len(results) != len(approaches):
            raise ValueError(f"O número de abordagens em {model} ({len(results)}) r
        for result in results:
            if len(result) != len(metrics):
                raise ValueError(f"Cada abordagem em {model} deve ter exatamente {l

    fig, ax = plt.subplots(figsize=(15, 8))

    # Plotar cada métrica com barras agrupadas
    for i, (model, results) in enumerate(zip(models, all_results)):
        for j, metric in enumerate(metrics):
            # Extrair apenas os dados da métrica específica para cada abordagem
            ric_scores = [result[j] for result in results]
```

```
            # Posicionar as barras de forma que elas fiquem agrupadas por métrica e
            ax.bar(x + i * bar_width + j * bar_width / len(metrics), metric_scores,
                   width=bar_width / len(metrics), label=f'{model} - {metric}')

    # Configurar o gráfico
    ax.set_xlabel('Approaches')
    ax.set_ylabel('Scores')
    ax.set_title('Comparison of Models and Metrics Across Approaches')
    ax.set_xticks(x + bar_width)
    ax.set_xticklabels(approaches)
    ax.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

    plt.tight_layout()
    plt.show()
```

-> 2D

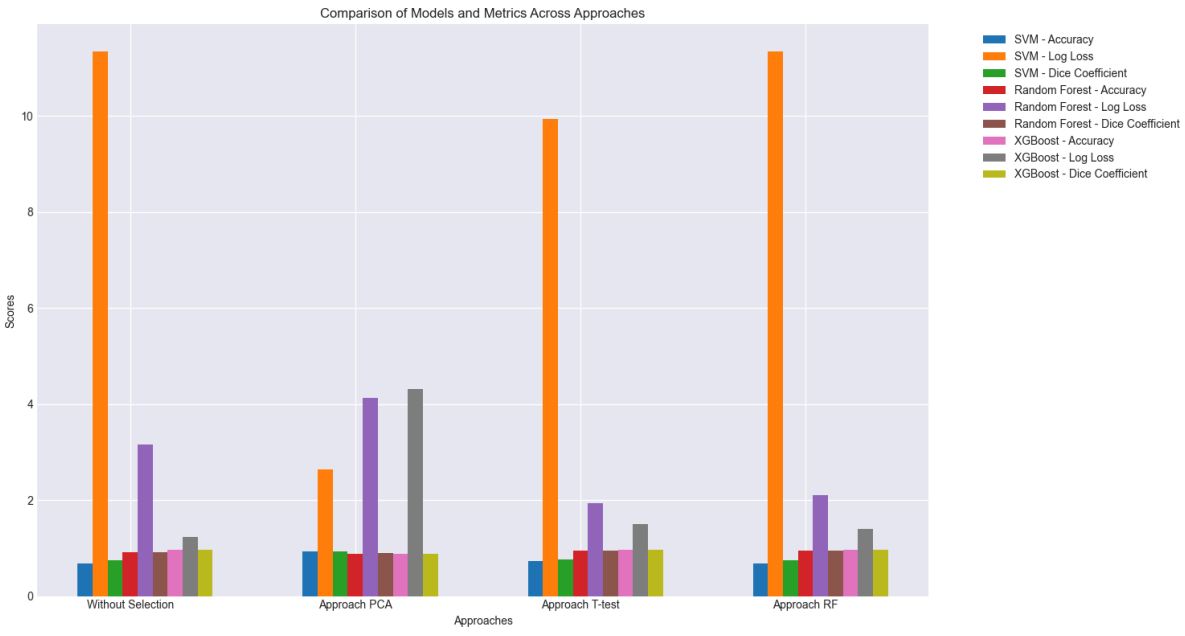
In [103…

```
print(results_svm_2d)
print(results_rf_2d)
print(results_xgb_2d)
```

```
[[0.6853658536585366, 11.34056411511247, 0.7418117663125008], [0.9268292682926829,
2.637340491886621, 0.9254514415467104], [0.724390243902439, 9.933982519439606, 0.7
593110494818275], [0.6853658536585366, 11.34056411511247, 0.7418117663125008]]
[[0.9121951219512194, 3.164808590263945, 0.9127866022021893], [0.8853658536585366,
4.13183343728904, 0.8890888664025738], [0.9463414634146341, 1.9340496940501886, 0.
9458706597285762], [0.9414634146341463, 2.109872393509297, 0.9403221775286242]]
[[0.9658536585365853, 1.2307588962137566, 0.9657709440318136], [0.880487804878048
8, 4.307656136748148, 0.8795964035964035], [0.9585365853658537, 1.494492945402418
7, 0.9576821206195625], [0.9609756097560975, 1.4065815956728644, 0.961123545933672
6]]
```

In [104…

```
plot_all_models_combined(results_svm_2d, results_rf_2d, results_xgb_2d)
```



-> 3D

In [105…

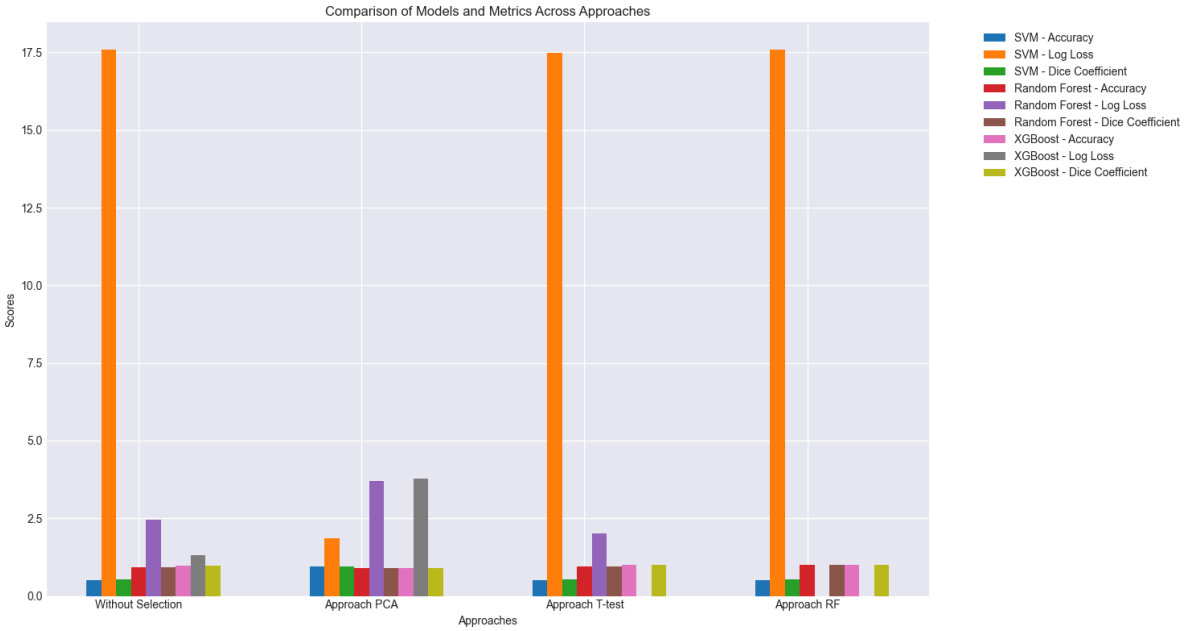
```
print(results_svm_3d)
print(results_rf_3d)
print(results_xgb_3d)
```

Loading [MathJax]/extensions/Safe.js

```
[[0.5121951219512195, 17.582269945910802, 0.5379390829067068], [0.948780487804878
1, 1.8461383443206347, 0.949257284252871], [0.5146341463414634, 17.49435859618125,
0.5391354936744037], [0.5121951219512195, 17.582269945910802, 0.5379390829067068]]
[[0.9317073170731707, 2.461517792427513, 0.9320121025795863], [0.8975609756097562,
3.692276688641269, 0.8955214018715599], [0.9439024390243903, 2.0219610437797426,
0.9430966701858932], [1.0, 2.2204460492503136e-16, 1.0]]
[[0.9634146341463415, 1.3186702459433106, 0.9638394170903058], [0.895121951219512
2, 3.780188038370823, 0.8942145990591273], [1.0, 2.2204460492503136e-16, 1.0], [1.
0, 2.2204460492503136e-16, 1.0]]
```

In [106…

```
plot_all_models_combined(results_svm_3d, results_rf_3d, results_xgb_3d)
```



Embora as médias forneçam uma visão preliminar do desempenho central de cada configuração, elas não permitem concluir se as diferenças observadas são estatisticamente significativas ou apenas resultados do acaso. Por esse motivo iremos agora realizar testes de hipóteses, que são preferíveis à simples análise das médias porque eles oferecem uma forma estatisticamente rigorosa de avaliar diferenças de desempenho entre os modelos e métodos de seleção de features.

Comparação entre diferentes modelos para o mesmo método de seleção de features

Este código foi desenvolvido para comparar o desempenho de três modelos de machine learning (SVM, Random Forest e XGBoost) em diferentes métodos de extração de features (Sem Seleção, PCA, t-test e Random Forest) usando três métricas de avaliação: accuracy, log loss e dice coefficient. Cada modelo foi avaliado utilizando 5 folds (ou partições) dos dados, e as métricas foram calculadas para cada fold em cada método de extração de features.

O principal objetivo deste código é identificar, com base em testes de hipóteses, se algum dos três modelos (SVM, Random Forest ou XGBoost) apresenta um desempenho significativamente superior aos outros em cada combinação de método de extração de features e métrica. Ao realizar essa análise, é possível escolher o modelo de machine learning mais adequado para cada cenário específico de extração de features, maximizando [illegible]nho de acordo com a métrica que é mais importante para a aplicação.

Uso dos Testes de Hipóteses Para identificar se há diferenças significativas no desempenho dos modelos, o código utiliza uma abordagem baseada em testes de hipóteses. Os testes aplicados incluem:

Teste ANOVA (Análise de Variância):

O teste ANOVA é utilizado para verificar se há diferenças estatisticamente significativas entre as médias das métricas dos três modelos (SVM, Random Forest, XGBoost) para cada método de extração de features. Para cada métrica (accuracy, log loss, e dice coefficient), o teste ANOVA compara as médias dos valores obtidos pelos modelos. Se o teste indicar uma diferença significativa, isso sugere que pelo menos um dos modelos apresenta um desempenho diferente dos outros para a métrica e o método de extração de features em questão. Quando o ANOVA identifica diferenças significâncias (com um p-valor menor que 0,05), passamos a considerar que pode haver diferenças reais entre os modelos.

Teste de Comparações Múltiplas de Tukey (Post Hoc):

Quando o ANOVA detecta uma diferença significativa, o código realiza uma análise post hoc com o teste de Tukey. O teste de Tukey é um teste de comparações múltiplas que nos permite identificar exatamente que pares de modelos são significativamente diferentes entre si. Isso é crucial para determinar se um modelo é consistentemente superior ou inferior aos outros em termos de uma métrica específica. Por exemplo, ele pode mostrar que um modelo (como o XGBoost) é estatisticamente melhor em accuracy do que o SVM e o Random Forest para um método de extração específico.

In [107…

```
def avaliar_modelos_por_metodo(th_svm, th_rf, th_xgb):
    # Nome das métricas e modelos para análise
    metricas = ["Accuracy", "Log Loss", "Dice Coefficient"]
    modelos = ["SVM", "Random Forest", "XGBoost"]
    metodos = ["Sem Seleção", "PCA", "t-test", "Random Forest"]

    # Iterar sobre cada método de extração de features
    for metodo_idx, metodo in enumerate(metodos):
        print(f"\nAnalisando método de extração de features: {metodo}\n" + "="*50)

        melhores_modelos = {}

        # Iterar sobre cada métrica (accuracy, log loss, dice coefficient)
        for metrica_idx, metrica in enumerate(metricas):
            print(f"\nMétrica: {metrica}\n" + "-"*50)

            # Extrair os dados para a métrica específica de cada modelo
            dados_metrica = [
                th_svm[metodo_idx][metrica_idx],  # Dados para SVM no método e métr
                th_rf[metodo_idx][metrica_idx],   # Dados para Random Forest no mét
                th_xgb[metodo_idx][metrica_idx]   # Dados para XGBoost no método e
            ]

            # Calcular as médias para cada modelo para a métrica em questão
            medias = [np.mean(valores) for valores in dados_metrica]

            # Executar o teste ANOVA de uma via para comparar as médias entre os mo
            f_stat, p_val = stats.f_oneway(*dados_metrica)

            if p_val < 0.05:
```

```
                print(f"Diferença significativa encontrada na métrica {metrica} (p

                # Preparar dados para o teste de Tukey
                dados_flat = np.concatenate(dados_metrica)
                grupos = np.array([modelo for modelo in modelos for _ in range(5)])

                # Executar o teste post hoc de Tukey para identificar quais modelos
                tukey_result = pairwise_tukeyhsd(dados_flat, grupos, alpha=0.05)
                print(tukey_result)

                # Determinar o melhor modelo com base nas comparações de Tukey
                melhores = []
                if metrica == "Log Loss":
                    # Para log loss, menor média é melhor
                    for j, media in enumerate(medias):
                        # Verificar se o modelo atual é significativamente melhor q
                        significativo = all(
                            tukey_result.reject[idx]
                            for idx, group in enumerate(tukey_result.groupsunique)
                            if modelos[j] == group
                        )
                        if significativo:
                            melhores.append((modelos[j], media))
                    melhor_modelo = min(melhores, key=lambda x: x[1])[0] if melhore

                else:
                    # Para accuracy e dice coefficient, maior média é melhor
                    for j, media in enumerate(medias):
                        # Verificar se o modelo atual é significativamente melhor q
                        significativo = all(
                            tukey_result.reject[idx]
                            for idx, group in enumerate(tukey_result.groupsunique)
                            if modelos[j] == group
                        )
                        if significativo:
                            melhores.append((modelos[j], media))
                    melhor_modelo = max(melhores, key=lambda x: x[1])[0] if melhore

                melhores_modelos[metrica] = melhor_modelo
                print(f"Melhor modelo para {metrica} com base no teste de hipóteses

            else:
                print(f"Não há diferença significativa entre os modelos para a métr
                melhores_modelos[metrica] = "Nenhum modelo se destacou"

        # Exibir o resumo dos melhores modelos para o método atual
        print(f"\nResumo dos melhores modelos para o método de extração '{metodo}':
        for metrica, modelo in melhores_modelos.items():
            print(f"{metrica}: {modelo}")
```

In [108…

```
melhores_modelos = avaliar_modelos_por_metodo(th_svm_2d, th_rf_2d, th_xgb_2d)
print("\nMelhores modelos por método de seleção:", melhores_modelos)
```

Loading [MathJax]/extensions/Safe.js

```
Analisando método de extração de features: Sem Seleção
==================================================

Métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0000)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
------------------------------------------------------------
Random Forest     SVM  -0.2268    0.0 -0.2787 -0.1749   True
Random Forest XGBoost   0.0537 0.0427  0.0017  0.1056   True
          SVM XGBoost   0.2805    0.0  0.2286  0.3324   True
------------------------------------------------------------
Melhor modelo para Accuracy com base no teste de hipóteses: XGBoost

Métrica: Log Loss
--------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0000)
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower    upper  reject
------------------------------------------------------------
Random Forest     SVM   8.1758    0.0   6.3044 10.0471   True
Random Forest XGBoost   -1.934 0.0427  -3.8054 -0.0627   True
          SVM XGBoost -10.1098    0.0 -11.9812 -8.2384   True
------------------------------------------------------------
Melhor modelo para Log Loss com base no teste de hipóteses: XGBoost

Métrica: Dice Coefficient
--------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0000)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
------------------------------------------------------------
Random Forest     SVM   -0.171    0.0 -0.2192 -0.1228   True
Random Forest XGBoost    0.053 0.0313  0.0048  0.1012   True
          SVM XGBoost    0.224    0.0  0.1758  0.2722   True
------------------------------------------------------------
Melhor modelo para Dice Coefficient com base no teste de hipóteses: XGBoost

Resumo dos melhores modelos para o método de extração 'Sem Seleção':
Accuracy: XGBoost
Log Loss: XGBoost
Dice Coefficient: XGBoost

Analisando método de extração de features: PCA
==================================================

Métrica: Accuracy
--------------------------------------------------
Não há diferença significativa entre os modelos para a métrica Accuracy (p = 0.250
5)

Métrica: Log Loss
--------------------------------------------------
Não há diferença significativa entre os modelos para a métrica Log Loss (p = 0.250
5)

Métrica: Dice Coefficient
--------------------------------------------------
Não há diferença significativa entre os modelos para a métrica Dice Coefficient (p
= 0.3111)
```

```
Resumo dos melhores modelos para o método de extração 'PCA':
Accuracy: Nenhum modelo se destacou
Log Loss: Nenhum modelo se destacou
Dice Coefficient: Nenhum modelo se destacou

Analisando método de extração de features: t-test
==================================================

Métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0000)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower  upper  reject
------------------------------------------------------------
Random Forest     SVM   -0.222    0.0 -0.2769 -0.167   True
Random Forest XGBoost   0.0122 0.8269 -0.0428 0.0672  False
          SVM XGBoost   0.2341    0.0  0.1792 0.2891   True
------------------------------------------------------------
Melhor modelo para Accuracy com base no teste de hipóteses: XGBoost

Métrica: Log Loss
--------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0000)
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower    upper  reject
------------------------------------------------------------
Random Forest     SVM   7.9999    0.0   6.0191  9.9808   True
Random Forest XGBoost  -0.4396 0.8269  -2.4204  1.5413  False
          SVM XGBoost  -8.4395    0.0 -10.4204 -6.4586   True
------------------------------------------------------------
Melhor modelo para Log Loss com base no teste de hipóteses: XGBoost

Métrica: Dice Coefficient
--------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0000)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
------------------------------------------------------------
Random Forest     SVM  -0.1866    0.0 -0.2418 -0.1313   True
Random Forest XGBoost   0.0118 0.8382 -0.0434  0.0671  False
          SVM XGBoost   0.1984    0.0  0.1431  0.2536   True
------------------------------------------------------------
Melhor modelo para Dice Coefficient com base no teste de hipóteses: XGBoost

Resumo dos melhores modelos para o método de extração 't-test':
Accuracy: XGBoost
Log Loss: XGBoost
Dice Coefficient: XGBoost

Analisando método de extração de features: Random Forest
==================================================

Métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0000)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
------------------------------------------------------------
Random Forest     SVM  -0.2561    0.0 -0.3113 -0.2009   True
```

```
Random Forest XGBoost   0.0195 0.6249 -0.0357  0.0747  False
          SVM XGBoost   0.2756    0.0  0.2204  0.3308   True
-------------------------------------------------------------
Melhor modelo para Accuracy com base no teste de hipóteses: XGBoost

Métrica: Log Loss
-------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0000)
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
    group1     group2 meandiff p-adj   lower    upper  reject
-------------------------------------------------------------
Random Forest     SVM   9.2307    0.0   7.2406 11.2208   True
Random Forest XGBoost  -0.7033 0.6249  -2.6934  1.2868  False
          SVM XGBoost   -9.934    0.0 -11.9241 -7.9439   True
-------------------------------------------------------------
Melhor modelo para Log Loss com base no teste de hipóteses: XGBoost

Métrica: Dice Coefficient
-------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0000)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
    group1     group2 meandiff p-adj   lower  upper  reject
-------------------------------------------------------------
Random Forest     SVM  -0.1985    0.0  -0.252 -0.145   True
Random Forest XGBoost   0.0208 0.5686 -0.0327 0.0743  False
          SVM XGBoost   0.2193    0.0  0.1658 0.2728   True
-------------------------------------------------------------
Melhor modelo para Dice Coefficient com base no teste de hipóteses: XGBoost

Resumo dos melhores modelos para o método de extração 'Random Forest':
Accuracy: XGBoost
Log Loss: XGBoost
Dice Coefficient: XGBoost

Melhores modelos por método de seleção: None
```

Análise e Discussão : comparação de modelos por método de seleção (2D)

Os resultados das análises indicam uma clara preferência pelo modelo XGBoost na maioria dos métodos de extração de features avaliados. No caso do método de extração Sem Seleção, o XGBoost destacou-se significativamente em todas as métricas, incluindo Accuracy, Log Loss e Dice Coefficient. Este desempenho foi superior ao dos outros modelos, como Random Forest e SVM, demonstrando que o XGBoost é mais robusto e preciso quando não é realizada qualquer seleção de features.

Quando analisado o método de extração PCA, no entanto, não houve diferença significativa entre os modelos. Em termos de Accuracy, Log Loss e Dice Coefficient, nenhum dos algoritmos – seja Random Forest, SVM ou XGBoost – apresentou uma vantagem estatisticamente relevante. Esse resultado sugere que o PCA, ao reduzir a dimensionalidade, pode nivelar o desempenho entre os modelos, tornando-os igualmente eficazes ou ineficazes, sem uma distinção clara entre eles.

Para o método de extração t-test, o XGBoost novamente mostrou-se superior. A análise revelou diferenças significativas em todas as métricas, e o XGBoost apresentou consistentemente o melhor desempenho em Accuracy, Log Loss e Dice Coefficient, [illegible] e, em alguns casos, o Random Forest. Esse resultado indica que o t-test,

como técnica de seleção de features, beneficia mais o XGBoost do que os outros algoritmos, permitindo que capture padrões relevantes no conjunto de dados de maneira mais eficaz.

Por fim, no método de extração Random Forest, o XGBoost manteve-se novamente como o modelo com o melhor desempenho, destacando-se em todas os critérios de avaliação. Tanto na Accuracy quanto na Log Loss e no Dice Coefficient, o XGBoost obteve resultados superiores aos do SVM e Random Forest, o que confirma a sua capacidade de generalização e precisão mesmo quando as features são extraídas com base em um modelo próprio de árvores.

Em síntese, o modelo XGBoost mostra-se como a melhor escolha em praticamente todos os cenários testados, com exceção do método PCA, onde os modelos apresentaram desempenhos similares. Esses resultados sugerem que o XGBoost é particularmente eficaz em capturar padrões complexos nos dados, adaptando-se bem a diferentes técnicas de seleção de features e entregando um desempenho consistentemente superior nas métricas de avaliação mais críticas, como Accuracy, Log Loss e Dice Coefficient.

Agora, vamos ver com as features 3d:

In [109…

```
melhores_modelos = avaliar_modelos_por_metodo(th_svm_3d, th_rf_3d, th_xgb_3d)
print("\nMelhores modelos por método de seleção:", melhores_modelos)
```

```
Analisando método de extração de features: Sem Seleção
==================================================

Métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0000)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower  upper  reject
------------------------------------------------------------
Random Forest     SVM  -0.4195    0.0  -0.481 -0.358   True
Random Forest XGBoost   0.0317 0.3836 -0.0298 0.0932  False
          SVM XGBoost   0.4512    0.0  0.3897 0.5127   True
------------------------------------------------------------
Melhor modelo para Accuracy com base no teste de hipóteses: XGBoost

Métrica: Log Loss
--------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0000)
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower    upper   reject
------------------------------------------------------------
Random Forest     SVM  15.1208    0.0   12.904  17.3375   True
Random Forest XGBoost  -1.1428 0.3836  -3.3596   1.0739  False
          SVM XGBoost -16.2636    0.0 -18.4803 -14.0469   True
------------------------------------------------------------
Melhor modelo para Log Loss com base no teste de hipóteses: XGBoost

Métrica: Dice Coefficient
--------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0022)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
------------------------------------------------------------
Random Forest     SVM  -0.3941 0.0062 -0.6684 -0.1198   True
Random Forest XGBoost   0.0318 0.9488 -0.2425  0.3061  False
          SVM XGBoost   0.4259 0.0036  0.1516  0.7002   True
------------------------------------------------------------
Melhor modelo para Dice Coefficient com base no teste de hipóteses: XGBoost

Resumo dos melhores modelos para o método de extração 'Sem Seleção':
Accuracy: XGBoost
Log Loss: XGBoost
Dice Coefficient: XGBoost

Analisando método de extração de features: PCA
==================================================

Métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0066)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
------------------------------------------------------------
Random Forest     SVM   0.0512 0.0148  0.0104   0.092   True
Random Forest XGBoost  -0.0024 0.9861 -0.0432  0.0384  False
          SVM XGBoost  -0.0537 0.0111 -0.0945 -0.0128   True
------------------------------------------------------------
Melhor modelo para Accuracy com base no teste de hipóteses: Random Forest
```



```
Métrica: Log Loss
```

```
--------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0066)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
-------------------------------------------------------------
Random Forest     SVM  -1.8461 0.0148 -3.3171 -0.3752   True
Random Forest XGBoost   0.0879 0.9861  -1.383  1.5588  False
          SVM XGBoost    1.934 0.0111  0.4631   3.405   True
-------------------------------------------------------------
Melhor modelo para Log Loss com base no teste de hipóteses: Random Forest

Métrica: Dice Coefficient
--------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0121)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
-------------------------------------------------------------
Random Forest     SVM   0.0537 0.0235  0.0074  0.1001   True
Random Forest XGBoost  -0.0013 0.9969 -0.0477  0.0451  False
          SVM XGBoost   -0.055 0.0206 -0.1014 -0.0087   True
-------------------------------------------------------------
Melhor modelo para Dice Coefficient com base no teste de hipóteses: Random Forest

Resumo dos melhores modelos para o método de extração 'PCA':
Accuracy: Random Forest
Log Loss: Random Forest
Dice Coefficient: Random Forest

Analisando método de extração de features: t-test
==================================================

Métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0000)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
-------------------------------------------------------------
Random Forest     SVM  -0.4293    0.0 -0.4844 -0.3742   True
Random Forest XGBoost   0.0561 0.0459   0.001  0.1112   True
          SVM XGBoost   0.4854    0.0  0.4303  0.5405   True
-------------------------------------------------------------
Melhor modelo para Accuracy com base no teste de hipóteses: XGBoost

Métrica: Log Loss
--------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0000)
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
=============================================================
    group1     group2 meandiff p-adj   lower    upper   reject
-------------------------------------------------------------
Random Forest     SVM  15.4724    0.0  13.4869  17.4579   True
Random Forest XGBoost   -2.022 0.0459  -4.0074  -0.0365   True
          SVM XGBoost -17.4944    0.0 -19.4798 -15.5089   True
-------------------------------------------------------------
Melhor modelo para Log Loss com base no teste de hipóteses: XGBoost

Métrica: Dice Coefficient
--------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0014)
mparison of Means - Tukey HSD, FWER=0.05
=============================================================
```

```
    group1     group2 meandiff p-adj   lower   upper  reject
-------------------------------------------------------------
Random Forest     SVM   -0.404 0.0052 -0.6776 -0.1303   True
Random Forest XGBoost   0.0569 0.8461 -0.2168  0.3306  False
          SVM XGBoost   0.4609  0.002  0.1872  0.7345   True
-------------------------------------------------------------
Melhor modelo para Dice Coefficient com base no teste de hipóteses: XGBoost

Resumo dos melhores modelos para o método de extração 't-test':
Accuracy: XGBoost
Log Loss: XGBoost
Dice Coefficient: XGBoost

Analisando método de extração de features: Random Forest
==================================================

Métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0000)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj  lower   upper  reject
------------------------------------------------------------
Random Forest     SVM  -0.4878   0.0 -0.5298 -0.4458   True
Random Forest XGBoost      0.0   1.0  -0.042   0.042  False
          SVM XGBoost   0.4878   0.0  0.4458  0.5298   True
------------------------------------------------------------
Melhor modelo para Accuracy com base no teste de hipóteses: Random Forest

Métrica: Log Loss
--------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0000)
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj  lower    upper   reject
------------------------------------------------------------
Random Forest     SVM  17.5823   0.0  16.0683  19.0962   True
Random Forest XGBoost      0.0   1.0  -1.5139   1.5139  False
          SVM XGBoost -17.5823   0.0 -19.0962 -16.0683   True
------------------------------------------------------------
Melhor modelo para Log Loss com base no teste de hipóteses: Random Forest

Métrica: Dice Coefficient
--------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0008)
    Multiple Comparison of Means - Tukey HSD, FWER=0.05
============================================================
    group1     group2 meandiff p-adj   lower   upper  reject
------------------------------------------------------------
Random Forest     SVM  -0.4621 0.0018  -0.733 -0.1911   True
Random Forest XGBoost      0.0    1.0 -0.2709  0.2709  False
          SVM XGBoost   0.4621 0.0018  0.1911   0.733   True
------------------------------------------------------------
Melhor modelo para Dice Coefficient com base no teste de hipóteses: Random Forest

Resumo dos melhores modelos para o método de extração 'Random Forest':
Accuracy: Random Forest
Log Loss: Random Forest
Dice Coefficient: Random Forest

Melhores modelos por método de seleção: None
```

Análise Discussão : comparação de modelos por método de seleção (3D)

A análise dos métodos de extração de features sem seleção, PCA, t-test e Random Forest revela diferenças no desempenho dos modelos Random Forest, SVM e XGBoost em cada métrica.

Para o método de Sem Seleção, o XGBoost apresentou o melhor desempenho em todas as métricas (Accuracy, Log Loss e Dice Coefficient), superando os outros modelos. Já no método de PCA, o modelo Random Forest obteve o melhor desempenho em todas as métricas, demonstrando vantagem em Accuracy, Log Loss e Dice Coefficient.

Ao utilizar o método de t-test, o XGBoost novamente destacou-se em todas as métricas, mostrando sua eficácia quando as features são selecionadas pelo t-test. No método de extração Random Forest, o próprio modelo Random Forest foi o melhor em Accuracy, Log Loss e Dice Coefficient, sugerindo um alinhamento natural com a técnica de extração.

Em resumo, os resultados mostram que XGBoost é o melhor modelo para os métodos Sem Seleção e t-test, enquanto o Random Forest destaca-se com os métodos de extração PCA e Random Forest. Não há um único modelo que se sobressaia em todos os métodos, indicando que o desempenho ideal depende da combinação específica de modelo e método de extração de features.

Comparação entre diferentes métodos de seleção de features para um mesmo modelo de classificação

Este código foi desenvolvido para avaliar diferentes métodos de seleção de features para um modelo de machine learning específico (por exemplo, SVM, Random Forest, ou XGBoost) com base em três métricas de desempenho: accuracy, log loss e dice coefficient. Ele compara quatro métodos de seleção de features (Sem Seleção, PCA, t-test e Random Forest) e utiliza testes de hipóteses para identificar se há diferenças estatisticamente significativas no desempenho desses métodos.

O código visa ajudar na escolha do melhor método de seleção de features para um modelo específico ao analisar qual método se destaca em cada métrica de avaliação. Isso é feito ao comparar as médias de desempenho dos métodos em várias métricas para avaliar se as diferenças observadas são estatisticamente significativas. Depois é identificado o método com o desempenho significativamente melhor para cada métrica, proporcionando uma escolha fundamentada para o pré-processamento de dados no contexto do modelo em uso.

Uso dos Testes de Hipóteses: Para assegurar que as diferenças de desempenho entre os métodos de seleção de features não são apenas fruto de variação aleatória, o código utiliza uma sequência de testes de hipóteses:

Teste ANOVA:

O teste ANOVA é utilizado inicialmente para avaliar se há uma diferença significativa entre as médias de desempenho dos métodos de seleção de features para cada métrica. Ele testa a hipótese nula de que as médias dos grupos (métodos de seleção) são iguais, contra a va de que pelo menos um dos grupos tem uma média diferente. Se o p-

valor do ANOVA é menor que 0,05, isso indica que existe uma diferença significativa entre pelo menos dois métodos, permitindo que se prossiga com uma análise mais detalhada.

Teste post hoc de Tukey (Comparações Múltiplas):

Se o ANOVA indicar significância, o teste de Tukey é aplicado como uma análise subsequente para identificar que pares de métodos de seleção de features apresentam diferenças significativas. O teste de Tukey fornece comparações emparelhadas entre todos os métodos, identificando que métodos específicos têm desempenhos significativamente melhores que os outros. Este teste permite não apenas identificar a existência de uma diferença significativa, mas também determinar que métodos são significativamente superiores.

Critério para Seleção do Melhor Método:

- **Para log loss:** O menor valor indica melhor desempenho, então o método de seleção de features com a menor média e com diferenças significativas em relação aos outros é escolhido.
- **Para accuracy e dice coefficient:** O maior valor indica melhor desempenho, então o método de seleção de features com a maior média e com diferenças significativas é selecionado.

Este critério garante que o método escolhido para cada métrica é significativamente melhor do que os outros métodos, e não apenas aparentemente superior devido a flutuações nos dados.

In [110…

```
def avaliar_metodos(th_svm, name):
    # Nome das métricas e métodos para análise
    metricas = ["Accuracy", "Log Loss", "Dice Coefficient"]
    metodos = ["Sem Seleção", "PCA", "t-test", "Random Forest"]
    melhores_metodos = {}

    print(f"\n --------- Modelo selecionado: {name} --------- \n")

    # Iterar sobre cada métrica (accuracy, log loss, dice coefficient)
    for i, metrica in enumerate(metricas):
        print(f"\nAnalisando métrica: {metrica}\n" + "-"*50)

        # Extrair os dados para cada métrica específica
        dados_metrica = [metodo[i] for metodo in th_svm]

        # Calcular as médias para cada método para a métrica em questão
        medias = [np.mean(valores) for valores in dados_metrica]

        # Executar o teste ANOVA de uma via para comparar as médias entre os métodc
        f_stat, p_val = stats.f_oneway(*dados_metrica)

        if p_val < 0.05:
            print(f"Diferença significativa encontrada na métrica {metrica} (p = {p

            # Preparar dados para o teste de Tukey
            dados_flat = np.concatenate(dados_metrica)
            grupos = np.array([metodo for metodo in metodos for _ in range(5)])  #

            # Executar o teste post hoc de Tukey para identificar quais métodos dif
            tukey_result = pairwise_tukeyhsd(dados_flat, grupos, alpha=0.05)
            print(tukey_result)
```

```
            # Identificar métodos que são significativamente melhores com base no t
            significativos = {metodos[j]: medias[j] for j in range(len(medias))
                              if all(tukey_result.reject[idx]
                                     for idx, comparacao in enumerate(tukey_result.
                                     if metodos[j] in comparacao[:2])}

            # Selecionar o melhor método entre os que têm diferença significativa
            if metrica == "Log Loss":
                # Para log loss, menor média é melhor
                melhor_metodo = min(significativos, key=significativos.get, default
            else:
                # Para accuracy e dice coefficient, maior média é melhor
                melhor_metodo = max(significativos, key=significativos.get, default

            melhores_metodos[metrica] = melhor_metodo
            print(f"Melhor método para {metrica} com base no teste de hipóteses: {m

        else:
            print(f"Não há diferença significativa entre os métodos para a métrica
            melhores_metodos[metrica] = "Nenhum método se destacou"

    # Exibir o resumo dos melhores métodos
    print("\nResumo dos melhores métodos:")
    for metrica, metodo in melhores_metodos.items():
        print(f"{metrica}: {metodo}")
```

Vamos ver se existem então diferenças significaticas entre os métodos de seleção:

In [111…

```
melhor_metodo_svm = avaliar_metodos(th_svm_2d, "SVM")
melhor_metodo_rf = avaliar_metodos(th_rf_2d, "Random Forest")
melhor_metodo_svm = avaliar_metodos(th_xgb_2d, "XGBoost")
```

```
 --------- Modelo selecionado: SVM ---------


Analisando métrica: Accuracy
------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0000)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
--------------------------------------------------------------------
          PCA Random Forest  -0.2415    0.0 -0.3189 -0.1641   True
          PCA   Sem Seleção  -0.2415    0.0 -0.3189 -0.1641   True
          PCA        t-test  -0.2024    0.0 -0.2798  -0.125   True
Random Forest   Sem Seleção      0.0    1.0 -0.0774  0.0774  False
Random Forest        t-test    0.039 0.4926 -0.0384  0.1164  False
  Sem Seleção        t-test    0.039 0.4926 -0.0384  0.1164  False
--------------------------------------------------------------------
Melhor método para Accuracy com base no teste de hipóteses: PCA

Analisando métrica: Log Loss
------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0000)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
--------------------------------------------------------------------
          PCA Random Forest   8.7032    0.0  5.9138 11.4927   True
          PCA   Sem Seleção   8.7032    0.0  5.9138 11.4927   True
          PCA        t-test   7.2966    0.0  4.5072 10.0861   True
Random Forest   Sem Seleção      0.0    1.0 -2.7894  2.7894  False
Random Forest        t-test  -1.4066 0.4926  -4.196  1.3829  False
  Sem Seleção        t-test  -1.4066 0.4926  -4.196  1.3829  False
--------------------------------------------------------------------
Melhor método para Log Loss com base no teste de hipóteses: PCA

Analisando métrica: Dice Coefficient
------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0000)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
--------------------------------------------------------------------
          PCA Random Forest  -0.1836    0.0 -0.2611 -0.1062   True
          PCA   Sem Seleção  -0.1836    0.0 -0.2611 -0.1062   True
          PCA        t-test  -0.1661 0.0001 -0.2436 -0.0887   True
Random Forest   Sem Seleção      0.0    1.0 -0.0774  0.0774  False
Random Forest        t-test   0.0175 0.9152 -0.0599  0.0949  False
  Sem Seleção        t-test   0.0175 0.9152 -0.0599  0.0949  False
--------------------------------------------------------------------
Melhor método para Dice Coefficient com base no teste de hipóteses: PCA

Resumo dos melhores métodos:
Accuracy: PCA
Log Loss: PCA
Dice Coefficient: PCA

 --------- Modelo selecionado: Random Forest ---------


Analisando métrica: Accuracy
------------------------------------------------
Não há diferença significativa entre os métodos para a métrica Accuracy (p = 0.075
```

```
Analisando métrica: Log Loss
--------------------------------------------------
Não há diferença significativa entre os métodos para a métrica Log Loss (p = 0.075
7)

Analisando métrica: Dice Coefficient
--------------------------------------------------
Não há diferença significativa entre os métodos para a métrica Dice Coefficient (p
= 0.0942)

Resumo dos melhores métodos:
Accuracy: Nenhum método se destacou
Log Loss: Nenhum método se destacou
Dice Coefficient: Nenhum método se destacou

 --------- Modelo selecionado: XGBoost ---------


Analisando métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0001)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================================
    group1        group2    meandiff p-adj   lower  upper  reject
--------------------------------------------------------------------
          PCA Random Forest   0.0805 0.0004  0.0365 0.1245   True
          PCA   Sem Seleção   0.0854 0.0002  0.0414 0.1294   True
          PCA        t-test    0.078 0.0006  0.0341  0.122   True
Random Forest   Sem Seleção   0.0049 0.9885 -0.0391 0.0489  False
Random Forest        t-test  -0.0024 0.9985 -0.0464 0.0416  False
  Sem Seleção        t-test  -0.0073 0.9633 -0.0513 0.0367  False
--------------------------------------------------------------------
Melhor método para Accuracy com base no teste de hipóteses: PCA

Analisando métrica: Log Loss
--------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0001)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
--------------------------------------------------------------------
          PCA Random Forest  -2.9011 0.0004 -4.4868 -1.3153   True
          PCA   Sem Seleção  -3.0769 0.0002 -4.6626 -1.4911   True
          PCA        t-test  -2.8132 0.0006 -4.3989 -1.2274   True
Random Forest   Sem Seleção  -0.1758 0.9885 -1.7616  1.4099  False
Random Forest        t-test   0.0879 0.9985 -1.4978  1.6737  False
  Sem Seleção        t-test   0.2637 0.9633  -1.322  1.8495  False
--------------------------------------------------------------------
Melhor método para Log Loss com base no teste de hipóteses: PCA

Analisando métrica: Dice Coefficient
--------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0002)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================================
    group1        group2    meandiff p-adj   lower  upper  reject
--------------------------------------------------------------------
          PCA Random Forest   0.0815 0.0007  0.0345 0.1285   True
          PCA   Sem Seleção   0.0862 0.0004  0.0392 0.1332   True
          PCA        t-test   0.0781 0.0011  0.0311 0.1251   True
Random Forest   Sem Seleção   0.0046 0.9918 -0.0424 0.0517  False
Random Forest        t-test  -0.0034 0.9966 -0.0505 0.0436  False
Random Forest        t-test  -0.0081 0.9597 -0.0551 0.0389  False
--------------------------------------------------------------------
```

```
Melhor método para Dice Coefficient com base no teste de hipóteses: PCA

Resumo dos melhores métodos:
Accuracy: PCA
Log Loss: PCA
Dice Coefficient: PCA
```

Análise e Discussão: Comparação de métodos de seleção de features para cada modelo (2D)

A análise revela que, para o modelo SVM, o método de extração PCA é significativamente superior nas métricas Accuracy, Log Loss e Dice Coefficient, quando comparado aos métodos Random Forest, Sem Seleção e t-test. O uso do PCA melhora a precisão pela sua capacidade de simplificar e destacar features relevantes. Assim, o PCA é o método que induz um melhor desempenho do SVM.

Para o modelo Random Forest, não houve diferença significativa entre os métodos de extração para as métricas de Accuracy, Log Loss e Dice Coefficient. Isso indica que nenhum dos métodos de extração testados oferece uma vantagem significativa para melhorar o desempenho do Random Forest.

Para o modelo XGBoost, o método de extração PCA destacou-se como o melhor em todas as métricas: Accuracy, Log Loss e Dice Coefficient. O PCA apresentou uma diferença significativa positiva para Accuracy e Dice Coefficient e uma redução significativa no Log Loss em comparação com os métodos Random Forest, Sem Seleção e t-test. Esses resultados indicam que o PCA é o método de extração ideal para otimizar o desempenho do XGBoost.

Vamos ver agora as diferenças significativas nos modelos 3D:

In [112…

```
melhor_metodo_svm = avaliar_metodos(th_svm_3d, "SVM")
melhor_metodo_rf = avaliar_metodos(th_rf_3d, "Random Forest")
melhor_metodo_xgb = avaliar_metodos(th_xgb_3d, "XGBoost")
```

```
 --------- Modelo selecionado: SVM ---------


Analisando métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0000)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
--------------------------------------------------------------------
          PCA Random Forest  -0.4366    0.0 -0.5071  -0.366   True
          PCA   Sem Seleção  -0.4366    0.0 -0.5071  -0.366   True
          PCA        t-test  -0.4341    0.0 -0.5047 -0.3636   True
Random Forest   Sem Seleção      0.0    1.0 -0.0706  0.0706  False
Random Forest        t-test   0.0024 0.9996 -0.0681   0.073  False
  Sem Seleção        t-test   0.0024 0.9996 -0.0681   0.073  False
--------------------------------------------------------------------
Melhor método para Accuracy com base no teste de hipóteses: PCA

Analisando métrica: Log Loss
--------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0000)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
--------------------------------------------------------------------
          PCA Random Forest  15.7361    0.0 13.1928 18.2794   True
          PCA   Sem Seleção  15.7361    0.0 13.1928 18.2794   True
          PCA        t-test  15.6482    0.0 13.1049 18.1915   True
Random Forest   Sem Seleção      0.0    1.0 -2.5433  2.5433  False
Random Forest        t-test  -0.0879 0.9996 -2.6312  2.4554  False
  Sem Seleção        t-test  -0.0879 0.9996 -2.6312  2.4554  False
--------------------------------------------------------------------
Melhor método para Log Loss com base no teste de hipóteses: PCA

Analisando métrica: Dice Coefficient
--------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0360)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================================
    group1        group2    meandiff p-adj   lower  upper  reject
--------------------------------------------------------------------
          PCA Random Forest  -0.4113 0.0683 -0.8477  0.025  False
          PCA   Sem Seleção  -0.4113 0.0683 -0.8477  0.025  False
          PCA        t-test  -0.4101 0.0693 -0.8465 0.0262  False
Random Forest   Sem Seleção      0.0    1.0 -0.4364 0.4364  False
Random Forest        t-test   0.0012    1.0 -0.4352 0.4376  False
  Sem Seleção        t-test   0.0012    1.0 -0.4352 0.4376  False
--------------------------------------------------------------------
Melhor método para Dice Coefficient com base no teste de hipóteses: Nenhum se dest
acou

Resumo dos melhores métodos:
Accuracy: PCA
Log Loss: PCA
Dice Coefficient: Nenhum se destacou

 --------- Modelo selecionado: Random Forest ---------


Analisando métrica: Accuracy
--------------------------------------------------
                                ficativa encontrada na métrica Accuracy (p = 0.0004)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
```

```
=================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
-----------------------------------------------------------------
          PCA Random Forest   0.1024 0.0002  0.0502  0.1547   True
          PCA   Sem Seleção   0.0341 0.2787 -0.0181  0.0864  False
          PCA        t-test   0.0463 0.0914 -0.0059  0.0986  False
Random Forest   Sem Seleção  -0.0683 0.0087 -0.1205 -0.0161   True
Random Forest        t-test  -0.0561 0.0331 -0.1083 -0.0039   True
  Sem Seleção        t-test   0.0122 0.9075   -0.04  0.0644  False
-----------------------------------------------------------------
Melhor método para Accuracy com base no teste de hipóteses: Random Forest

Analisando métrica: Log Loss
--------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0004)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
-----------------------------------------------------------------
          PCA Random Forest  -3.6923 0.0002 -5.5745 -1.8101   True
          PCA   Sem Seleção  -1.2308 0.2787 -3.1129  0.6514  False
          PCA        t-test  -1.6703 0.0914 -3.5525  0.2119  False
Random Forest   Sem Seleção   2.4615 0.0087  0.5793  4.3437   True
Random Forest        t-test    2.022 0.0331  0.1398  3.9041   True
  Sem Seleção        t-test  -0.4396 0.9075 -2.3217  1.4426  False
-----------------------------------------------------------------
Melhor método para Log Loss com base no teste de hipóteses: Random Forest

Analisando métrica: Dice Coefficient
--------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0005)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
-----------------------------------------------------------------
          PCA Random Forest   0.1045 0.0003  0.0496  0.1593   True
          PCA   Sem Seleção   0.0365 0.2656 -0.0184  0.0913  False
          PCA        t-test   0.0476 0.1014 -0.0073  0.1024  False
Random Forest   Sem Seleção   -0.068 0.0129 -0.1228 -0.0131   True
Random Forest        t-test  -0.0569 0.0407 -0.1118  -0.002   True
  Sem Seleção        t-test   0.0111 0.9372 -0.0438  0.0659  False
-----------------------------------------------------------------
Melhor método para Dice Coefficient com base no teste de hipóteses: Random Forest

Resumo dos melhores métodos:
Accuracy: Random Forest
Log Loss: Random Forest
Dice Coefficient: Random Forest

 --------- Modelo selecionado: XGBoost ---------


Analisando métrica: Accuracy
--------------------------------------------------
Diferença significativa encontrada na métrica Accuracy (p = 0.0000)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
-----------------------------------------------------------------
          PCA Random Forest   0.1049    0.0  0.0712  0.1385   True
          PCA   Sem Seleção   0.0683 0.0001  0.0346  0.1019   True
          PCA        t-test   0.1049    0.0  0.0712  0.1385   True
Random Forest   Sem Seleção  -0.0366 0.0308 -0.0702 -0.0029   True
Random Forest        t-test      0.0    1.0 -0.0336  0.0336  False
```

```
  Sem Seleção        t-test   0.0366 0.0308  0.0029  0.0702   True
-----------------------------------------------------------------
Melhor método para Accuracy com base no teste de hipóteses: Sem Seleção

Analisando métrica: Log Loss
-------------------------------------------------
Diferença significativa encontrada na métrica Log Loss (p = 0.0000)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
-----------------------------------------------------------------
          PCA Random Forest  -3.7802    0.0  -4.993 -2.5674   True
          PCA   Sem Seleção  -2.4615 0.0001 -3.6743 -1.2488   True
          PCA        t-test  -3.7802    0.0  -4.993 -2.5674   True
Random Forest   Sem Seleção   1.3187 0.0308  0.1059  2.5314   True
Random Forest        t-test      0.0    1.0 -1.2128  1.2128  False
  Sem Seleção        t-test  -1.3187 0.0308 -2.5314 -0.1059   True
-----------------------------------------------------------------
Melhor método para Log Loss com base no teste de hipóteses: Sem Seleção

Analisando métrica: Dice Coefficient
-------------------------------------------------
Diferença significativa encontrada na métrica Dice Coefficient (p = 0.0000)
       Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================================
    group1        group2    meandiff p-adj   lower   upper  reject
-----------------------------------------------------------------
          PCA Random Forest   0.1058    0.0  0.0703  0.1413   True
          PCA   Sem Seleção   0.0696 0.0002  0.0341  0.1051   True
          PCA        t-test   0.1058    0.0  0.0703  0.1413   True
Random Forest   Sem Seleção  -0.0362 0.0451 -0.0717 -0.0007   True
Random Forest        t-test      0.0    1.0 -0.0355  0.0355  False
  Sem Seleção        t-test   0.0362 0.0451  0.0007  0.0717   True
-----------------------------------------------------------------
Melhor método para Dice Coefficient com base no teste de hipóteses: Sem Seleção

Resumo dos melhores métodos:
Accuracy: Sem Seleção
Log Loss: Sem Seleção
Dice Coefficient: Sem Seleção
```

Análise e Discussão: Comparação de métodos de seleção de features para cada modelo (3D)

A análise revela que, para o modelo SVM, o método PCA foi o melhor para as métricas de Accuracy e Log Loss, apresentando uma diferença significativa em relação aos outros métodos. No entanto, para o Dice Coefficient, nenhum método se destacou significativamente.

No caso do Random Forest, o método de extração Random Forest foi o mais eficaz para todas as métricas (Accuracy, Log Loss e Dice Coefficient), apresentando diferenças significativas quando comparado a todos os restantes métodos

Para o modelo XGBoost, o método Sem Seleção apresentou os melhores resultados em todas as métricas.

Comparação 2D VS 3D

Este código foi desenvolvido para comparar o desempenho de dois tipos de dados, 2D e 3D, em três modelos de machine learning (SVM, Random Forest e XGBoost) e em quatro métodos de seleção de features (Sem Seleção, PCA, t-test e Random Forest) usando três métricas: accuracy, log loss e dice coefficient. O objetivo é identificar se existe um tipo de dados que, de modo geral, apresenta desempenho superior em relação ao outro em uma série de combinações de modelo, método de seleção e métrica.

O objetivo do código é determinar, para cada combinação de modelo e método de seleção de features, se os dados 2D ou 3D possuem um desempenho significativamente superior, com base nas métricas de avaliação. No final, o código produz uma visão geral indicando que tipo de dados (2D ou 3D) se destaca na maioria dos casos e, assim, pode ser considerado o melhor de forma geral.

Uso dos Testes de Hipóteses Para comparar os tipos de dados 2D e 3D em cada métrica, modelo e método de seleção de features, o código utiliza o teste t para amostras emparelhadas. O uso dos testes de hipóteses tem como objetivo verificar se as diferenças observadas entre os desempenhos de 2D e 3D são estatisticamente significativas, evitando conclusões baseadas em variações aleatórias.

Teste t para amostras emparelhadas:

O teste t para amostras emparelhadas é aplicado para cada métrica, modelo e método de seleção de features, comparando os resultados dos dados 2D e 3D nos 5 folds disponíveis. O teste t é adequado aqui porque ele verifica se a diferença média entre os dois conjuntos de dados (2D e 3D) é significativamente diferente de zero, considerando que os mesmos folds foram aplicados para cada tipo de dados. Se o p-valor do teste for menor que 0,05, consideramos que existe uma diferença significativa entre os desempenhos dos dados 2D e 3D para a combinação específica de modelo, método de extração e métrica.

Critério para Definir o Melhor Desempenho:

- **Para log loss**, menor valor significa melhor desempenho, portanto, o tipo de dados com a média mais baixa é considerado melhor.
- **Para accuracy e dice coefficient**, maior valor significa melhor desempenho, então o tipo de dados com a média mais alta é preferido.

Ao longo de todas as comparações, o código acumula o número de "vitórias" para dados 2D e 3D. No final, o tipo de dados com o maior número de vitórias é considerado o melhor de forma geral, indicando qual tipo de dados (2D ou 3D) tende a apresentar desempenho superior em um contexto amplo.

In [113…

```
import numpy as np
import scipy.stats as stats

def avaliar_dados_2d_vs_3d(th_svm_2d, th_rf_2d, th_xgb_2d, th_svm_3d, th_rf_3d, th_
    # Nome das métricas, modelos e métodos para análise
    metricas = ["Accuracy", "Log Loss", "Dice Coefficient"]
    modelos = ["SVM", "Random Forest", "XGBoost"]
    metodos = ["Sem Seleção", "PCA", "t-test", "Random Forest"]

    # Dicionário para armazenar os resultados de cada tipo (2D e 3D) para cada modelo
    tipos_dados = {
```

```
        "Dados 2D": [th_svm_2d, th_rf_2d, th_xgb_2d],
        "Dados 3D": [th_svm_3d, th_rf_3d, th_xgb_3d]
    }

    # Contadores para o desempenho geral dos dados 2D e 3D
    vitorias_2d = 0
    vitorias_3d = 0

    # Iterar sobre cada modelo (SVM, Random Forest, XGBoost)
    for modelo_idx, modelo in enumerate(modelos):
        print(f"\nAnalisando o modelo: {modelo}\n" + "="*50)

        # Iterar sobre cada método de extração de features
        for metodo_idx, metodo in enumerate(metodos):
            print(f"\nMétodo de extração de features: {metodo}\n" + "="*50)

            # Iterar sobre cada métrica (accuracy, log loss, dice coefficient)
            for metrica_idx, metrica in enumerate(metricas):
                print(f"\nMétrica: {metrica}\n" + "-"*50)

                # Obter os dados 2D e 3D para a métrica, modelo e método atual
                dados_2d = tipos_dados["Dados 2D"][modelo_idx][metodo_idx][metrica_
                dados_3d = tipos_dados["Dados 3D"][modelo_idx][metodo_idx][metrica_

                # Calcular a média dos dados 2D e 3D
                media_2d = np.mean(dados_2d)
                media_3d = np.mean(dados_3d)

                # Executar o teste t pareado para comparar 2D e 3D
                t_stat, p_val = stats.ttest_rel(dados_2d, dados_3d)

                if p_val < 0.05:
                    # Determinar qual tipo de dado foi superior para a métrica atua
                    if metrica == "Log Loss":
                        # Para log loss, menor média é melhor
                        if media_2d < media_3d:
                            print(f"Dados 2D têm desempenho significativamente melh
                            vitorias_2d += 1
                        else:
                            print(f"Dados 3D têm desempenho significativamente melh
                            vitorias_3d += 1
                    else:
                        # Para accuracy e dice coefficient, maior média é melhor
                        if media_2d > media_3d:
                            print(f"Dados 2D têm desempenho significativamente melh
                            vitorias_2d += 1
                        else:
                            print(f"Dados 3D têm desempenho significativamente melh
                            vitorias_3d += 1
                else:
                    print(f"Não há diferença significativa entre dados 2D e 3D para

    # Comparação final para determinar qual tipo de dados foi melhor em geral
    print("\nComparação Geral entre Dados 2D e Dados 3D:")
    print(f"Vitórias para Dados 2D: {vitorias_2d}")
    print(f"Vitórias para Dados 3D: {vitorias_3d}")

    if vitorias_2d > vitorias_3d:
        print("Dados 2D apresentam, em geral, melhor desempenho.")
    elif vitorias_3d > vitorias_2d:
        print("Dados 3D apresentam, em geral, melhor desempenho.")
    else:
        print("Desempenho geral entre Dados 2D e Dados 3D é equivalente.")
```

In [114…

```
avaliar_dados_2d_vs_3d(th_svm_2d, th_rf_2d, th_xgb_2d,th_svm_3d, th_rf_3d, th_xgb_3
```

```
Analisando o modelo: SVM
==================================================

Método de extração de features: Sem Seleção
==================================================

Métrica: Accuracy
--------------------------------------------------
Dados 2D têm desempenho significativamente melhor em Accuracy (p = 0.0008)

Métrica: Log Loss
--------------------------------------------------
Dados 2D têm desempenho significativamente melhor em Log Loss (p = 0.0008)

Métrica: Dice Coefficient
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Dice Coefficient (p = 0.18
87)

Método de extração de features: PCA
==================================================

Métrica: Accuracy
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Accuracy (p = 0.3989)

Métrica: Log Loss
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Log Loss (p = 0.3989)

Métrica: Dice Coefficient
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Dice Coefficient (p = 0.39
47)

Método de extração de features: t-test
==================================================

Métrica: Accuracy
--------------------------------------------------
Dados 2D têm desempenho significativamente melhor em Accuracy (p = 0.0002)

Métrica: Log Loss
--------------------------------------------------
Dados 2D têm desempenho significativamente melhor em Log Loss (p = 0.0002)

Métrica: Dice Coefficient
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Dice Coefficient (p = 0.16
60)

Método de extração de features: Random Forest
==================================================

Métrica: Accuracy
--------------------------------------------------
Dados 2D têm desempenho significativamente melhor em Accuracy (p = 0.0008)

Métrica: Log Loss
--------------------------------------------------
Dados 2D têm desempenho significativamente melhor em Log Loss (p = 0.0008)

Métrica: Dice Coefficient
--------------------------------------------------
```

```
Não há diferença significativa entre dados 2D e 3D para Dice Coefficient (p = 0.18
87)

Analisando o modelo: Random Forest
==================================================

Método de extração de features: Sem Seleção
==================================================

Métrica: Accuracy
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Accuracy (p = 0.0161)

Métrica: Log Loss
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Log Loss (p = 0.0161)

Métrica: Dice Coefficient
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Dice Coefficient (p = 0.0125)

Método de extração de features: PCA
==================================================

Métrica: Accuracy
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Accuracy (p = 0.2980)

Métrica: Log Loss
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Log Loss (p = 0.2980)

Métrica: Dice Coefficient
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Dice Coefficient (p = 0.46
94)

Método de extração de features: t-test
==================================================

Métrica: Accuracy
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Accuracy (p = 0.6213)

Métrica: Log Loss
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Log Loss (p = 0.6213)

Métrica: Dice Coefficient
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Dice Coefficient (p = 0.57
44)

Método de extração de features: Random Forest
==================================================

Métrica: Accuracy
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Accuracy (p = 0.0240)

Métrica: Log Loss
--------------------------------------------------
sempenho significativamente melhor em Log Loss (p = 0.0240)
```

```
Métrica: Dice Coefficient
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Dice Coefficient (p = 0.0232)

Analisando o modelo: XGBoost
==================================================

Método de extração de features: Sem Seleção
==================================================

Métrica: Accuracy
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Accuracy (p = 0.7990)

Métrica: Log Loss
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Log Loss (p = 0.7990)

Métrica: Dice Coefficient
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Dice Coefficient (p = 0.83
88)

Método de extração de features: PCA
==================================================

Métrica: Accuracy
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Accuracy (p = 0.4263)

Métrica: Log Loss
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Log Loss (p = 0.4263)

Métrica: Dice Coefficient
--------------------------------------------------
Não há diferença significativa entre dados 2D e 3D para Dice Coefficient (p = 0.44
31)

Método de extração de features: t-test
==================================================

Métrica: Accuracy
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Accuracy (p = 0.0299)

Métrica: Log Loss
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Log Loss (p = 0.0299)

Métrica: Dice Coefficient
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Dice Coefficient (p = 0.0287)

Método de extração de features: Random Forest
==================================================

Métrica: Accuracy
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Accuracy (p = 0.0054)

Métrica: Log Loss
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Log Loss (p = 0.0054)
```

```
Métrica: Dice Coefficient
--------------------------------------------------
Dados 3D têm desempenho significativamente melhor em Dice Coefficient (p = 0.0062)

Comparação Geral entre Dados 2D e Dados 3D:
Vitórias para Dados 2D: 6
Vitórias para Dados 3D: 12
Dados 3D apresentam, em geral, melhor desempenho.
```

Análise e Discussão: comparação entre datasets obtidos a partir de imagens 2D vs 3D

A análise mostra que, em termos gerais, os dados 3D têm um desempenho superior ao dos dados 2D, com 12 vitórias contra 6.

Para o modelo SVM, os dados 2D destacam-se em Accuracy e Log Loss em alguns métodos de extração, como Sem Seleção e t-test, mas não houve diferença significativa para Dice Coefficient. No Random Forest, os dados 3D apresentaram desempenho significativamente melhor em todos os métodos onde houve diferença, especialmente em Accuracy, Log Loss e Dice Coefficient com as extrações Sem Seleção e Random Forest. O modelo XGBoost também mostrou melhores resultados com dados 3D em métodos como t-test e Random Forest, enquanto nos demais métodos não houve diferença significativa entre 2D e 3D.

Em resumo, embora os dados 2D sejam competitivos em alguns casos, os dados 3D se destacam como a melhor opção na maioria das configurações de modelo e extração de features, especialmente para Random Forest e XGBoost.

Conclusões finais

A análise geral evidencia que o desempenho dos modelos XGBoost, Random Forest e SVM é altamente dependente da combinação entre o tipo de dados (2D ou 3D) e o método de seleção de features. O XGBoost demonstrou ser o modelo mais versátil e robusto, destacando-se principalmente em dados 3D, onde apresentou os melhores resultados com os métodos Sem Seleção e t-test. Nos dados 2D, o XGBoost também teve um bom desempenho, embora as diferenças entre os modelos sejam menos acentuadas.

Para o Random Forest, os dados 3D foram especialmente benéficos, apresentando os melhores resultados. Nos dados 2D, nenhum método de seleção se destacou, o que revela que o Random Forest é menos sensível a métodos de seleção de features em baixa dimensionalidade.

O modelo SVM obteve melhores resultados com o método PCA nos dados 2D, reforçando que o PCA é eficaz na redução de dimensionalidade e melhoria de performance do SVM. Nos dados em 3D, o SVM mostrou-se menos influenciado pelo método de seleção.

Na comparação entre dados 2D e 3D, observa-se que os dados 3D potenciam melhores resultados em 12 das 18 comparações, especialmente para os modelos Random Forest e XGBoost. Assim, podemos concluir que os dados 3D são preferíveis para a maioria das abordagens, uma vez que permitem uma análise mais completa dos nódulos pulmonares a classificar. corrigi a conclusão do ficheiro acima

Referências

[voltar ao índice]

[LLZ18] Lu Liu, Yapeng Liu, and Hongyuan Zhao. Benign and malignant solitary pulmonary nod ules classification based on CNN and SVM. In ACM International Conference Proceeding Series, 2018.

[XZX+18] Yutong Xie, Jianpeng Zhang, Yong Xia, Michael Fulham, and Yanning Zhang. Fus- ing texture, shape and deep model-learned information at decision level for automated classification of lung nodules on chest CT. Information Fusion, 2018.

[Tor23] https://easychair.org/publications/preprint/x1bJT

[PCA] https://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch18.pdf

[CBMLN] https://www.researchgate.net/publication/355690036_Classification_of_Benign_and_Malignant_L

[PSO] https://ieeexplore.ieee.org/document/8940822

[DCLC] https://iopscience.iop.org/article/10.1088/1742-6596/2286/1/012011/pdf

[LG18] Bak SH Lee HY Lee G, Park H. Radiomics in Lung Cancer from Basic to Advanced: Current Status and Future Directions. Korean J Radiol, 2018.