Detecção de Câncer de Mama com Machine Learning e Algoritmos de Classificação

Murilo Holtz Foltran
Universidade Federal de São Paulo
São José dos Campos
murilo.holtz@unifesp.br

Resumo* — *Neste trabalho, foram implementados algoritmos de classificação e conceitos de Inteligência Artificial e Machine Learning para análise de dados e detecção de câncer de mama a partir de uma base de dados com 569 pacientes.

Palavras-chave* — *inteligência artificial, aprendizado de máquina, algoritmos de classificação, câncer de mama

Introdução

Em 2016, foi noticiado que a startup Israelence *Zebra Medical Vision*, fundada em 2014, havia começado a desenvolver um algoritmo que detectaria casos de câncer de mama, até mesmo casos não identificados por radiologistas [1]. Com isso, fica claro a importância da ciência de dados em pesquisas científicas e como elas podem auxiliar na saúde pública e na sociedade, como um todo. O câncer de mama, objeto de estudo neste projeto, é uma doença causada pela multiplicação desordenada de células anormais da mama. Com isso, é formado um tumor com potencial de invadir outros órgãos [2]. A partir de estudos com punção aspirativa por agulha fina (Fine-needle aspiration), foram compilados dados sobre câncer de mama adquiridos em análises de imagens digitalizadas. Os dados descrevem as características dos núcleos celulares presentes na imagem [3].

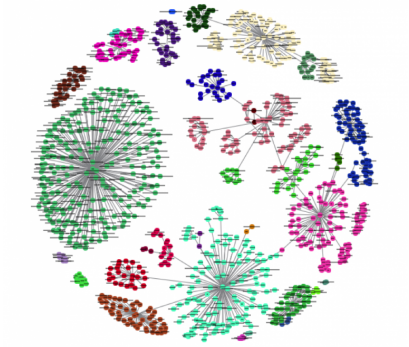

Figura 1. Representação do algoritmo de K-Nearest Neighbour [5].

Com base nisso, foram implementados algoritmos de classificação e predição, como regressão logística, K-nearest neighbour, árvore de decisão e floresta aleatória. Além disso, foi implementado o conceito de mapas auto-organizáveis. Esses algoritmos são capazes de, a partir de técnicas estatísticas, observar dados específicos de uma base e prenunciar valores tomados por uma variável categórica a partir de uma série de variáveis explicativas contínuas e/ou binárias [4]

Objetivo

Tendo em vista a importância da Inteligência Artificial e ciência de dados na medicina, tem-se como objetivo utilizar uma base de dados para detecção de câncer de mama em pacientes. A detecção é feita a partir de análises estatísticas com base em variáveis como *radius*, *texture*, *perimeter*, *area*, entre outras. Além de demonstrar a relevância de algoritmos de classificação e aprendizado de máquina para diversos problemas de saúde, servindo de apoio para diagnósticos e pesquisas científicas.

Materiais

A implementação do código foi feita utilizando a linguagem de programação Python 3.8.5. Com ela, foram importadas algumas bibliotecas específicas da linguagem para auxílio no carregamento dos dados, impressão de tabelas e gráficos. No decorrer do código, foram utilizados conceitos de listas e definição de funções. O ambiente de desenvolvimento escolhido foi o Google Colab, caderno *online* de programação que não requer configurações externas e é executado na nuvem [6].

A máquina utilizada para a implementação dos códigos foi um hardware com processador Intel Core i5, memória instalada de 8gb e um Sistema Operacional instalado do Windows 10 de 64 bits.

A. Importações

É inevitável que a linguagem utilizada neste trabalho possui diversas bibliotecas importantes e necessárias para a ciência de dados. No caso do atual projeto, foram importadas algumas bibliotecas para a implementação dos conceitos dados em aula. Primeiramente, importou-se a biblioteca *numpy*, interessante para a realização de cálculos matemáticos, *matplotlib*, que permite a impressão de gráficos e visualização dos dados a serem trabalhados, *pandas*, para visualização da base de dados em formato de tabelas, *seaborn*, biblioteca baseada no matplotlib, *minisom*, baseado em numpy para a criação de mapas auto-organizáveis (requer instalação), e, por fim, *sklearn*,

biblioteca importante para trabalhos com aprendizado de máquina

B. Base de dados

Como dito anteriormente, os dados utilizados no projeto descrevem as características dos núcleos celulares presentes na imagem digitalizada e um estudo de punção aspirativa por agulha fina, que caracteriza câncer de mama em pacientes como maligno ou benigno. A base de dados foi retirada do site Kaggle e contém informações como a identificação do paciente, o diagnóstico (M ou B), média da distância do centro aos pontos do perímetro, desvio padrão, perímetro, área, suavidade, compacidade, concavidade, pontos côncavos, simetria e dimensão fractal. Todos os dados atribuídos às variáveis contém 4 pontos flutuantes. No caso do estudo, temos como principal variável o diagnóstico do paciente, estabelecido na base de dados como 'M' para maligno e 'B' para benigno. Para a realização da análise, foram alterados para 1 e 0, respectivamente.

C. Métodos

Após a importação e instalação do conteúdo necessário para o desenvolvimento do projeto, iniciou-se gerando um menu, com google.colab > files, para que o usuário possa colocar o arquivo da base de dados em questão e carregá-lo na variável 'df', para *Data Frame*. Com a base de dados carregadas, foram criadas as variáveis 'populacao' e 'info' e atribuídos os valores do número de linhas e número de colunas, respectivamente. O arquivo .csv carregado possui 569 linhas e 33 colunas, portanto, pode-se ter noção do tamanho da base a ser trabalhada e que ela representa 569 pacientes. Após esse processo, foram retirados os valores vazios das colunas. Por definição, os valores são considerados vazios quando o conteúdo da variável é 'NaN', 'NAN', 'na' ou ''. Utilizando o método 'dropna', esse conteúdo foi removido da tabela para melhor utilização dos dados. Chamando as primeiras cinco linhas do *Data Frame*, temos uma tabela que permite a visualização da nomenclatura de cada representação das colunas, como o diagnóstico, raio, textura, perímetro, etc. Com a biblioteca seaborn, foi gerado o primeiro gráfico do código. Temos, então, no eixo Y a população e no eixo X o diagnóstico, com uma barra azul representando os casos malignos (212) e uma barra laranja representando os casos benignos (357).

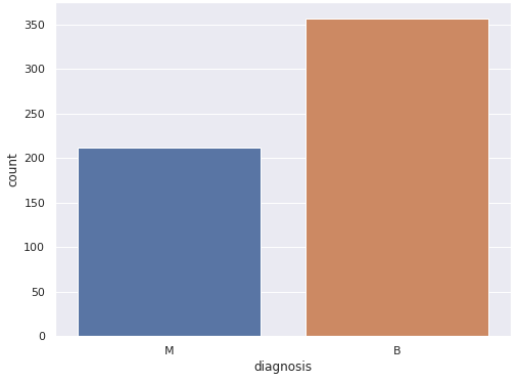

Figura 2. Gráfico população x diagnóstico.

Durante o código, foram chamadas algumas funções para esclarecer alguns conceitos do conteúdo, como o tipo das variáveis. Podemos ver, então, que a primeira coluna ('id') contém valores inteiros, a segunda coluna ('diagnosis') contém objetos e, no restante dos dados, temos valores flutuantes para cada variável.

É importante tratar a variável 'diagnosis' como um booleano, assim, temos maior facilidade para lidar com esses dados. Portanto, com o módulo LabelEncoder, importado anteriormente, podemos fazer essa transformação. Depois disso, com 'pairplot', da biblioteca *seaborn*, podemos visualizar alguns gráficos que demonstram esses valores de diagnósticos.

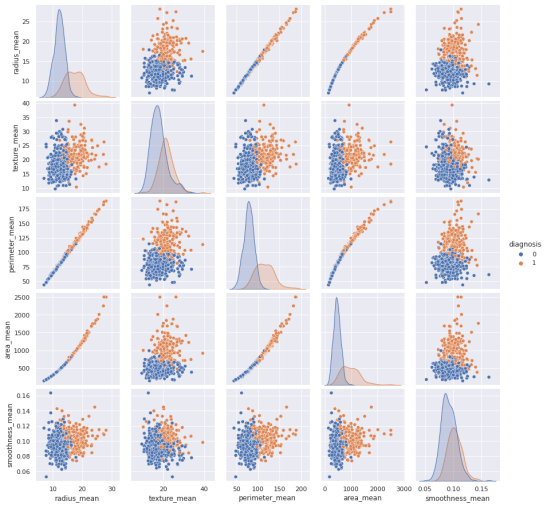

Figura 3. Gráfico gerados com dados da segunda à sexta coluna.

Para entender a correlação entre as variáveis das colunas, utilizou-se o método corr( ), da segunda à décima primeira coluna, que retorna uma tabela com o valor de cada correlação entre os dados. Com esses valores, é possível gerar um "mapa de calor", que mostra a magnitude dessas relações em duas dimensões. A princípio, um mapa de calor tem apenas as cores representando a magnitude de cada correlação. Porém, com a função heatmap( ) e utilizando algumas entradas como 'annot=True' e fmt='.0%', foi possível incluir os valores de cada espaço e transformá-los em porcentagens (Figura 4).

Dando fim ao processo de explorar e limpar os dados para o projeto, esses dados foram separados em dois tipos: independente (X) e dependente (Y). No caso dos dados independentes, temos todas as colunas a partir da segunda (de índice 1), para os dados dependentes, temos os valores da segunda coluna (diagnóstico).

```
X = df.iloc[:,2:31].values
Y = df.iloc[:,1].values
```

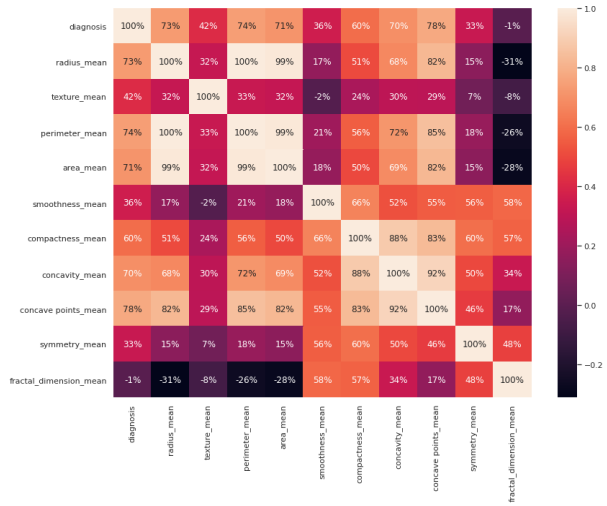

Figura 4. Mapa de calor (correlações).

Mapas auto-organizáveis

O primeiro conceito implementado com essas listas de valores foi o mapa auto-organizável. Os mapas auto-organizáveis são algoritmos de aprendizagem não-supervisionado e possuem forte inspiração na neurofísica. Com eles, tratamos os indivíduos de um estudo como neurônio e, entre eles, podemos ter diferentes pesos. Para a implementação, vamos utilizar a função 'MinMaxScaler' para normalizar os valores entre 0 e 1, isso fará com que teremos porcentagens para poder trabalhar. Depois disso, com os valores atribuídos à variável 'Xn', foi utilizada a função 'MiniSom' para gerar uma matriz que, posteriormente, foi transformada em um mapa auto-organizável. Um atributo interessante utilizado nesse processo foi o 'activation_response', com ele, podemos ver quantas vezes um neurônio foi selecionado como *best matching unit*, visto que em cada interação, um neurônio é considerado vencedor pois possui o maior campo local (aprendizagem competitiva) [6].

```
[ 8.,  6.,  6.,  1.,  4., 10.,  3.,  5.,  5.,  7.,  7.]
[ 4.,  6.,  6.,  0.,  1.,  7.,  9.,  6., 19.,  8.,  4.]
[ 6.,  3.,  1.,  7.,  4.,  2.,  5.,  3., 24.,  4.,  6.]
[ 3.,  7.,  0.,  2.,  1.,  2.,  1.,  9.,  5.,  2.,  0.]
[ 1.,  2.,  1.,  7.,  1.,  2., 10.,  2.,  0.,  2.,  4.]
[ 7.,  5.,  1.,  8., 11.,  3.,  3.,  6.,  3.,  2.,  0.]
[ 4.,  9.,  1.,  8., 15.,  3.,  8.,  3.,  3.,  3.,  7.]
[ 7.,  1.,  6.,  9.,  3., 12.,  4.,  2.,  3.,  1.,  3.]
[11.,  5.,  8., 11.,  4.,  4.,  2.,  8.,  3.,  0.,  6.]
[ 4.,  6.,  4.,  5.,  3.,  5.,  2.,  2.,  0.,  6.,  2.]
[ 3.,  3.,  2.,  3.,  2.,  5.,  3.,  4.,  4., 10.,  4.]
```

Matriz de activation_response.

Com as funções 'pcolor' e 'colorbar', criou-se o padrão visual para o mapa auto-organizável, assim, temos uma matriz 11 por 11 que recebe esses valores e os mostra a partir de um mapa de cores, conceito parecido com o mapa de calor, implementado anteriormente. A partir da atribuição de valores para vetores 'markers' e 'color', ao utilizá-los dentro de um laço for que percorre toda a variável 'Xn', temos um mapa completo. Então, para descrição, os círculos verdes representam os neurônios considerados benignos e os quadrados vermelhos os neurônios considerados malignos, nesse caso, referindo-se ao câncer de mama da base de dados.

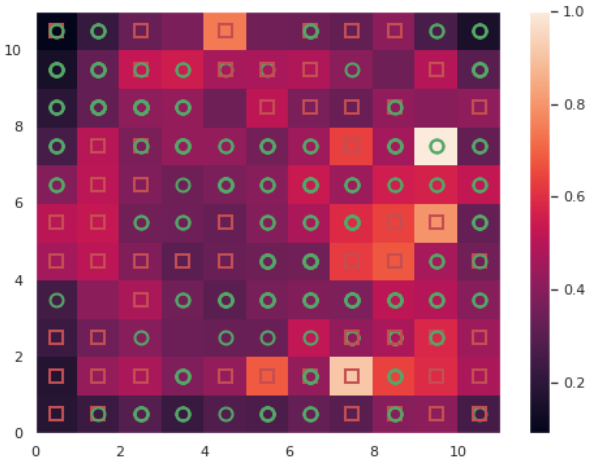

Figura 4. Mapa auto-organizável.

Algoritmos de classificação

Para a terceira parte do projeto, foram implementados algoritmos de classificação com aprendizado de máquina a fim de prever quais pacientes possuem caso de câncer maligno e quais possuem caso de câncer benigno. Os cálculos são feitos a partir das variáveis de cada coluna. Para início, a base de dados foi separada em duas partes: ¾ (75%) para treinamento e ¼ (25%) para teste. Com a função SandardScaler, foram normalizados os intervalos de variáveis independentes (ou seja, as variáveis que podem definir o tipo de câncer).

Com a base de dados atual, é possível utilizar diversos algoritmos de classificação para realizar essa detecção proposta no projeto, como o classificador Naive Bayes, SVC (Support Vector Machine), entre outros. A decisão tomada foi utilizar, nesse caso, a lógica de Regressão Logística, K-Nearest Neighbours, Árvore de Decisão e Floresta Aleatória.

O algoritmo de Regressão Logística é um classificador binário, como o SVM. A principal ideia dessa lógica é ajustar uma equação com múltiplas variáveis, a partir de uma combinação linear dos vários atributos que a base de dados estudada possui. No caso do K-Nearest Neighbours (ou K-vizinhos mais próximos), o algoritmo baseia-se na lógica de distância entre os pontos, assim, classificando os valores a partir de uma predição. Com esse algoritmo, é possível plotar gráficos de 3 dimensões, o que deixa a visualização de dados ainda mais interessante. Porém, no caso da base de dados do projeto, temos como classificações apenas dois valores (maligno e benigno) [8]. A Árvore de Decisão tem como referência a decisão e expressão de regras obtidas ao construir uma tabela, porém, sob a forma de árvore. As árvores são treinadas de acordo com um conjunto de treino. Paralelamente, a Floresta Aleatória pode

ser entendida como uma "coleção" de árvores de decisão, porém, as decisões são feitas de modo mais aleatório [9].

Com a teoria vista em aula e em fontes bibliográficas, foi utilizada a biblioteca *sklearn* para criar esses diferentes modelos de classificação. Para isso, foi criada uma função "algdeclass( )" com duas entradas: x e y, nela, construíram-se quatro variáveis: lr, knn, tree, forest. As quatro variáveis formam o retorno da função e possuem as informações de cada modelo. Também, dentro da função, estamos imprimindo a acurácia de cada modelo. Então, ao criar uma variável para receber o retorno dessa função, temos os testes realizados.

Com isso, pôde-se criar uma matriz de confusão. Essa matriz é utilizada em problemas de classificação binária, pois podemos tabelar os acertos e erros que o modelo apresenta.

		Valor Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Figura 5. Matriz de confusão [10].

Com essa tabela, além de podermos ver o número de casos verdadeiro, falso, falso negativo e verdadeiro negativo, podemos também calcular a acurácia do teste a partir dessa tabela com a seguinte fórmula:

(TP + TN) / (TP + TN + FP + TN)

Então, ao realizar esses cálculos dentro de um *laço for* que percorre todo o retorno da função "algdeclass( )" para os dados de treinamento, temos como output dessa execução:

```
Regressão Logística:
[[86  4]
 [ 3 50]]
Acurácia do teste:
0.951048951048951


KNN:
[[89  1]
 [ 6 47]]
Acurácia do teste:
0.951048951048951


Árvore de decisão:
[[83  7]
 [ 2 51]]
Acurácia do teste:
0.9370629370629371
```

```
Floresta Aleatória:
[[87  3]
 [ 2 51]]
Acurácia do teste:
0.965034965034965
```

Além disso, também é possível calcular precisão, recall e f1-score com esses valores a partir de outras fórmulas. Porém, optou-se por utilizar algumas funções como "classification_report", do *sklearn*, e "accuracy_score". também da mesma biblioteca, para verificar valores e comparar com o valor da precisão gerado na matriz de confusão.

Com as métricas do sklearn, dentro de um for percorrendo toda a variável que recebeu as saídas da função, temos outputs que mostram para o usuário o valor das variáveis precision, recall, f1-score e support. A precisão pode ser vista como uma medida de exatidão do classificador, recall é a medida da completude do classificador, f1 socre é uma média ponderada de precisão e recall, onde 1 é o melhor e 0 é o pior, e o support representa o número de ocorrências da classe em uma base de dados.

Resultados e discussão

Um dos principais objetivos do trabalho é mostrar a possibilidade de detectar qualquer tipo de diagnóstico em pacientes a partir de dados e valores que, ao analisar correlação e magnitude, pode-se aplicar diferentes tipos de algoritmos de classificação para permitir que essa detecção possa ser possível.

Depois de realizar todos os cálculos e métricas necessárias, chegou-se em resultados com relação a cada um dos modelos. Pela acurácia e precisão obtidas, pôde-se perceber que o modelo que obteve o melhor resultado foi a Floresta Aleatória, com uma acurácia de 96,5%.

Então, ao utilizar esse modelo para detectar células de câncer em pacientes, foi feita a predição e classificação nos dados de teste e mostrados, tanto os dados classificados, quanto os dados reais.

```
  Predição:       [1 0 0 0 0 0 0 0 0 0 1
0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1
0 1 0 1 0 1 0 1 0 0 1 0 0 1 0 0 0 1 1 1 1
0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0
1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0
0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0
1 1 0 0 0 1]

  Dados reais:    [1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1
0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1
0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0
0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0
1 1 0 0 0 1]
```

Ao analisar as listas com os valores, pode-se perceber que o modelo, apesar de ter a melhor acurácia, ainda diagnosticou cinco (5) pacientes de forma errônea. Então, no fim, foi criado uma figura com os valores das duas listas para visualizar as exatidões e imprecisões. Então, quando as barras apresentam diferença no centro da imagem, o diagnóstico não foi bem sucedido, isso se dá ao fato de que o algoritmo da Floresta Aleatória não possui uma acurácia de 100%.

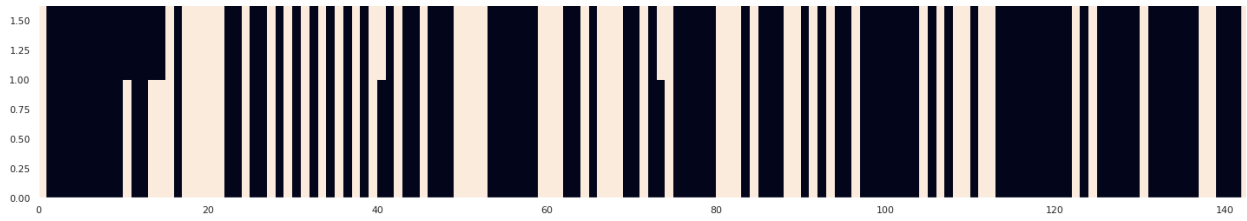

Figura 6. Representação visual dos resultados de predição comparados com dados reais.

Conclusão

Ao utilizar a base de dados de câncer de mama e analisá-la a partir de diferentes algoritmos de classificação, é possível perceber, na prática, que cada um deles possui sua vantagem e desvantagem, porém nenhum age perfeitamente. Esse é o caso de todo algoritmo de classificação, visto que sua lógica principal é prever/deduzir a classe de um certo indivíduo dentre diversos outros em uma base de dados.

Apesar disso, é possível, também, reparar na importância de um algoritmo na automação de um processo. No caso do algoritmo utilizado, temos dados que referenciam pacientes cujo diagnóstico é, ou não, câncer maligno. O tema envolto nesses dados é sério e de grande amplitude, portanto, deve-se tomar cuidado ao automatizar um processo de diagnóstico e sempre buscar, nesse caso, uma acurácia de 100%, pois estão sendo analisadas vidas humanas. Por outro lado, com as acurácias encontradas nos processos de treinamento ao decorrer do algoritmo, pode-se perceber uma alta porcentagem, e isso mostra o quão úteis esses algoritmos podem ser para automação de processos da saúde e/ou sociedade.

Referências

[1] Startup israelense diz ter ensinado algoritmo a detectar câncer de mama. <https://forumsaudedigital.com.br/startup-israelense-diz-ter-ensinado-um-algoritmo-para-detectar-cancer-de-mama/?utm_campaign=forum_saude_digital_-_181016&utm_medium=email&utm_source=RD+Station>. Acesso em: 28/07/2021.

[2] Tipos de câncer | INCA. <https://www.inca.gov.br/tipos-de-cancer/cancer-de-mama/>. Acesso em: 28/07/2021.

[3] Breast Cancer Wisconsin (Diagnostic) Data Set. <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data>. Acesso em: 28/07/2021.

[4] Regressão Logística. <https://pt.wikipedia.org/wiki/Regressão_logística>. Acesso em: 28/07/2021.

[5] K-Nearest Neighbours Algorithm | KNN Regression Python. <https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/>. Acesso em: 28/07/2021.

[6] Google Colab <https://kenzie.com.br/blog/google-colab/>. Acesso em: 28/07/2021.

[7] Mapas Auto-Organizáveis | UFPE. <https://www.cin.ufpe.br/~lfsc/cursos/introducaoainteligenciaartificial/IA-Aula11-MapasAutoOrganizaveis.pdf>. Acesso em: 01/08/2021.

[8] Algoritmos de Classificação: uma introdução. <https://awari.com.br/algoritmos-de-classificacao-uma-introducao/>. Acesso em: 31/07/2021.

[9] Aprendendo em uma Floresta Aleatória. <https://medium.com/machina-sapiens/o-algoritmo-da-floresta-aleatoria-3545f6babdf8> Acesso em: 31/07/2021.

[10] Performance de Machine Learning - Matriz de Confusão. <https://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/> Acesso em: 01/07/2021.