Um Estudo Aprofundado sobre a Predição da Recidiva de Câncer Usando Técnicas de Aprendizado de Máquina

André Eidi Maeda[1,2], Mariangela Dametto[1], Júlio César dos Reis[2], Rodrigo Bonacin[1]

amaeda@cti.gov.br, mdametto@cti.gov.br,jreis@ic.unicamp.br, rodrigo.bonacin@cti.gov.br

[1] Divisão de Metodologias da Computação - DIMEC, Centro de Tecnologia da Informação Renato Archer – CTI

[2] Instituto de Computação, Universidade Estadual de Campinas - Unicamp

Abstract. *Machine learning techniques are able to deal with data that we provide to computational systems, and in possession of this database it can analyze data in order to find patterns or trends about the chosen features. There are many areas of health in which this type of technique can be applied, with results of extreme social importance, such as identifying relationships that can determine the well-being of a person or even an entire population. Thus, health is a propitious domain for the development of machine learning techniques. In this sense, this scientific initiation article aims to report advances in research on the application of machine learning and deep learning algorithms in an open base with the objective of determining the chances of a patient having cancer relapse. In this way, it is expected to contribute to the best service of health professionals with more accurate diagnoses and medical decisions.*

Resumo. *Técnicas de aprendizado de máquina são capazes de lidar com dados que fornecemos para sistemas computacionais e em posse dessa base de dados pode analisá-los a fim de encontrar padrões ou tendências sobre os atributos escolhidos. Existem muitas áreas da saúde em que esse tipo de técnica pode ser aplicado com resultados de extrema importância social, tais como ao identificar relações que podem determinar o bem-estar de uma pessoa ou até mesmo uma população inteira. Sendo assim, a saúde é um domínio propício para a desenvolver as técnicas de aprendizado de máquina. Nesse sentido, este artigo de iniciação científica visa relatar os avanços nas pesquisas sobre a aplicação de algoritmos de aprendizado de máquina e aprendizado profundo em base aberta com o objetivo de determinar as chances de um paciente ter recidiva de câncer. Dessa maneira espera-se contribuir para o melhor serviço dos profissionais de saúde com diagnósticos e decisões médicas mais precisas.*

Palavras-chaves: Aprendizado de Máquina, Deep Learning, Predição, Inteligência Artificial, Informática em Saúde

[2] https://scikit-learn.org

[3] https://keras.io/

1. Introdução

A tecnologia está cada vez mais presente na vida das pessoas nos ajudando nos mais diferentes campos e áreas de atuação. Dentre os vários campos em que a tecnologia auxilia a vida dos humanos está a área da saúde, na qual sua atuação, extremamente abrangente, pode assistir nas mais distintas tarefas, tais como a predição de recidiva em casos de câncer, tema principal deste projeto. Isso é possível graças à aplicação de diferentes técnicas para análise de dados [Bishop 2006], tais como aprendizado de máquina e aprendizado profundo. Para tanto, foi utilizada base de dados aberta da Fundação Oncocentro de São Paulo (FOSP)[1]. Este banco de dados aberto nos fornece informações sobre mais de 1 milhão de pacientes dos quais cerca de 9% registram a recidiva, evidenciando a importância da questão abordada.

Neste trabalho, são apresentados os avanços com relação ao primeiro artigo publicado na jornada de iniciação científica de 2022 [Maeda et al. 2002]. Sendo assim, este artigo apresenta resultados de alguns algoritmos já explorados anteriormente, porém com alguns hiper parâmetros alterados, além algoritmos pouco explorados na publicação anterior, em adição, a aplicação de aprendizado profundo (*Deep Learning*). Por fim, foram realizados diferentes testes com outras *features*, além das usadas no primeiro trabalho.

Para implementação dessas técnicas usou-se como, na publicação anterior, a biblioteca para Python [Müller e Guido 2016], sci-kit learn[2], uma biblioteca open-source para apoiar o desenvolvimento de aplicações práticas de “Machine Learning”. Além disso, usou a biblioteca Keras (2023), uma biblioteca de código aberto criada para auxiliar aplicações de aprendizado profundo com Python.

O artigo está estruturado em seções da seguinte maneira: a seção 2 apresenta alguns dos conceitos não apresentados na primeira publicação, abordando seus aspectos teóricos, conceitos e técnicas. Já a seção 3 apresenta o detalhamento da metodologia aplicada no trabalho. Em seguida, a seção 4 apresenta os resultados obtidos com a aplicação das técnicas mencionadas. Finalmente, a seção 5 apresenta a conclusão e os próximos passos para a pesquisa.

Este projeto é coordenado e orientado pelo pesquisador Rodrigo Bonacin, no CTI Renato Archer, e conta com a participação de outros pesquisadores e bolsistas. Como trabalho de iniciação científica conta com o professor Júlio César dos Reis como orientador revisor do projeto.

2. Conceitos, Técnicas e Algoritmos

Nesta seção é apresentado brevemente os conceitos teóricos do projeto, tais como alguns dos algoritmos e bibliotecas utilizados. O aprendizado de máquina é um processo no qual um conjunto de dados é fornecido a algoritmos e com isso tentamos fazer inferências sobre os dados. Nesse sentido, espera-se que, com base nas informações fornecidas, seja possível fazer previsões ou encontrar padrões para obter respostas que podem ser difíceis de visualizar sem a devida atenção. No caso deste estudo, o objetivo é encontrar um modelo de aprendizado de máquina capaz de predizer casos de recorrência de câncer com base nas características dos pacientes, da doença dos tratamentos, entre outros parâmetros.

Este artigo é focado no uso de técnicas de aprendizagem de classificação supervisionada, que envolve o treinamento de um modelo, com base em informações de entrada e saída fornecidas anteriormente. Dessa forma, o modelo pode prever resultados futuros com base em parâmetros (características) derivados de resultados anteriores. Os algoritmos de classificação, que prevêem resultados classificando-os em variáveis discretas.

Para colocar essas técnicas em prática, a biblioteca Python scikit-learn e Keras foram usadas. Entre os vários algoritmos de classificação existentes, Árvore de Decisão, Random Tree, KNN, MLP e CNN para deep learning são estudados em profundidade neste artigo.

O algoritmo da árvore de decisão, também conhecido como *Decision Tree*, é um dos mais conhecidos algoritmos para aprendizado de máquina, usado tanto para classificação quanto para regressão. O algoritmo consiste em analisar os dados de forma que seja criado uma árvore na qual cada nó é uma decisão e as folhas são os resultados finais. O trabalho do algoritmo é encontrar quais serão os nós e condições alocados em cada posição. Para isso, o algoritmo faz uma série de cálculos, buscando o melhor arranjo de nós para obter os melhores resultados. Nesse sentido, damos atenção especial a dois atributos, a entropia e o ganho de informação. Essas duas variáveis dizem respeito a desorganização e falta de uniformidade nos dados, quanto mais entropia mais caóticos e misturados estão os dados, já quanto menor a entropia teremos uma base homogênea e menos distribuída.

Para definir os posicionamentos é calculada a entropia das classes de saída de cada nó e o ganho de informação dos atributos. Assim, a escolha da condição ou nó alocado em cada posição se dá pela informação do ganho de informação da feature. Sendo assim, escolhemos o atributo que melhor divide os dados nas classes de saída, de forma que cada ramificação seja praticamente homogênea.

O próximo algoritmo analisado neste relatório é o *random forest*. Assim como o próprio nome indica este classificador funciona de forma parecida com uma árvore de decisão. Conforme ilustra a Figura 1, o *random forest* cria várias árvores de decisão, originando uma floresta, sendo que cada uma delas gerará um resultado para a instância pedida. Sendo assim, no caso de problemas de classificação o valor modal é o que prevalece e é devolvido como resultado.

Um dos detalhes importantes que devemos nos atentar no algoritmo de *random forest* é que este utilizado do método *ensemble*. Nesse sentido, em geral, algoritmos criam apenas um modelo para resolver o problema e retornar o resultado. No entanto, algoritmos como a *random forest* criam vários modelos de algoritmos e decidem seu resultado com base em diversas respostas. Essa estratégia, apesar de ser mais custosa computacionalmente, costuma ter resultados mais satisfatórios. Ademais, diferente da árvore de decisão, o algoritmo de *random forest* para montar a primeira árvore seleciona aleatoriamente algumas amostras do conjunto de treino, não o conjunto inteiro. Com essas amostras é construída a árvore de decisão, porém a escolha da *feature* de decisão para cada nó não é feita com base em todas as disponíveis, mas sim em duas ou mais de forma aleatória e fazer os cálculos com base nas amostras selecionadas, nesta etapa é utilizado o *bootstrap*, que é um método de reamostragem onde as amostras selecionadas podem ser repetidas na seleção. Para os próximos nós repete-se esses passos. Assim como para cada árvore. Portanto, é possível que para uma árvore escolha features ruins para, por exemplo, o nó raiz. Entretanto, note que como são geradas várias árvores, isso não será um grande problema. Finalmente, o resultado que foi apresentado mais vezes será devolvido pelo algoritmo (ver Figura 1).

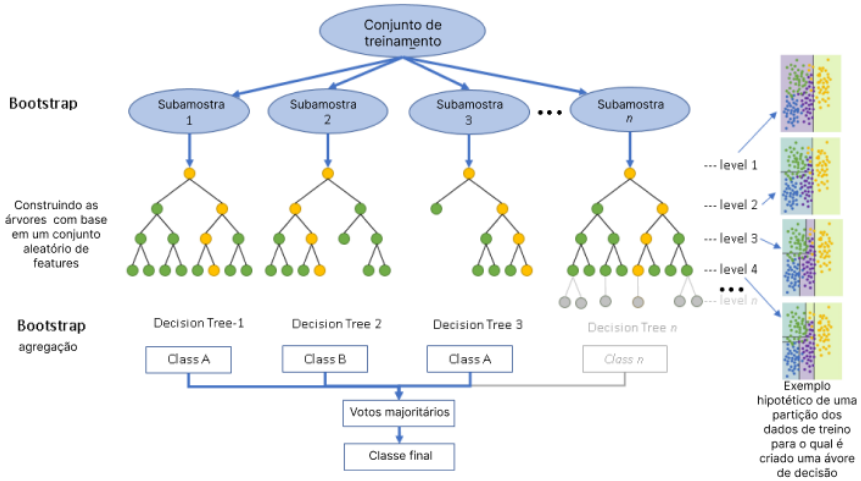

Figura 1. Na imagem observa-se uma representação do processo de decisão do algoritmo random forest. (adaptada: https://catalyst.earth/catalyst-system-files/help/concepts/focus_c/oa_classif_intro_rt.html)

Outro algoritmo utilizado foi o KNN (*K-Nearest-Neighbors*). De forma simples, o algoritmo KNN tenta classificar as amostras do conjunto avaliando a distância da instância analisada em relação aos vizinhos mais próximos já analisados. Assim, com base na majoritariedade de uma classe nestes vizinhos mais próximos o algoritmo escolhe qual classe alocar uma instância.

Para realizar essa decisão o algoritmo faz o cálculo das distâncias mais próximas das K instâncias mais próximas. Nesse sentido, como o próprio nome indica, é possível escolher o parâmetro K. Entretanto, note que quando K é pequeno, o algoritmo pode cair no problema de *overfitting*, já com K muito grande, a classificação pode ficar sujeita a *underfitting*. Portanto, é necessário testar diferentes combinações de parâmetros, uma boa análise, para escolher o valor de K adequado para o problema.

Por mim, são apresentados conceitos sobre redes neurais artificiais e aprendizado profundo. O aprendizado profundo é um dos ramos mais importantes do aprendizado de máquina. Nesse caso, os algoritmos possuem várias camadas de processamento dos dados por meio de redes neurais artificiais, que buscam simular o comportamento do cérebro humano. Nesse sentido, as redes neurais, de maneira simplificada, são pequenas unidades conectadas de forma organizada de modo que operações simples realizadas por cada neurônio levará a soluções de problemas complexos. É importante ressaltar que as redes neurais artificiais são utilizadas em aprendizado profundo, no entanto nem todo modelo de redes neurais artificiais é caracterizado como aprendizado profundo, por exemplo, ao utilizar algoritmos com apenas uma camada.

Para criar uma rede neural multicamada é necessário criar camadas intermediárias entre a entrada de dados e a saída desejada. Conforme ilustra a Figura 2, os dados de entrada serão passados para os da camada oculta, sendo que cada dado tem um peso específico a depender do neurônio que irá receber o dado. Portanto, na saída de cada neurônio o valor fornecido passou por uma série de transformações, como somas e multiplicações, além de um valor *offset* denominado *Bias*. Ao final teremos um resultado final que a depender da função escolhida de análise, os pesos e o bias do modelo o algoritmo retorna a decisão. Esse é o princípio do algoritmo MLP (*multilayer perceptron*), na qual toda camada, com exceção da entrada, possui uma relação direta com a camada anterior. Sendo assim, cada camada considera os valores anteriores para calcular a próxima camada até a saída.

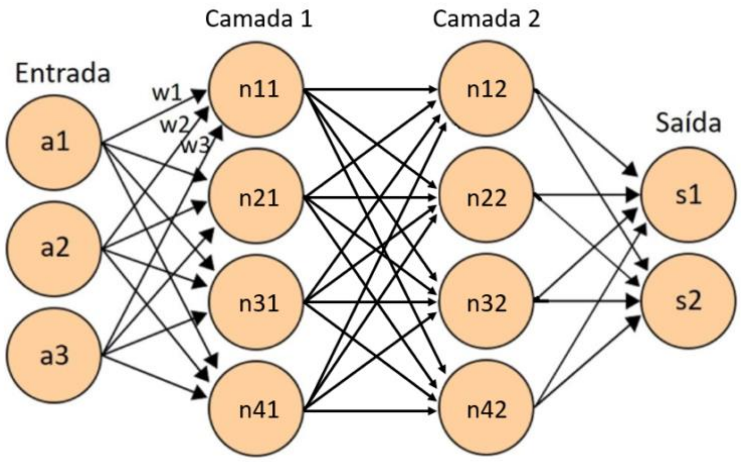

Figura 2. Na imagem observa-se uma representação de uma rede neural profunda. (fonte: https://didatica.tech/introducao-a-redes-neurais-e-deep-learning/)

Neste trabalho, além de aplicar o algoritmo de MLP foi aplicado o algoritmo CNN (Rede Neural Convolucional, em inglês, *Convolutional Neural Network*), que é um dos mais utilizados em aprendizado profundo. Tipicamente as redes neurais utilizadas no aprendizado profundo possuem diversas camadas e são submetidas a treinamento por grande volume de dados. A CNN é adequada para trabalhar com imagens, entretanto é possível usá-la em outros contextos como é o caso deste trabalho. Diferente da MLP, o algoritmo CNN analisa a camada anterior em pequenos conjuntos e produz uma saída para cada conjunto analisado. Então, por exemplo, se têm-se 12 valores de entrada, um neurônio de CNN pode, por exemplo, analisar 3 em 3 valores e gerar uma saída para cada conjunto. Por isso, essa técnica é muito aplicada para classificação de imagens visto que pode analisar trechos da imagem de forma separada.

3. Metodologia aplicada

Conforme supracitado, para desenvolver este projeto foi escolhido um método de classificação em aprendizagem supervisionada com o objetivo de fazer previsões sobre a recidiva de casos de câncer. As principais etapas desse processo são detalhadas nas subseções a seguir.

3.1 Seleção e Pré-processamento

Os dados sobre pacientes com câncer foram obtidos a partir do banco de dados aberto da FOSP (Fundação Oncocentro de São Paulo). A partir desses dados, foram selecionadas *features* as quais foram aplicados os algoritmos de aprendizado de máquina apresentados na seção 2. Nesse sentido, optamos por trabalhar com as *features* indicativas do tipo de tratamento, cirurgia, radioterapia, quimioterapia, etc. para o câncer de próstata. Além disso, ao verificar quais *features* causavam maior influência na predição de recidiva, buscou-se, também, fazer testes com as *features* EC (estadiamento clínico) e TRATCONS e META01.

A seleção de *features* considerou a qualidade, disponibilidade e relevância dos dados para o objetivo proposto (previsão de recorrência do câncer). Para tal, envolveu-se diretamente no projeto um biomédico especialista com mestrado em oncologia, havendo consultas pontuais deste especialista com outros especialistas (médicos especialistas). Vale ressaltar que o software de carregamento e pré-processamento deve ser desenvolvido antes de escolhê-lo. Os dados foram originalmente fornecidos como arquivos ".dbf". É uma extensão de banco de dados padrão que organiza informações em vários registros e armazena campos em forma de lista. Para processar esses arquivos, tivemos que usar a biblioteca 'simpledbf' para converter

os arquivos em tabelas (dataframes pandas) descrevendo os dados. Isso facilitou o trabalho com a base.

3.2. Treinamento do modelo

Após a seleção e pré-processamento dos dados, é necessário dividir a base para treinamento, validação e teste do modelo selecionado. Nesse sentido, foram utilizadas as seguintes estratégias descritas a seguir.

Primeiramente, a base de dados foi dividida em uma proporção de 80/20, ou seja, 80% dos dados são dedicados ao treinamento do modelo, os 20% restantes são aplicados ao teste do mesmo modelo. Em seguida, para tentar obter resultados mais confiáveis, foi aplicada uma técnica de validação cruzada chamada "K-fold". Essa técnica, disponível na biblioteca scikit-learning, envolve a divisão do conjunto de dados em 'K' subconjuntos do mesmo tamanho 'K'. Um desses subconjuntos é selecionado para teste e o restante para treinamento. Esse processo é feito 'K' vezes para que, ao final dessas 'K' interações, a precisão e o erro sejam calculados. Assim, resultados mais confiáveis são obtidos.

Por fim, assim como em Maeda et al. (2022) houve o problema de desbalanceamento das classes observadas, sendo assim há mais indicadores de não recidiva do que de recidiva. Esse desequilíbrio no banco de dados pode fazer com que o algoritmo aplicado retorne resultados errôneos ou enganosos, pois retornar a maioria dos dados resultará em dados altamente precisos, mas não realistas. Para isso, foi utilizada a técnica chamada SMOTE (*Synthetic Minority Sampling Technique*) [Chawla et al. 2002] usando a biblioteca *imblearn*. Essa técnica funciona para sobreamostrar a classe com a menor quantidade de dados, ou seja, são gerados exemplos agregados. Esses exemplos compostos são formados pela combinação das características de dois casos selecionados aleatoriamente 'a' e 'b'.

3.3. Aplicação dos algoritmos

As bibliotecas scikit-learning e Keras foram utilizadas na aplicação dos algoritmos selecionados. É importante especificar que em ambos os casos a maioria dos parâmetros foram persistidos com valores padrão, exceto por alguns algoritmos que foram realizados ajustes de hiperparâmetros a fim de tentar obter resultados melhores. Esses ajustes, são mencionados juntos com os respectivos resultados na próxima seção. Além disso, vale ressaltar que para este relatório a grande maioria dos algoritmos foram implementados de forma que as técnicas de classificação cruzada (K-fold) e smote foram aplicadas em conjunto.

4. Testes e Resultados

Após a aplicação do algoritmo, os resultados mostrados abaixo foram coletados, interpretados e discutidos quanto à sua validade. A seguir serão mostrados os resultados para os algoritmos. Os quatro primeiros algoritmos foram aplicados com auxílio da biblioteca scikit-learn. Os algoritmos CNN e ANN, respectivamente, foram aplicados com auxílio da biblioteca Keras. Para as Tabelas 1 a 9 a relação que se busca predizer foi entre os tratamentos e a recidiva. Já para as Tabelas 10 e 11 mostra-se os resultados da relação entre EC e recidiva, TRATCONS e recidiva e META01 e recidiva, respectivamente.

A Tabela 1 apresenta os resultados obtidos com a técnica árvore de decisão, incluindo as medidas de precisão, acurácia e F1-Score (média harmônica entre precisão e

revocação/cobertura). Os resultados foram obtidos usando 5 folds e a técnica de smote sobre os dados de treinamento. Quanto aos seus hiperparâmetros, foram colocados, criterion: “gini”, splitter: “best”, max_depth: ”None”, min_samples_split: “2”, min_samples_leaf: “1”, min_weight_fraction_leaf: “0.0”, max_features: “None”, random_state: “None”, max_leaf_nodes: “None”, min_impurity_decrease:”0.0”, class_weight: “None”, ccp_alpha: “”0.0. Sendo assim, a profundidade da árvore é aquela que melhor descreve os dados presentes na base visto que não foi imposta uma restrição de máxima profundidade.

Tabela 1.Resultados obtidos para cada estratégia de treinamento do modelo com Decision Tree

Decision Tree	***Resultados***
Acurácia	62.3
Precisão	59.5
F1 Score	66.6

A Tabela 2 apresenta os resultados obtidos com a técnica Random Forest, incluindo as medidas de precisão, acurácia e F1-Score (média harmônica entre precisão e revocação/cobertura). Os resultados foram obtidos usando validação cruzada (5 folds) e smote nos dados de treinamento. Finalmente, quanto aos seus hiperparâmetros, foram colocados, n_estimators: “100”, criterion: “gini”, max_depth: ”None”, min_samples_split: “2”, min_samples_leaf: “1”, min_weight_fraction_leaf: “0.0”, max_features: “sqrt”, max_leaf_nodes: “None”, min_impurity_decrease: “0.0”, bootstrap: “True”, oob_score: “False”, n_jobs: “None”, random_state: “None”, verbose: “0”, warm_start:”False”, class_weight: “None”, ccp_alpha: “0.0”, max_samples: “None”

Tabela 2.Resultados obtidos para cada estratégia de treinamento do modelo com Random Forest

Random Forest	***Resultados***
Acurácia	62.1
Precisão	59.3
F1 Score	66.5

A Tabela 3 apresenta os resultados obtidos com a técnica KNN, incluindo as medidas de precisão, acurácia e F1-Score (média harmônica entre precisão e revocação/cobertura). Os resultados foram obtidos usando validação cruzada (5 folds) e smote nos dados de treinamento.

Tabela 3.Resultados obtidos para cada estratégia de treinamento do modelo com KNN

KNN	***Resultados***
Acurácia	55.4

Precisão	54.4
F1 Score	62.7

As Tabelas 4 a 7 apresenta os resultados obtidos com a técnica MLP, incluindo as medidas de precisão, acurácia e F1-Score (média harmônica entre precisão e revocação/cobertura). Os resultados foram obtidos usando validação cruzada (5 folds) e smote nos dados de treinamento. Para essa primeira tentativa de utilizar redes neurais, fez-se testes com diferentes números máximos de iterações. Estas são: max_iter = 1500, 3000, 30000, 1000000. Quanto aos outros hiperparâmetros temos, hidden_layer_sizes: (11, 11), activation: 'relu' , solver: 'adam', alpha: 0.0001, batch_size: 'auto', learning_rate: 'constant', learning_rate_init: 0.001, power_t: 0.5, shuffle: True, random_state: None, tol: 0.00001, verbose: False, warm_start: False, momentum: 0.9, nesterovs_momentum: True, early_stopping: False, validation_fraction: 0.1, beta_1: 0.9, beta_2: 0.999, epsilon: 1e-08, n_iter_no_change: 10, max_fun: 15000

Tabela 4.Resultados obtidos para cada estratégia de treinamento do modelo com o classificador MLP

MLP classifier - 1500 Max_iter	***Resultados***
Acurácia	61.6
Precisão	59.0
F1 Score	66.0

Tabela 5.Resultados obtidos para cada estratégia de treinamento do modelo com o classificador MLP

MLP classifier - 3000 Max_iter	***Resultados***
Acurácia	61.5
Precisão	59.0
F1 Score	66.0

Tabela 6.Resultados obtidos para cada estratégia de treinamento do modelo com o classificador MLP

MLP classifier - 30000 Max_iter	***Resultados***
Acurácia	62.1
Precisão	59.0

F1 Score	66.0

Tabela 7.Resultados obtidos para cada estratégia de treinamento do modelo com o classificador MLP

MLP classifier - 1000000 Max_iter	***Resultados***
Acurácia	61.1
Precisão	58.0
F1 Score	66.0

A Tabela 8 apresenta os resultados obtidos com a técnica CNN, incluindo as medidas de precisão, acurácia e F1-Score (média harmônica entre precisão e revocação/cobertura). Os resultados foram obtidos usando validação cruzada (5 folds) e smote nos dados de treinamento. Para esse algoritmo utilizamos a biblioteca Keras. Foram adicionadas as seguintes camadas ao modelo: layers.Embedding(input_dim=230, output_dim=64, input_length=23); layers.Conv1D(filters=230, kernel_size=3, padding='same', activation='relu'); layers.MaxPooling1D(pool_size=2); layers.Dropout(0.5); layers.Flatten(); layers.Dense(22, activation="relu"); layers.Dense(1, activation="sigmoid"). Todos nessa ordem. Para compilação, model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Tabela 8.Resultados obtidos para cada estratégia de treinamento do modelo com CNN

CNN	***Resultados***
Acurácia	73.5
Precisão	92.8
F1 Score	83.8

A Tabela 9 apresenta os resultados obtidos com a técnica ANN, incluindo as medidas de precisão, acurácia e F1-Score (média harmônica entre precisão e revocação/cobertura). Os resultados foram obtidos usando validação cruzada (5 folds) e smote nos dados de treinamento. Para esse algoritmo utilizamos a biblioteca Keras. Este modelo foi construído com as seguintes camadas: layers.Dense(16, activation="relu"); layers.Dense(8, activation="relu"); layers.Dense(4, activation="relu"); layers.Dense(1, activation="sigmoid"), nesta ordem. Com as propriedades de compilação: model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Tabela 9.Resultados obtidos para cada estratégia de treinamento do modelo com ANN

ANN	***Resultados***
Acurácia	73.1
Precisão	90.2
F1 Score	84.4

A Tabela 10 apresenta os resultados obtidos com a técnica ANN, incluindo as medidas de precisão, acurácia e F1-Score (média harmônica entre precisão e revocação/cobertura). Os resultados foram obtidos usando validação cruzada (5 folds) e smote nos dados de treinamento. Para esse algoritmo utilizamos a biblioteca Keras. Isso para a feature EC (Estado Clínico), individualmente, para a predição de Recidiva. Com relação aos hiperparâmetros, teremos: Dense(78, activation='relu'), Dropout(0.2), Dense(39, activation='relu'), Dropout(0.2), Dense(19, activation='relu'), Dropout(0.2), Dense(units=1, activation='sigmoid'), nesta ordem. Com as propriedades de compilação, model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Tabela 10.Resultados obtidos para cada estratégia de treinamento do modelo com ANN

ANN	***Resultados***
Acurácia	51.0
Precisão	93.7
F1 Score	62.4

A Tabela 11 apresenta os resultados obtidos com a técnica ANN, incluindo as medidas de precisão, acurácia e F1-Score (média harmônica entre precisão e revocação/cobertura). Os resultados foram obtidos usando validação cruzada (5 folds) e smote nos dados de treinamento. Para esse algoritmo utilizamos a biblioteca Keras. A *feature* TRATCONS foi utilizada individualmente para prever a recidiva. Assim como o caso anterior teremos os hiperparâmetros: Dense(78, activation='relu'), Dropout(0.2), Dense(39, activation='relu'), Dropout(0.2), Dense(19, activation='relu'), Dropout(0.2), Dense(units=1, activation='sigmoid'), nesta ordem. Com as propriedades de compilação, model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Tabela 11.Resultados obtidos para cada estratégia de treinamento do modelo com ANN

ANN	***Resultados***
Acurácia	63.7
Precisão	89.8
F1 Score	70.3

A Tabela 12 apresenta os resultados obtidos com a técnica ANN, incluindo as medidas de precisão, acurácia e F1-Score (média harmônica entre precisão e revocação/cobertura). Isso, sobre os resultados da aplicação do algoritmo com técnicas de validação cruzada (5 folds) e smote nos dados de treinamento. Para esse algoritmo utilizamos a biblioteca Keras. A *feature* META01 foi utilizada individualmente para prever recidiva. Finalmente, os hiperparâmetros: Dense(78, activation='relu'), Dropout(0.2), Dense(39, activation='relu'), Dropout(0.2), Dense(19, activation='relu'), Dropout(0.2), Dense(units=1, activation='sigmoid'), nesta ordem. Com as propriedades de compilação, model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Tabela 11.Resultados obtidos para cada estratégia de treinamento do modelo com ANN

ANN	***Resultados***
Acurácia	84.0
Precisão	90.4
F1 Score	91.1

4.1. Interpretação e discussão dos resultados

No total foram aplicados 6 algoritmos na base de dados coletada e pré processada. Com isso, foi possível obter resultados em vários diferentes modelos, cada um com suas particularidades

Diferente do primeiro relatório do projeto, disponível em [Maeda et al. 2022], nos algoritmos aplicados foram utilizadas técnicas de validação cruzada e balanceamento de classes. Sendo assim, notou-se que os algoritmos não baseados em aprendizado de máquina mantiveram uma tendência de resultados próximos, mesmo quando comparados aos resultados apresentados no relatório anterior. Isto é, mantiveram um padrão de cerca de 60% de acurácia, com valores de precisão e F1-score medianos. Entretanto, ao observar os resultados apresentados para o algoritmo CNN e ANN de aprendizagem profunda notou-se que eles foram consideravelmente melhores, com acurácia de cerca de 70% e precisão, chegando até 80% de acurácia e F1-score ligeiramente melhores. Dessa forma, observa-se uma tendência de melhores resultados com o uso de *deep learning*.

5. Conclusão

Os resultados do relatório ainda são preliminares, mas já contribuem ao apontar para uma possível associação entre as features analisadas e a recidiva. Além disso, ao observar os resultados obtidos e os resultados anteriormente apresentados nota-se que o ganho da utilização da maioria destes novos algoritmos na maioria dos casos não foi significativo, com exceção do uso de ANN com as *feature* META01 o qual apresentou uma acurácia de 84%, sendo consideravelmente maior que os resultados obtidos anteriormente. Sendo assim, é necessário buscar as melhores configurações para os modelos e tentar compreender melhor as correlações dentro da base de dados. Dessa forma, é possível, e necessário, uma pesquisa detalhada sobre o tema. Uma das principais dificuldades é o custo computacional, entretanto com mais recursos é possível atingir de fato melhores resultados. Além disso, serão realizadas pesquisas adicionais sobre a relação entre estas features e a previsão de recorrência do câncer.

5. Agradecimentos

Este trabalho foi financiado em parte pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) processo nº 405352/2021-2. Agradecemos a Fundação Oncocentro de São Paulo por disponibilizar publicamente os dados utilizados neste projeto.

6. Referências

Bishop, C. M. (2006) Pattern recognition and machine learning. [S.l.]: springer.

Chawla, N. V. et al. (2002) Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research, v. 16, p. 321–357.

Maeda, A.E., Crocco, P.F., Ruppert, G.C.S, Dametto, M., Bonacin. R. Um Estudo sobre a Predição da Recidiva de Câncer Usando Técnicas de Aprendizado de Máquina. XXIV Jornada de Iniciação Científica do Centro de Tecnologia da Informação Renato Archer - JICC´2022 PIBIC/CNPq/CTI - Outubro de 2022 – Campinas – São Paulo. Disponível em: https://www.gov.br/cti/pt-br/publicacoes/producao-cientifica/jicc/xxiv-jicc-2022 Último acesso em 14 de agosto de 2023.

Müller, A. C., and Guido, S. (2016). Introduction to machine learning with Python: a guide for data scientists. " O'Reilly Media, Inc.

Keras. Simple, Flexible, Powerful. Disponível em: https://keras.io/ Último acesso14 em de agosto de 2023.