Aplicação de Técnicas de Aprendizado de Máquina para Predição do Estado Final do Paciente de Câncer usando Banco de Dados Tabulares Abertos

Vinícius Carvalho Pimpim[1], **Rodrigo Bonacin**[1] **(Orientador)**

vpimpim@cti.gov.br, rodrigo.bonacim@cti.gov.br

[1]Divisão de Metodologias da Computação – DIMEC

Centro de Tecnologia da Informação Renato Archer - CTI

Abstract. *This paper presents research on the application of machine learning and deep learning techniques in open tabular databases. The study initially focuses on the database made available by INCA (National Cancer Institute), which contains information on millions of cancer patients from all over Brazil. The focus was to investigate how to support health professionals in predicting the patient's condition after the first treatment.*

Resumo. *Nesse artigo apresenta uma pesquisa sobre a aplicação de técnicas de aprendizado de máquina e aprendizagem profunda em bases tabulares abertas. O estudo foca inicialmente na base de dados disponibilizada pelo INCA (Instituto Nacional do Câncer), no qual contém informações sobre milhões de pacientes com câncer de todo Brasil. O foco foi investigar como apoiar profissionais de saúde na predição do estado do paciente após o primeiro tratamento.*

1. Introdução

Com os avanços tecnológicos atuais e o desenvolvimento de novos algoritmos mais robustos de aprendizado de máquina, torna-se possível detectar padrões nos quais a análise humana não é capaz de identificar. Nesse contexto, é possível utilizar essas novas tecnologias com foco em aplicações na área da saúde. Assim, é possível prover a sociedade uma oportunidade de aprimoramento, via tecnologia, da prevenção, dos cuidados e dos tratamentos para doenças graves tais como o câncer. Atualmente, a quantidade de dados cresce exponencialmente, tornando viável o uso de ferramentas de predição que utilizam esses dados para tal.

Com isso, o advento dos big data e o uso de registros eletrônicos de saúde permitem explorar soluções para questões de saúde populacional, antes consideradas inviáveis. Além disso, a integração de dados clínicos e genotípicos em plataformas unificadas possibilita a descoberta de novos padrões biológicos, promovendo um entendimento mais amplo da progressão de doenças [Ngiam & Khor, 2019].

Nesse sentido, utilizando uma base de dados fornecida pelo INCA (Instituto Nacional do Câncer), contendo dados tabulares, foram aplicados diversos algoritmos de aprendizado de máquina por meio do método de aprendizado supervisionado. Com objetivo de auxiliar na predição do estado final do paciente ao final do primeiro

tratamento do câncer, foram exploradas técnicas de aprendizado de máquina e aprendizagem profunda. As técnicas que mais se destacaram incluem: XGBoost[1], um algoritmo baseado em árvores de decisão que aplica um gradiente descendente, no qual cada árvore seguinte tende a corrigir os erros que as anteriores causaram. Tabnet[2], uma rede neural profunda que utiliza mecanismos de atenção, sendo capaz de identificar entre um conjunto de dados as *features* mais relevantes. Catboost[3] é outro algoritmo que combina árvores de decisão com boosting por gradiente, otimizando o processo de treinamento para lidar com dados categóricos de modo eficiente.

O restante deste documento está organizado da seguinte maneira: a seção 2 apresenta a base de dados, a seção 3 detalha as técnicas aplicadas, incluindo configurações, treinamento e resultados obtidos, por fim, a seção 4 conclui este artigo e apresenta os próximos passos da pesquisa.

2. Base de Dados

A base de dados em questão foi fornecida pelo INCA (Instituto Nacional do Câncer) e conta ao todo dados de *2.973.447* pacientes, e foi disponibilizada no formato csv (*Comma-Separated Values*) na internet.

2.1 Detalhamento da base de dados

Buscando uma redução de parâmetros incompatíveis ou desnecessários, um tratamento inicial dos dados foi realizado, constituindo ao final um conjunto de 14 classes utilizadas para a predição, que podem ser verificadas na tabela presente na seção Apêndice A. Para a classe alvo, onde o objetivo é determinar qual o estado final do paciente, contava com os seguintes valores: *(1) Remissão completa, (2) Remissão parcial, (3) Doença estável, (4) Doença em progressão, (5) Suporte terapêutico oncológico, (6) Óbito, (8) Não se aplica e (9) Sem informação*. A Figura 1 apresenta o quantitativo inicial de pacientes por classe preditora.

[1] https://xgboost.readthedocs.io/
[2] https://pypi.org/project/pytorch-tabnet/
[3] https://catboost.ai/

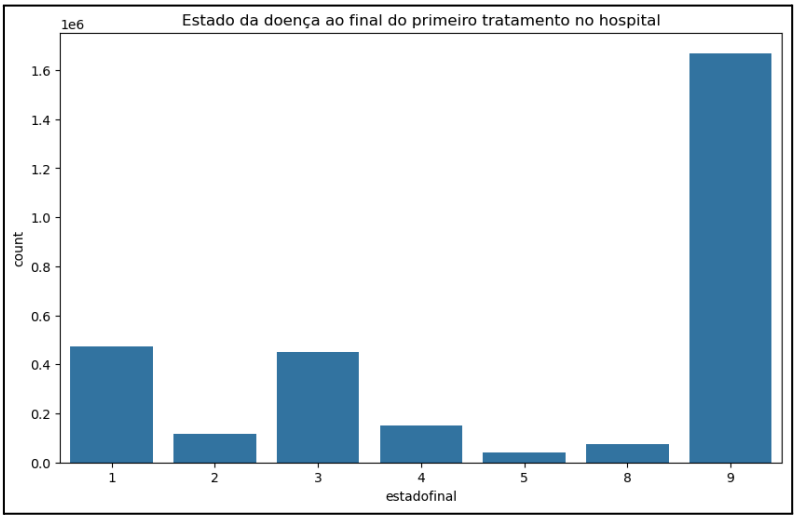

Figura 1. Quantitativo inicial de pacientes por classe preditora

2.2 Classe previsora utilizada no treinamento do modelo

Na etapa de tratamento de dados optamos por remover os seguintes valores: *(5) Suporte terapêutico oncológico, (6) óbito, (8) Não se aplica e (9) sem informação*. Foi decidido a remoção desses pois os valores (5) Suporte terapêutico oncológico, (8) não se aplica e (9) sem informação mostraram-se imprecisos, enquanto o valor (6) óbito não havia número substancial de casos, além disso foi considerado que essas classes removidas, não se enquadram no escopo objetivo dessa pesquisa de prever o estado final do câncer. Assim, para o treinamento dos modelos a classe preditora contou com: ***(0) remissão completa, (1) Remissão parcial, (2) Doença estável e (3) Doença em progressão.*** A Figura 2 apresenta a quantidade de pacientes por classe preditora após o tratamento de dados.

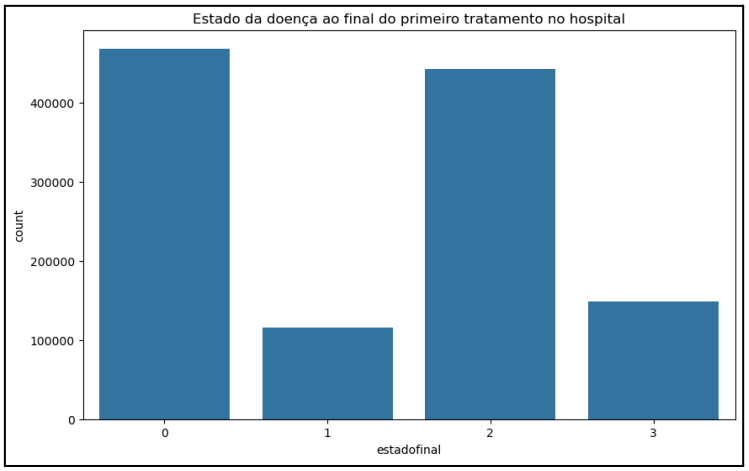

Figura 2. Quantitativo de pacientes por classe preditora após tratamento de dados.

2.3 Desafios encontrados na Base de dados

Um problema relevante encontrado na base de dados foi a dominância considerável da quantidade de dados com rótulo *"Sem informação"* . Esses dados não puderam ser utilizados no treinamento, pois são equivalentes a valores nulos. Além disso, algumas colunas de dados contavam com valores mistos e inválidos.

Ao todo o conjunto de dados conta com dados de cerca de 1.760.000 pacientes com o rótulo sem informação, em que mesmo após aplicados testes utilizando algoritmos de reconstrução de dados faltantes como o KNN (K-Nearest Neighbors), que faz uso da proximidade entre os dados para estimar os valores ausentes, identificando os k vizinhos mais próximos e substituindo o valor faltante pela média dos rótulos desses vizinhos, os resultados não foram satisfatórios. Diante disso, optou-se pela remoção desses registros. Assim, os treinamentos foram realizados com cerca de 40% do conjunto de dados original (Figura 2).

3. Técnicas Aplicadas

Conforme supracitado, os algoritmos que tiveram melhores resultados em testes preliminares são o XGboost, Tabnet e Catboost. O uso desses algoritmos é detalhado nesta seção. Foram utilizadas técnicas de buscas em hiperparâmetros, a fim de se encontrar a melhor combinação de valores para a construção do modelo. Em específico a busca Bayesiana. Essa abordagem ajusta os parâmetros iterativamente, avaliando os resultados obtidos em cada etapa, priorizando dentro do espaço de busca, regiões mais promissoras, com base nos resultados obtidos a cada iteração, reduzindo tentativas em áreas que provavelmente não encontrarão uma solução adequada. O processo de busca é otimizado, o que leva a encontrar os parâmetros que fornecem uma solução satisfatória mais rapidamente.[Gorodetski, 2021].

O método *BayesSearchCV* pertencente à biblioteca Python *scikit-optimize*[4] foi utilizado para este fim.

3.1 Divisão de dados e Tratamento de atributos categóricos

A divisão utilizada foi de 10% dos dados para o teste (Figura 3), o restante foi utilizado para treinamento e validação cruzada.

```
X_train, X_test, y_train , y_test  = train_test_split(X, y, test_size = 0.10, random_state = 8)
```

Figura 3. Separação de dados para teste.

O tratamento de atributos categóricos foi aplicado às *features* TNM e estadiamento clínico do tumor pois são categóricas e apresentam muitos valores

[4] https://scikit-optimize.github.io/

distintos e não ordinais respectivamente *302* e *51*. Conforme apresenta a Figura 4, o *encoder* (codificador) utilizado foi o *TargetEncoder*. Esse *encoder* realiza a transformação com base no conjunto de treino, aproveitando a relação entre os atributos categóricos e a classe previsora. Em seguida (Figura 5) foi usado o algoritmo *StandardScaler*, no qual faz uma modifica os dados para que tenham média zero e desvio padrão um.

```
from category_encoders.target_encoder import TargetEncoder
encoder = TargetEncoder(cols=['tnm', 'estadiam'])
X_train = encoder.fit_transform(X_train,y_train)
X_test = encoder.transform(X_test)
```

Figura 4. Trecho de programa para codificar (encoder) os atributos categóricos.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_sc = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_test_sc = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)
```

Figura 5. Trecho de programa para ajuste da escala de códigos.

3.3 XGboost

Para o XGboost, a fim de determinar uma boa combinação de hiperparâmetros, foi construído um espaço de busca (Figura 6), combinado com a biblioteca *scikit-optimize*, onde usou-se o método *BayesSearchCV* que implementa a otimização bayesiana de busca sobre os hiperparâmetros (Figura 6).

```
estimators = [
    ('clf', XGBClassifier(random_state = 8))
]
pipe = Pipeline(steps=estimators)
search_space = {
    'clf__max_depth': Integer(2,8),
    'clf__learning_rate': Real(0.001, 1.0, prior='log-uniform'),
    'clf__subsample': Real(0.5, 1.0),
    'clf__colsample_bytree': Real(0.5, 1.0),
    'clf__colsample_bylevel': Real(0.5, 1.0),
    'clf__colsample_bynode' : Real(0.5, 1.0),
    'clf__reg_alpha': Real(0.0, 10.0),
    'clf__reg_lambda': Real(0.0, 10.0),
    'clf__gamma': Real(0.0, 10.0)
}
opt = BayesSearchCV(pipe, search_space, cv=3, n_iter=100, random_state=8)
opt_result = opt.fit(X_train_sc.values, y_train.values)
best_model = opt_result.best_estimator_
```

Figura 6. Configuração do espaço de busca e ajuste de hiperparâmetros para o XGboost

Após a realização do treino, verificou-se que os melhores parâmetros encontrados para o XGboost, foram: *colsample_bylevel = 1.0, colsample_bynode = 1.0,*

colsample_bytree = 1.0, gamma = 0.0, learning_rate = 0.5966432489601267, max_depth = 8, reg_alpha = 10.0, reg_lambda = 0.0, subsample = 1.0

Com essa configuração foi obtida a acurácia de *0,696*. A Figura 7 apresenta o relatório com as métricas de precisão, revocação e *F1Score* por classe (lado esquerdo), bem como a matriz de confusão para o algoritmo XGboost (lado direito).

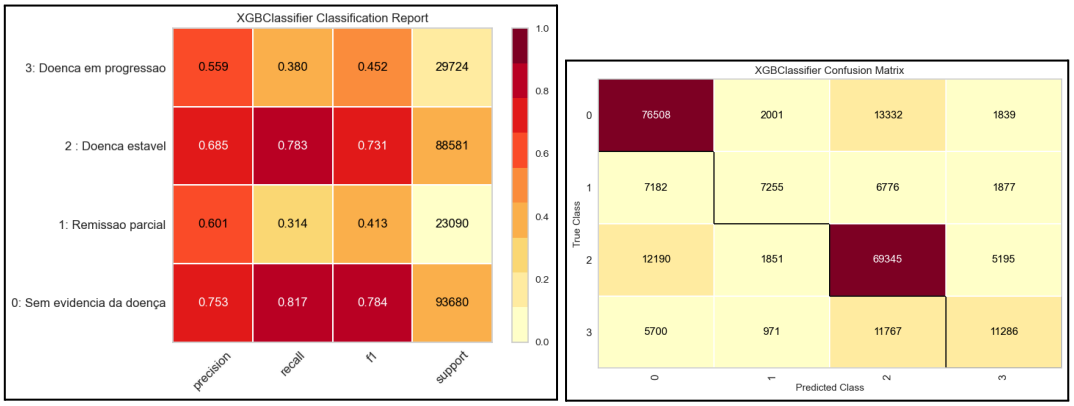

Figura 7. Relatório de métricas obtidas e matriz de confusão para o algoritmo XGboost

Para a verificação da qualidade do modelo, também foi utilizada a métrica AUC, obtida através do cálculo da área sob a curva ROC, a qual é construída traçando a taxa de verdadeiros positivos (TPR) contra a taxa de falsos positivos (FPR) para diferentes limiares de decisão. Valores de AUC próximos a 1 indicam que o modelo consegue separar os verdadeiros positivos dos falsos positivos [Avelar, 2019].

Para tal foi aplicada a técnica *One vs Rest* (OvR), que transforma problemas de classificação multiclasse em classificação binária, onde é escolhida uma classe de interesse, colocando-a contra todas as outras, essas são agrupadas como resto.

A Figura 8 apresenta as curvas AUC e a Tabela 1 os valores por classe usando OvR para o algoritmo XGboost. Tais figuras foram construídas a partir dos *scripts* em python disponibilizado por Trevisan (2022).

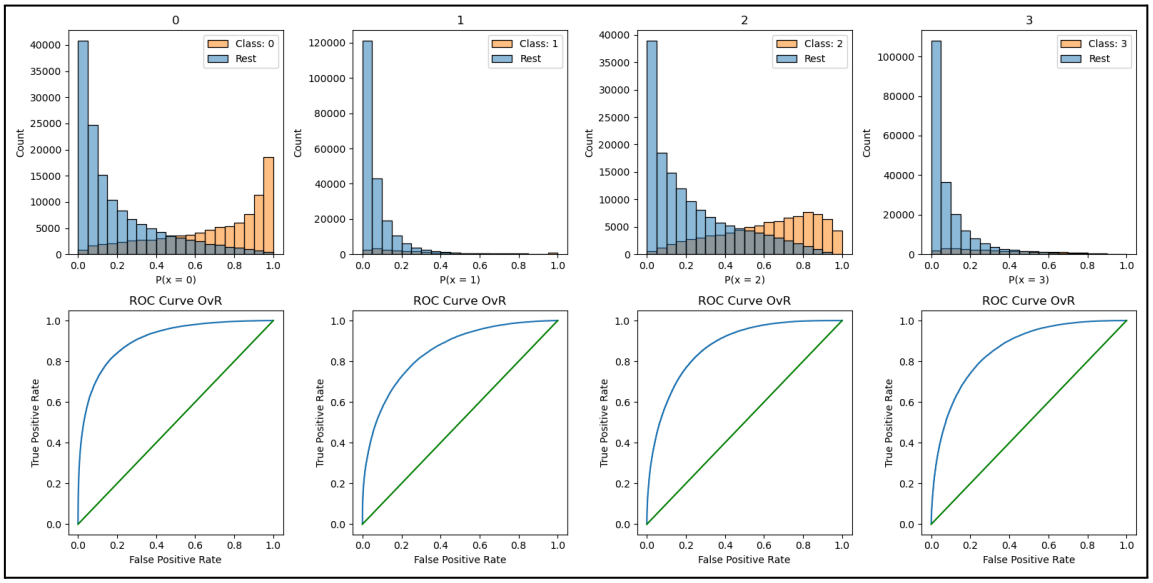

Figura 8. Curvas AUC para o algoritmo XGboost

Tabela 1. Resultado da medida AUC por classe para o algoritmo XGboost

ROC AUC OvR	
0. remissão completa	0,9049
1.Remissão parcial	0,8488
2.Doença estável	0,8703
3.Doença em progressão	0,8569
average ROC AUC OvR:	0,8702

3.4 Tabnet

Para a rede neural profunda Tabnet foi utilizado o otimizador Adam, unido a um espaço simples de busca, de forma a reduzir o esforço computacional durante o treinamento. Novamente, foi necessário realizar a busca em hiperparâmetros através do método *BayesSearchCV* (Figura 9).

```
estimators = [
    ('clf', TabNetClassifier(optimizer_fn=torch.optim.Adam,
                             scheduler_params={"step_size":10,
                                               "gamma":0.9},
                             scheduler_fn=torch.optim.lr_scheduler.StepLR))
]
pipe = Pipeline(steps=estimators)
search_space = {
    'clf__n_d': Integer(8, 64),
    'clf__n_a': Integer(8, 64),
    'clf__n_steps': Integer(3, 10),
    'clf__gamma': Real(1.0, 2.0),
    'clf__lambda_sparse': Real(0.0, 1.0),
    'clf__momentum': Real(0.01, 0.4),
    'clf__clip_value': Real(1.0, 5.0),
    'clf__mask_type': Categorical(['sparsemax', 'entmax']),
    'clf__n_shared': Integer(1, 4),
    'clf__n_independent': Integer(1, 4),
}
opt = BayesSearchCV(pipe, search_space, cv=2, n_iter=n_iter, scoring='accuracy', random_state=8)
opt_result = opt.fit(X_train_sc.values, y_train,
                     clf__eval_set=[(X_train_sc.values, y_train), (X_test_sc.values, y_test)],
                     clf__eval_name=['train', 'test'])
best_model = opt_result.best_estimator_
```

Figura 9. Configuração do espaço de busca e ajuste de hiperparâmetros para a Tabnet

Após a realização do treino, verificou-se que os melhores parâmetros encontrados para a Tabnet, foram: *clip_value = 4.4440484076201185 , gamma = 1.7612392999500042, lambda_sparse = 0.0, mask_type = "entmax", momentum = 0.3687172834647959, n_a = 64 , n_d = 8, independent = 2, n_shared = 2, n_steps = 3.*

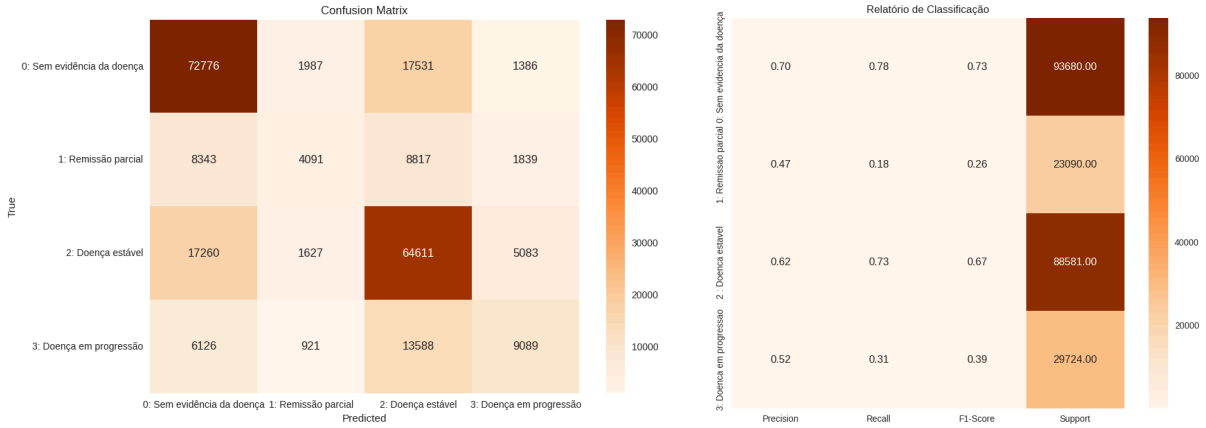

Figura 10. Relatório de métricas obtidas e matriz de confusão para a Tabnet

Com essa configuração foi obtida a acurácia de *0,636*. A Figura 10 apresenta o relatório com as métricas de precisão, revocação e *F1Score* por classe (lado esquerdo), bem como a matriz de confusão para a rede neural profunda Tabnet (lado direito).

A Figura 11 detalha a função de perda, que está diretamente relacionada à taxa de aprendizado durante o treinamento.

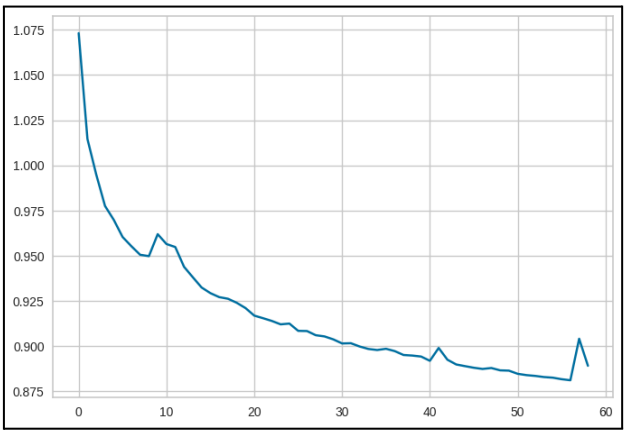

Figura 11. Função perda durante o treinamento da Tabnet.

A Figura 12 apresenta as curvas AUC e a Tabela 2 os valores por classe usando OvR para a Tabnet.

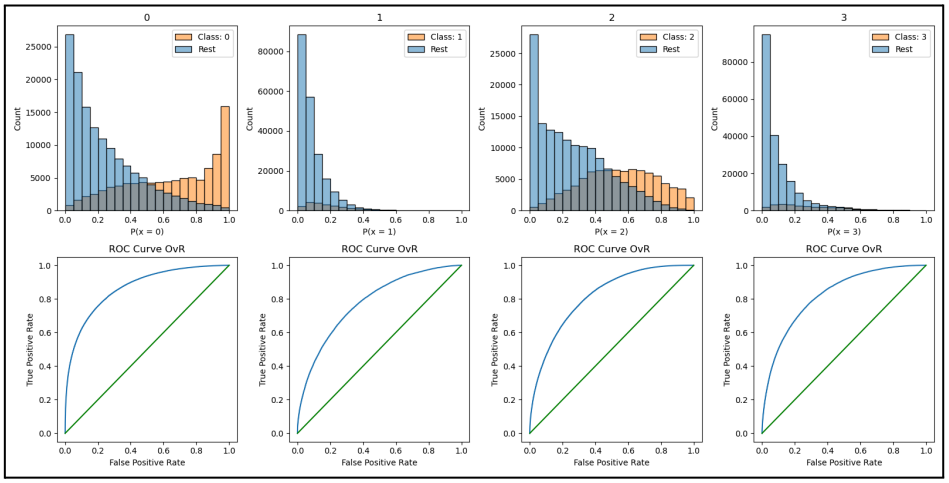

Figura 12. Curvas AUC para a Tabnet

Tabela 2. Resultado da medida AUC por classe para a Tabnet

ROC AUC OvR	
0. remissão completa	0,8652
1.Remissão parcial	0,7777
2.Doença estável	0,8151
3.Doença em progressão	0,8237
average ROC AUC OvR:	0,8204

3.5 Catboost

Devido a grande quantidade de hiperparâmetros presentes no algoritmo Catboost, foi usado, durante o treinamento, novamente a otimização de busca *BayesSearchCV* da biblioteca *scikit-optimize*. Nesse caso, foi realizado uma validação cruzada de 2, e uma quantidade de iterações para busca do melhor modelo de 100.

```
estimators = [
    ('clf', CatBoostClassifier())
]

pipe = Pipeline(steps=estimators)
search_space = {
    'clf__iterations': Integer(10, 4000),
    'clf__depth': Integer(1, 12),
    'clf__learning_rate': Real(0.01, 1.0, 'log-uniform'),
    'clf__random_strength': Real(1e-9, 10, 'log-uniform'),
    'clf__l2_leaf_reg': Integer(2, 100),
    'clf__bagging_temperature': Real(0.0, 1.0),
    'clf__l2_leaf_reg': Integer(2, 100),
    'clf__border_count': Integer(32, 100),
   }

opt = BayesSearchCV(pipe, search_space, cv=2, n_iter=n_iter, scoring='accuracy', random_state=8)
opt_result = opt.fit(X_train_sc.values, y_train)
best_model = opt_result.best_estimator_
```

Figura 13. Configuração do espaço de busca e hiperparâmetros para a Catboost

Assim os melhores hiperparâmetros obtidos foram: *bagging_temperature = 0.0, border_count = 100, depth = 10 , iterations = 4000, l2_leaf_reg = 2 , learning_rate = 0.05180818662890271, random_strength = 1e-09*

Com essa configuração foi obtida a acurácia de *0,693*. Figura 14 apresenta o relatório com as métricas de precisão, revocação e F1Score por classe (lado esquerdo), bem como a matriz de confusão para o algoritmo Catboost (lado direito).

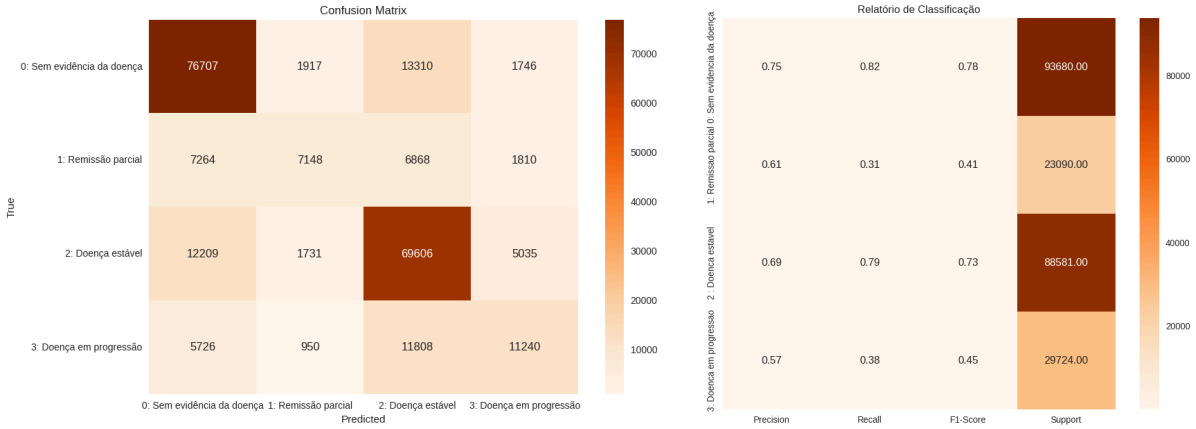

Figura 14. Relatório de métricas obtidas e matriz de confusão para a Catboost

A Figura 15 apresenta as curvas AUC e a Tabela 3 os valores por classe usando OvR para o algoritmo Catboost.

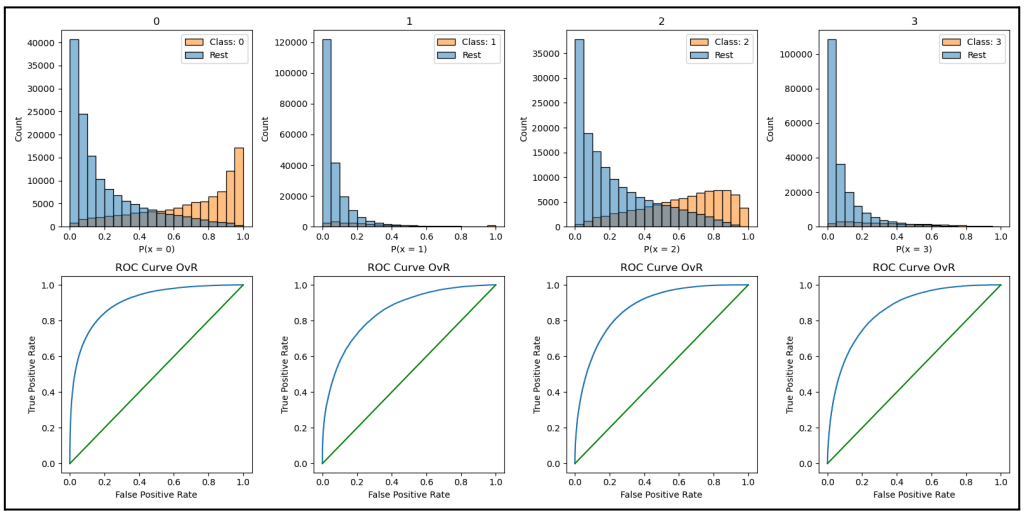

Figura 15. Curvas AUC para o algoritmo Catboost

Tabela 3. Resultado da medida AUC por classe para o algoritmo Catboost

ROC AUC OvR	
0. remissão completa	0,9054
1.Remissão parcial	0,8488
2.Doença estável	0,8710
3.Doença em progressão	0,8581
average ROC AUC OvR:	0,8708

4.Conclusão

Algoritmos de aprendizado de máquina e aprendizagem profunda podem ser empregados para auxiliar profissionais de saúde no diagnóstico e tratamento de doenças. Este artigo apresentou estudo com técnicas para auxílio à predição do estado final de pacientes de câncer depois do primeiro tratamento usando a base de dados tabular aberta do INCA. Os resultados confirmam o potencial desta abordagem, com destaque às técnicas de Catboost e o XGBoost, apresentando métricas AUC média por volta de *0,87*.

A dificuldade enfrentada nos modelos para aumentar os resultados nas classes 1 e 3 pode estar relacionado em essas serem as classes minoritárias, mesmo com auxílio da técnica de *oversampling* no conjunto de treinamento. Estudos adicionais são necessários para lidar com este desbalanceamento. Entre os problemas enfrentados para a realização desta pesquisa destaca-se o desafio em lidar de maneira adequada com as

várias características e especificações dos dados contidos nesta base de dados, uma vez que determinados atributos contavam com informações referentes à área da saúde, sendo necessário um conhecimento multidisciplinar para esse trabalho.

Outro fato importante, é que uma grande quantidade da base de dados não foi utilizada devido a quantidade dominante de valores nulos na classe previsora, dificultando a coleta de dados explicativos para a aprendizagem do modelo. Com isso, é necessário continuar buscando maneiras de contornar problemas do conjunto de dados, no qual uma alternativa a ser investigada é o aprendizado semi-supervisionado, em que é possível realizar o treinamento com dados faltantes [Arik & Pfister, 2020] por meio da rede Tabnet.

Por fim, após a realização dessa pesquisa é importante destacar a possibilidade de integração de um modelo preditivo de inteligência artificial, dentro do domínio da oncologia. Isso pode ser viabilizado através do desenvolvimento de um aplicativo ou website em que o usuário pode inserir as informações necessárias para a predição e receber os resultados gerados pelo modelo. Esse método tem o potencial de auxiliar profissionais responsáveis pelo tratamento de câncer, oferecendo informações úteis sobre o possível desenvolvimento da doença

Como trabalho futuro, também é proposto o uso de outras técnicas e arquiteturas, bem como expandir o estudo para outras bases abertas.

Referencias

ARIK, S. O.; PFISTER, T. TabNet: Attentive Interpretable Tabular Learning. arXiv:1908.07442 [cs, stat], 9 dez. 2020.

AVELAR, A. (2019). O que é AUC e ROC nos modelos de Machine Learning - Adriano Avelar - Medium. Disponível em: <https://medium.com/@eam.avelar/o-que-%C3%A9-auc-e-roc-nos-modelos-de-machine-learning-2e2c4112033d>. Acesso em: 25 ago. 2024.

GORODETSKI, M. (2021) Hyperparameter Tuning Methods — Grid, Random or Bayesian Search? Towards Data Science, Ago, 2021. Disponível em: <https://towardsdatascience.com/bayesian-optimization-for-hyperparameter-tuning-how-and-why-655b0ee0b399>. Acesso em 10 de setembro de 2024.

NGIAM, K.Y., KHOR, I.W. (2019) Big data and machine learning algorithms for health-care delivery. Lancet Oncol. 2019 May;20(5):e262-e273. doi: 10.1016/S1470-2045(19)30149-4. Erratum in: Lancet Oncol. 2019 Jun;20(6):293. doi: 10.1016/S1470-2045(19)30294-3. PMID: 31044724.

TREVISAN, V. (2022) Multiclass classification evaluation with ROC Curves and ROC AUC. Towards Data Science, Feb, 2022. Disponível em:

<https://towardsdatascience.com/multiclass-classification-evaluation-with-roc-curves-and-roc-auc-294fd4617e3a>. Acesso em 10 de setembro de 2024.

Apêndice A

Tabela 4. Identificação das variáveis preditoras utilizadas para o processo de treinamento

Classe preditora	Descrição	Categorias/Códigos
Sexo	Gênero do paciente	1. Masculino, 2. Feminino
Raça/Cor	Cor/etnia declarada	1. Branca, 2. Preta, 3. Amarela, 4. Parda, 5. Indígena, 9. Sem informação
Clínica de início do tratamento	Clínica onde o tratamento foi iniciado	Codificação da tabela de clínicas do SisRHC
Histórico familiar de câncer	Histórico de câncer na família	1. Sim, 2. Não, 9. Sem informação
Histórico de consumo de álcool	Histórico de consumo de bebida alcoólica	1. Nunca, 2. Ex-consumidor, 3. Sim, 4. Não avaliado, 8. Não se aplica, 9. Sem informação
Histórico de consumo de tabaco	Histórico de consumo de tabaco	1. Nunca, 2. Ex-consumidor, 3. Sim, 4. Não avaliado, 8. Não se aplica, 9. Sem informação
Localização primária do tumor	Localização inicial do tumor	Código da CID-O (3 dígitos)
Estádio clínico (TNM)	Codificação estádio clínico (TNM)	Codificação TNM
Estadiamento clínico	Estádio clínico do tumor segundo a classificação TNM	Codificação de grupamento TNM
Primeiro tratamento	Primeiro tratamento hospitalar	1. Nenhum, 2. Cirurgia, 3. Radioterapia, 4. Quimioterapia, 5. Hormonioterapia, 6. Transplante de

recebido		medula óssea, 7. Imunoterapia, 8. Outras, 9. Sem informação
Ocupação principal	Codificação trabalhista	Codificação da Tabela do Código Brasileiro de Ocupações
Idade na primeira consulta	Idade	Valor inteiro
Dias após a primeira consulta	Tempo após a primeira consulta (em dias)	Valor inteiro
Tipo histológico do tumor	Classificação morfológica do tumor	Codificação da morfologia do tumor pela CID-O