Relatório Final do Trabalho Prático de Causalidade em Machine Learning

Aprendizado de Máquina Causal utilizando o dataset TWINS

Arthur Veloso Kuahara

Daniel Souza de Campos

Renato Polanczyk Resende

1 - Introdução

A mortalidade infantil é um problema muito grave, principalmente, em países subdesenvolvidos. Dessa forma, inúmeros estudos já foram feitos tentando prever a chance de um recém-nascido vir a óbito utilizando de machine learning[7][8]. Trabalhos nessa área estão presentes em todos os países devido principalmente a uma maior demanda na melhora da saúde neonatal. Regiões de maior taxa de mortalidade infantil devido a condições socioeconômicas extremas inclusive usam, muitas vezes, features peculiares, como o nível de higiene dos banheiros e o nível de educação da mãe[4][5][6].

Além dessas features convencionais e inusitadas, também é possível utilizar dados em formato de time-series com Deep Learning para obter bons resultados[9].

Dessa forma, com um modelo de machine learning treinado, seria de interesse conseguir prever se um bebê irá morrer ou não. No caso positivo, também seria de grande contribuição entender o que deveria ser diferente nas características disponíveis no bebê para que ele não venha a óbito.

Assim, o objetivo desse trabalho é utilizar do dataset TWINS e do algoritmo NICE (ambos detalhados nas próximas seções) para tentar prever a possibilidade de falecimento em um bebê e quais deveriam ser as suas características para que ele não morresse. Se o trabalho tiver êxito, isso poderia ser utilizado para, o quanto antes, tentar melhorar as condições do nascimento do bebê e aumentar sua probabilidade de viver.

2 - Dataset TWINS

O Dataset TWINS[1] é um conjunto de dados que já foi citado como sendo padrão recomendado para aprender sobre efeitos causais[2][3]. Esse dataset se refere à mortalidade infantil em pares de gêmeos nascidos nos EUA durante o período de 1989 até 1991.

De acordo com o lugar onde conseguimos esse dataset, ele é composto de três arquivos diferentes. São eles:

- twin_pairs_X_3years_samesex.csv: Esse arquivo possui 71345 entradas com 52 *features*. Essas *features* já serão detalhadas. Cada entrada se refere aos dois gêmeos ao mesmo tempo, já que eles possuem os mesmos pais mas também existem *features* específicas para cada bebê
- twin_pairs_T_3years_samesex.csv: Esse arquivo possui os pesos dos dois bebês
- twin_pairs_Y_3years_samesex.csv: Esse arquivo possui a informação de se um bebê morreu ou não.

As *features* no arquivo principal podem ser divididas, basicamente, em: *features* presentes na mãe que aumentam o risco de morte no bebê e *features* sociais sobre os pais como raça, idade, onde nasceu etc…

As *features* em todos os arquivos são, em sua maioria, categóricas, com exceções dos pesos dos bebês, idades dos pais e outras.

O nome das colunas em si são confusos, entretanto, também existe um pequeno arquivo de metadados sobre as colunas que explica o que elas são. Alguns exemplos de colunas presentes são:

- alcohol: Fator de risco de uso de álcool
- anemia: Fator de risco de anemia
- cigar6: Número de cigarros fumados por dia
- diabetes: Fator de risco de diabetes
- drink5: Número de *drinks* bebidos por dia
- mager8: Idade da mãe
- phyper: Fator de risco de hipertensão relacionado à gravidez

3 - Nearest Instance Counterfactual Explanations (NICE)

Nearest Instance Counterfactual Explanations[1º] (NICE) é um algoritmo proposto para gerar explicações contrafactuais para dados tabulares heterogêneos.

O termo ’Explicações Contrafactuais’ faz referência a um cenário hipotético, nos moldes de ”Se um evento A não tivesse acontecido, então um outro evento B também não teria acontecido”. Por exemplo, se uma pessoa não tivesse atendido ao telefone, então ela não teria conversado com quem a ligou.

No contexto de machine learning, a ação consistiria na mudança das features utilizadas pelo modelo e o novo resultado final seria o segundo evento. Assim, é possível modificar parâmetros passados ao algoritmo para verificar se as chances de um dos gêmeos sobreviver é maior ou menor do que a chance obtida anteriormente.

3.1 - Funcionamento do NICE

O algoritmo NICE tenta encontrar o número mínimo de alterações nas features de uma entrada x_0 de modo a classificar o resultado final x_c em outra banda com base nos dados de treino disponíveis. Seu funcionamento se dá como o a seguir:

1. O primeiro passo é encontrar o vizinho de categoria diferente x_n mais próximo (Nearest Unlike Neighbor) a x_0, ou seja, em que f(x_0) != f (x_n), presente nos dados de treino. Além disso, iniciamos o contrafactual objetivo como sendo igual a x_0 (x_c = x_0).
2. O segundo passo é identificar uma lista L_f de features nos quais x_c e x_n divergem.
3. Em terceiro, serão gerados vetores de features híbridos entre x_c atual e x_n chamados de x_{h,i} a cada iteração i. Nessas iterações, para cada feature de nome L_f [j] divergente em L_f, formamos um vetor x_{h,i} exatamente igual ao x_c atual mas com o valor da feature x_c [L_f [j]] assumindo o valor de x_n[L_f [j]], ou seja, x_c[L_f [j]] = x_n[L_f [j]].
4. Para cada um desses vetores gerados, calcula-se um valor de recompensa R. Salvamos o vetor hébrido com tal maior valor de recompensa: x_{h,i}. Esse vetor possui a feature L_f [j] como a que foi alterada.
5. Define-se xc = x_{h,i}. Compara-se a predição do modelo f (x_c) com f (x_0). Se forem diferentes, então o contrafactual foi encontrado, portanto, retorna-se x_c. Se não, retira-se L_f [j] de L_f e recomeça o processo a partir do passo 3.

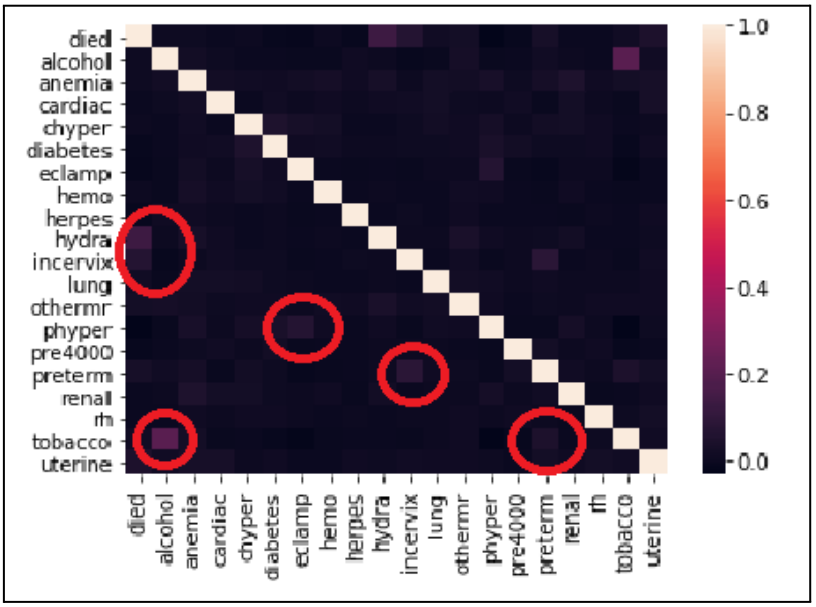
Como outro exemplo na vida real, podemos considerar a negação de um empréstimo, por exemplo. Qual seria a menor mudança possível no perfil do solicitante para que o empréstimo fosse aprovado de acordo com dados já presentes na base de treino?

4 - Exploração dos dados

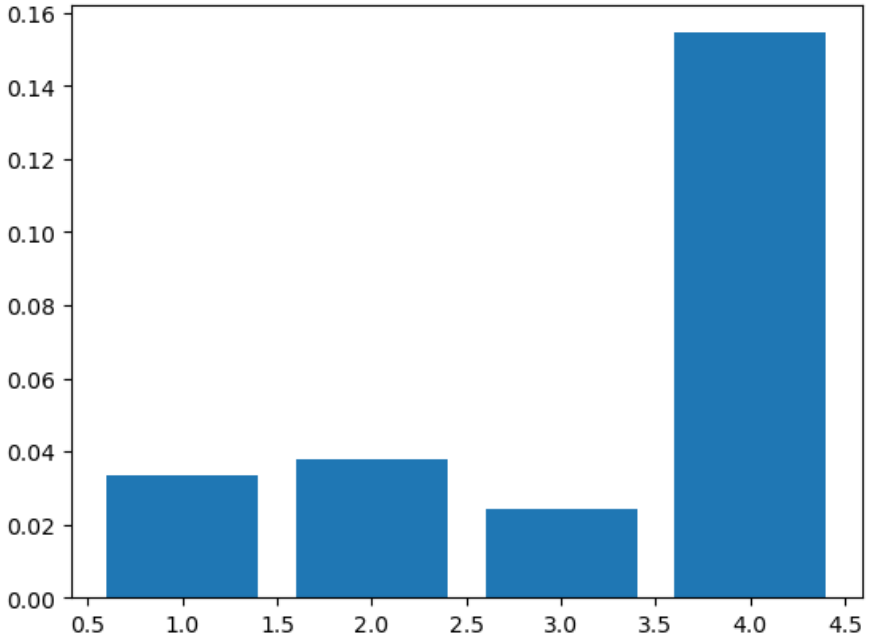
A exploração inicial dos dados foi focada nas correlações entre os diferentes fatores de risco e sua relação com a taxa de mortalidade da população neonatal. Um dos objetivos principais foi encontrar correlações espúrias entre fatores de risco e a mortalidade, um exemplo disso sendo um fator de risco relacionado a uma menor taxa de mortalidade na população em que ele está presente. Dentre as correlações encontradas, algumas delas caíram na categoria de correlações espúrias.

Primeiramente, as maiores correlações entre fatores de risco e morte se deram nas features “hydra” e “incervix”, que dizem a respeito de um aumento no

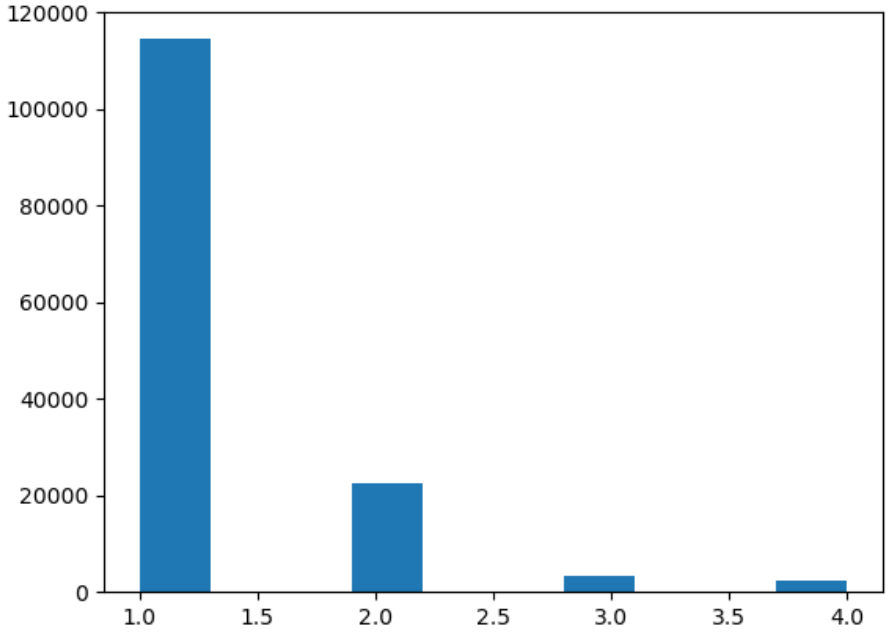
líquido amniótico e cérvix incompetente, respectivamente. Outras correlações observadas são, principalmente, entre fatores de álcool e tabaco (quem bebe na gravidez também tende a fumar), eclâmpsia e hipertensão relacionada à gravidez (eclâmpsia é um aumento anormal na pressão), cérvix impróprio e nascimentos prematuros e uso de tabaco e nascimentos prematuros.



Além disso, foi explorado o efeito do acompanhamento pré-natal na mortalidade neonatal. Pudemos perceber que com qualquer nível de acompanhamento pré-natal, a taxa de mortalidade é consideravelmente menor do que sem nenhum acompanhamento, mesmo que ele comece em trimestres mais tardios da gravidez. A figura a seguir mostra a mortalidade dos recém nascidos divididos pelo trimestre de começo do cuidado neonatal, sendo a categoria 4 a ausência de cuidado.



Vale ressaltar, no entanto, que na vasta maioria dos casos o tratamento neonatal tem início nos primeiros trimestres da gravidez, como recomendado, exibido na figura abaixo:



4.1- Follow-Up da Apresentação Parcial

Na análise exploratória dos dados, foram encontradas algumas correlações inesperadas, uma delas sendo entre a taxa de mortalidade e a presença do fator de risco para a eclâmpsia. Indivíduos com o fator de risco tinham uma taxa de mortalidade bem menor do que os demais, algo análogo ao resultado de um artigo famoso já publicado na área da saúde. Nesse artigo, relacionado à asma, adultos com asma controlada costumam sofrer menos complicações hospitalares por se tratarem melhor e acompanharem de perto a saúde, justamente por causa da presença do fator de risco aumentado.

Apesar da situação ser de fato análoga e da conjuntura fazer sentido, nenhuma informação presente no dataset corroborou com a ideia. Utilizando métricas relacionadas ao cuidado neonatal, não percebemos grandes diferenças entre o comportamento de indivíduos com o fator de risco presente e ausente.

Dentre os resultados, temos a média do começo do cuidado pré-natal (1.26 com o fator presente, 1.26 com o fator ausente), medido em trimestres da gravidez; O número quintil da quantidade de visitas clínicas (1.93 com o fator presente, 1.90 com o fator ausente) e também um parâmetro chamado de Adequacy, que mede o quão adequado o atendimento foi (1.31 com o fator presente, 1.31 com o fator ausente). Apesar desse parâmetro parecer bastante subjetivo, uma vez que os dados não são providos por apenas um hospital e os padrões podem variar bastante, achamos importante incluí-lo como métrica nesse caso.

4.1.1- Feature de Dados Faltantes

Como mencionado no próximo item, ao longo do desenvolvimento do projeto e por sugestões feitas durante a apresentação parcial, foi incluída uma nova feature ao modelo: a de dados faltantes. Ela nos dá a oportunidade de ganhar novos insights sobre os dados e fazer suposições sobre o nosso modelo.

Por exemplo, caso os dados tenham sido corrompidos ao longo do tempo depois da publicação do dataset, é esperado que a taxa de mortalidade dentre as linhas do dataset afetadas seja semelhante à do resto do dataset. No entanto, vemos que existe uma diferença relativamente grande ao segregar o dataset em relação à integridade dos dados. Enquanto a taxa de mortalidade dos indivíduos sem dados faltando foi de 2.9%, essa mesma métrica para os indivíduos com NaN's em suas linhas foi de 4.1%, um aumento de quase 50%. Além disso, a porcentagem de indivíduos que morreram e tinham dados faltando foi de 63%, enquanto a de indivíduos vivos com informações faltando foi de 54%.

Essas métricas nos indicam que pode existir um motivo para esses dados estarem faltando além do acaso de uma eventual corrupção do dataset. Sendo assim, podemos supor que esses dados já entraram no dataset como faltantes, o que pode ter ocorrido por alguns motivos: falta de familiaridade com prontuários eletrônicos (dados foram obtidos no começo da década de 1990), falta de organização dos funcionários hospitalares, erros de logística, falta de atenção, diferentes critérios médicos em diferentes hospitais…)

Seja o que for, qualquer motivo desses é sinal de um pior atendimento hospitalar, o que pode levar a uma pior tomada de decisão e maior mortalidade. Dessa forma, sob essas suposições, foi incluída no modelo a nova feature em forma de flag, que indica se uma determinada linha do dataset sofreu com dados faltantes ou não.

5 - Tratamento dos dados

Como já dito anteriormente, os dados vieram em três tabelas separadas. A primeira tarefa foi, então, unir esses dados em uma única tabela.

Como um segundo passo, deve-se lembrar que cada linha era relativa aos dois bebês ao mesmo tempo. Como queremos realizar as predições sobre apenas um dos bebês, tivemos que dividir essas linhas existentes em duas, efetivamente dobrando a quantidade de entradas para cerca de 142690 entradas. Cada linha possuía *features* compartilhadas pelos bebês e outras poucas específicas para cada um deles. Dessa forma, bastou copiar os dados compartilhados para as duas novas linhas e definir os dados específicos de cada bebê nas novas linhas.

No primeiro momento do trabalho, foi analisada a quantidade de dados nulos totais por coluna. Das 52 *features* na tabela final, 37 possuíam dados faltantes. Dessas 37 colunas, a maioria possuía menos de 20 mil dados faltando. Dessa

forma, foi decidido remover as colunas que possuíam mais do que 20 mil dados nulos. Foram elas: frace (dad race); herpes (risk factor, Herpes); dfageq (octile age of father); orfath (dad hispanic); feduc6 (education category); alcohol (risk factor, alcohol use); drink5: (num of drinks /week, quantiled); tobacco (risk factor, tobacco use); cigar6 (num of cigarettes /day, quantiled). Dessa forma, sobram 43 *features* para o treinamento.

Para tratar as outras entradas com dados faltando, decidimos eliminar quaisquer entradas com mais de 5 *features* nulas. Isso reduziu a quantidade de dados para 119794. Para completar o resto de dados faltando, aplicamos a lógica de “valor mais comum” para cada *feature*.

5.1- Alternativas e adições ao tratamento de dados

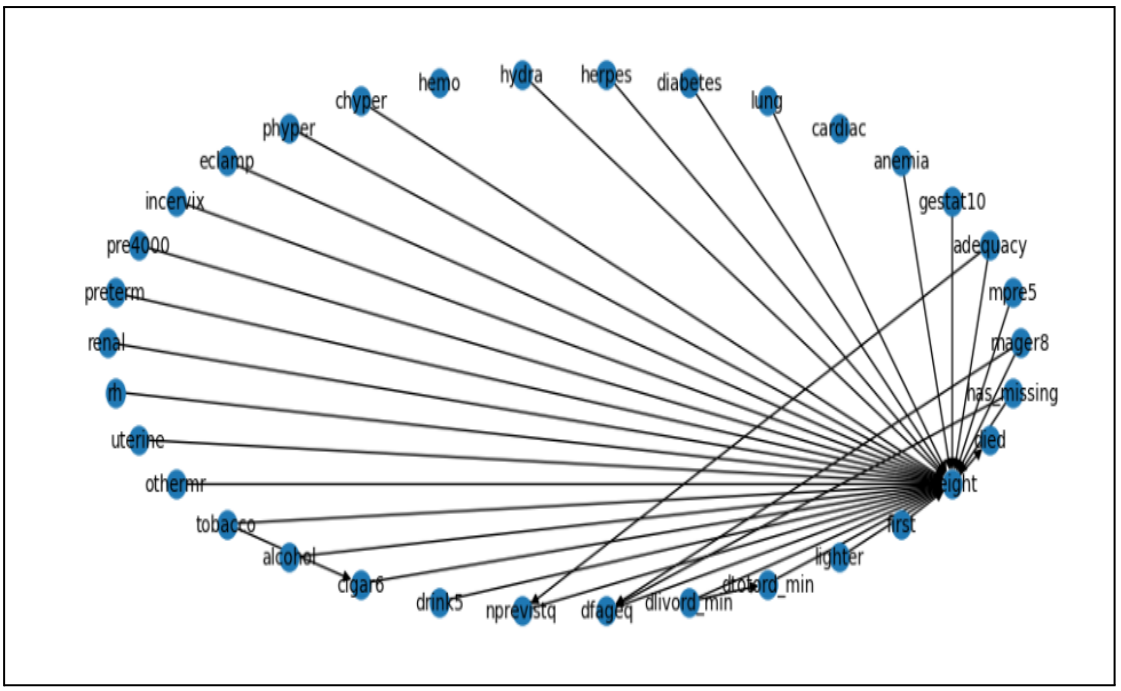
Após a apresentação parcial, recebemos como feedback uma possível alternativa para preencher os dados faltantes no dataset original. Usando a inferência causal, poderíamos recuperar valores tidos como NaN a partir das relações causais com as outras colunas que ainda tinham seus valores intactos.

Entretanto não se provou uma alternativa viável, uma vez que na maioria das vezes que um valor estava faltando, múltiplos outros também estavam. Por causa disso, os dados que ainda estavam intactos muitas vezes eram insuficientes e acabariam resultando em uma saída influenciada pelas correlações espúrias, algo que julgamos melhor evitar.

Sendo assim, tentamos uma outra alternativa, um meio termo entre a solução inicial e o approach causal. Ainda substituímos todos os NaN’s pelos valores mais comuns sem remover nenhuma coluna no dataset, porém, sempre que isso era feito em uma linha, adicionamos a ela uma flag em uma nova coluna do dataset chamada *has_missing*. Essa flag é feita com uma variável booleana que nos permite encontrar novas correlações e tirar novas conclusões preliminares sobre uma parcela do dataset, mesmo que os valores faltantes em si não adicionem novas informações ao modelo. Algumas dessas correlações e conclusões podem ser encontradas no item anterior do relatório.

5.2- Grafo Causal

Agora, com o tratamento de dados completo e uma nova feature criada, temos o necessário para gerar o Grafo Causal formal, feito a partir de bibliotecas de Causalidade (CausalNex[11]) e conhecimento do domínio. Ele conta com os diversos fatores de risco à saúde e foram removidas colunas que apenas adicionariam elementos espúrios, como o código postal do hospital, a ascendência dos pais do bebê e seus níveis de educação, por exemplo.



É possível perceber que quase todas as variáveis utilizadas para criar o grafo causal, menos as *features hemo* e *cardiac*, apontam para a *feature weight*. Além disso, a partir do conhecimento do domínio, também adicionamos a relação causal de *weight* para *died*.

6 - Aplicação de modelos

O algoritmo NICE precisa, primeiramente, de um modelo de machine learning que possa ser usado na classificação dos vetores híbridos para descobrir qual é o contrafactual da entrada alvo. Dessa forma, resolvemos testar 4 modelos diferentes utilizando da biblioteca scikit-learn[12]. Para tal, variamos os seus hiperparâmetros usando a técnica GridSearch com validação cruzada de 5-folds. No primeiro momento do trabalho, havíamos removido as colunas com mais de 20 mil dados faltando, o que fez com que sobrassem 42 *features*. Já no segundo momento, não removemos colunas seguindo essa lógica, mas sim, a sua utilidade aparente no grafo causal gerado, o que fez com que sobrassem 31 *features*.

Os modelos testados são:

- Modelo KNN: Apesar de considerar a dimensionalidade como razoavelmente alta, gostaríamos de ver se um modelo de agrupamento simples conseguiria performar bem. Para tal, de acordo com a documentação, o algoritmo 'ball-tree' deve performar melhor em quantidade de computações quando a dimensionalidade do dado é maior do que 20 comparado ao K-DTree.
- Modelo HistGradientBoostingClassifier: De acordo com a documentação, o sklearn possui essa nova implementação de gradient boosting trees para classificação. Esse modelo é inspirado no LightGBM e usa histogramas para dividir os dados. Além disso, na documentação é dito que ele poderia ser

ordens de magnitude melhor do que o GradientBoosting caso a quantidade de dados seja maior do que dezenas de milhares, que é o caso atual.
- Modelo Neural: Definimos o solver do modelo neural como 'adam', uma variação de stochastic gradient-based optimizer, pois, na documentação, está descrito como funcionar relativamente bem em datasets razoáveis, que, no caso, é de 119k entradas.
- Modelo RandomForest: O modelo RandomForest não possui nada fixo, os seus principais parâmetros são variados no GridSearchCV.

Os parâmetros variados foram:
- HistGradientBoostingClassifier: Por ser um modelo de boosting, os decision stumps do modelo podem/devem ser de baixíssima capacidade. Por padrão, o sklearn não define a profundidade máxima das árvores, portanto, variamos esse valor entre 1 e 20 da forma range(1, 20, 4). Além disso, variamos a quantidade máxima de decision stumps até 1000 da forma range(100,1000,200) pois os modelos de boosting são bem robustos. Também variamos o learning_rate como [0.001,0.01,0.1].
- KNN: As principais questões do modelo KNN são como computar a distância dos pontos e a quantidade de vizinhos a serem levados em consideração. Dessa forma, definimos a métrica de distância como a distância euclidiana e variamos a quantidade de vizinhos até 18 da forma [5, 9, 13, 18]. Um valor baixo de vizinhos (5) supõe que as classes de dados sejam bem definidas e um valor mais alto (18) supõe que eles estejam mais misturados/sejam ruidosos.
- Neural: No primeiro momento do trabalho, eram 42 features de entrada, já no segundo, eram 31. Dessa forma, em ambos os casos foram testadas quatro configurações se diferenciando apenas pela quantidade de nós na primeira camada. Na primeira versão eram 42 nós e, na segunda, eram 31.
    - 1 camada oculta: Possui uma camada oculta com 42 nós na primeira versão e 31 na segunda.
    - 2 camadas ocultas: A primeira camada oculta com 42/31 nós e a segunda com 20 nós
    - 3 camadas ocultas: A primeira camada oculta com 42/31 nós, a segunda com 20 e a terceira com 10
    - 4 camadas ocultas: A primeira camada oculta com 42/31 nós, a segunda com 20, a terceira com 10 e a quarta com 5

Por padrão, o early-stopping já é definido como False, mas, apesar disso, o treinamento é interrompido se o modelo não melhorar por 0.0001 durante 10 iterações. Considerando que temos 119k entradas com 5 folds, considerei que um número de iterações máximo de 25 mil talvez já fosse suficiente para o modelo convergir.

- RandomForest: Os principais parâmetros do RandomForest são a quantidade de árvores utilizadas, em que variamos da forma [30, 50, 70, 100], e a

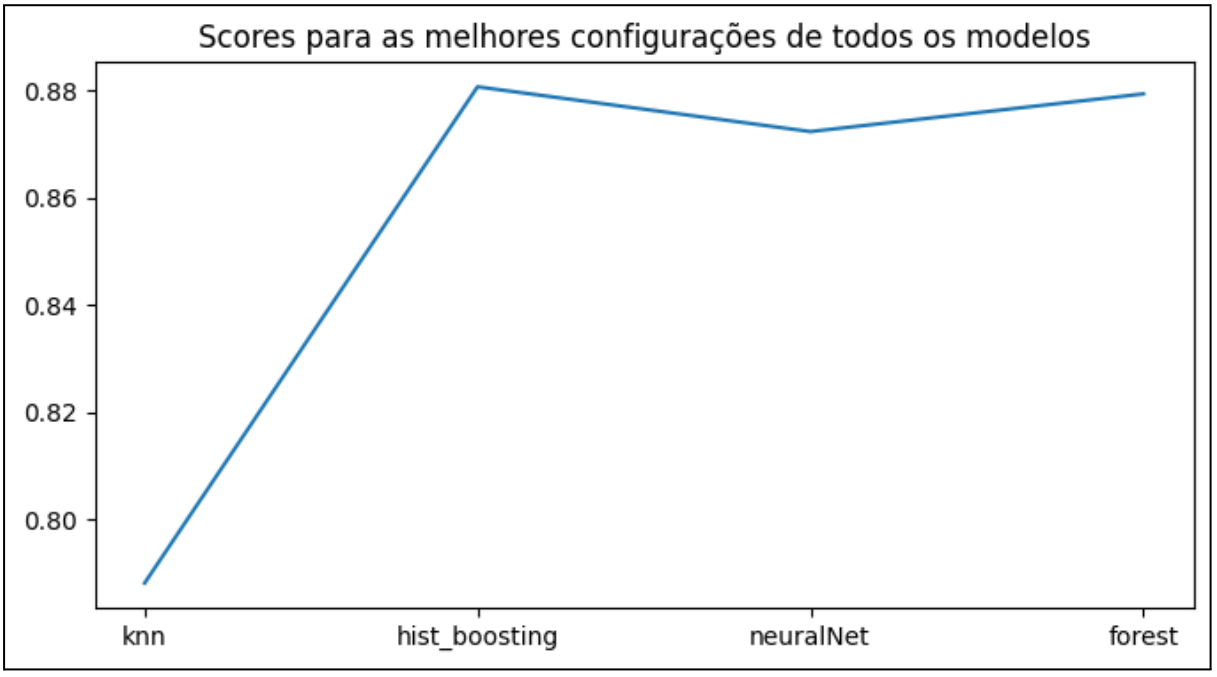
quantidade mínima de entradas para se tentar dividir os dados em que variamos da forma [2, 4, 8, 16].

Um dos problemas encontrados em ambos os momentos do trabalho foi a grande diferença de proporção de dados representando cada categoria, o problema da classe rara. Na primeira versão, tínhamos 115800 representando os bebês que não morreram e 3994 que não morreram. Já na segunda versão, representando todas as instâncias, tínhamos 137566 representando bebês que não morreram e 5124 para os que morreram.

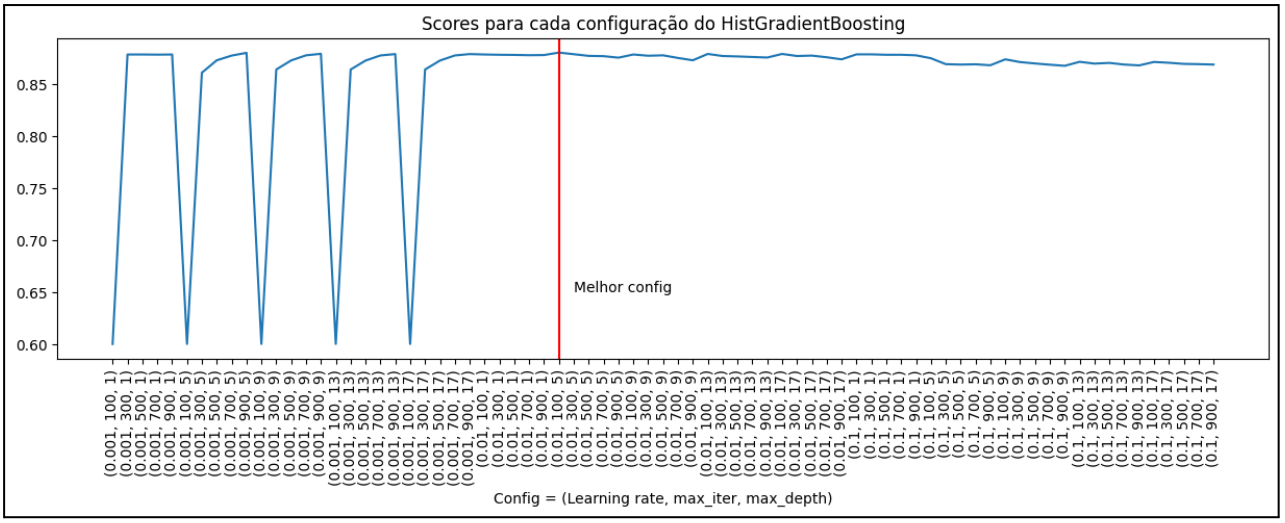
Dessa forma, foi necessário treinar em um número menor de dados para que eles ficassem balanceados no treino. Portanto, na primeira versão, o treino foi realizado em 4518 (60%) entradas representando os que não morreram e 3012 (40%) os que morreram. Já na segunda versão, essa proporção foi de 5691 (60%) e 3794 (40%) respectivamente. O restante dos dados será utilizado nos testes.

7 - Análise dos resultados

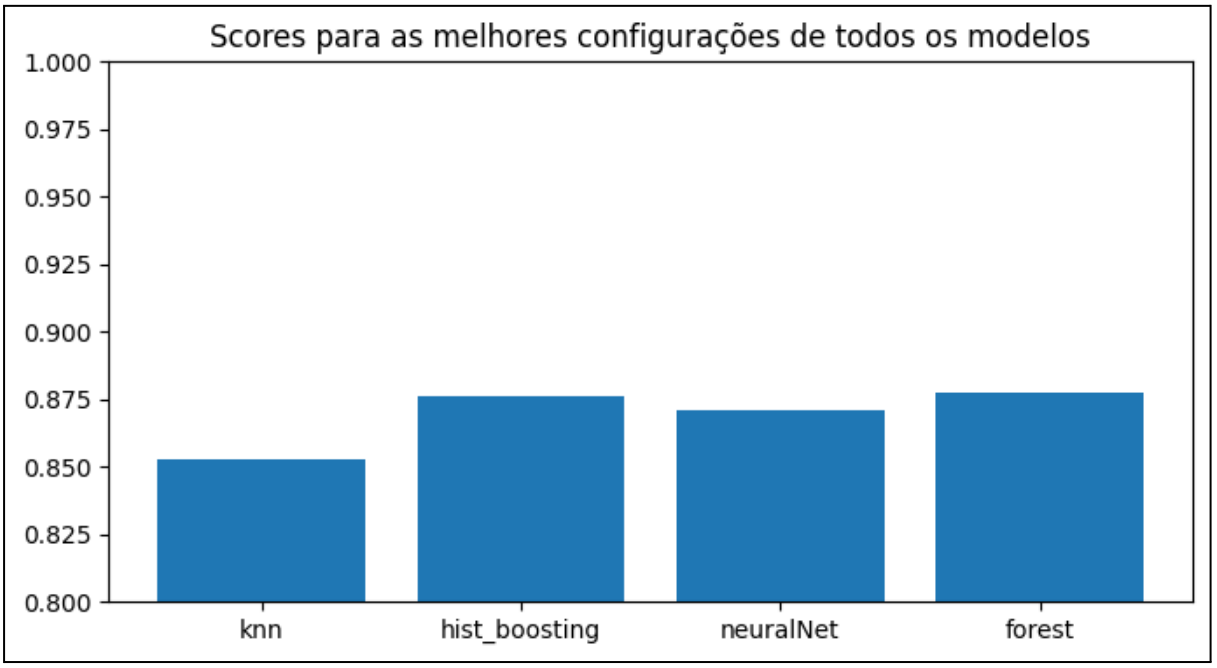
Na primeira versão, o resultado para a melhor configuração de todos os modelos pode ser vista na próxima imagem.



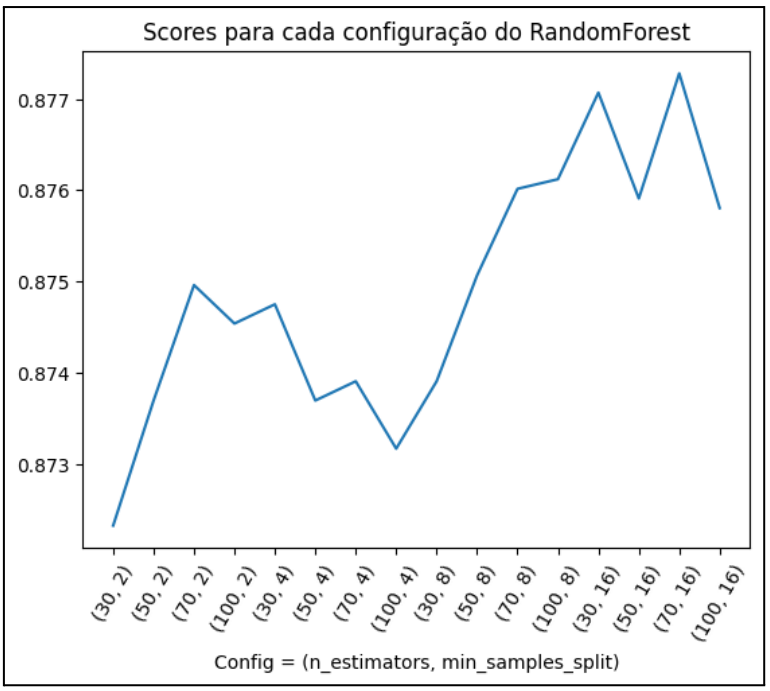
Assim, é possível ver que o melhor modelo encontrado foi o HistGradientBoostingClassifier com score de 0.88. Na próxima imagem é possível ver o score para cada configuração do HistGradientBoostingClassifier.



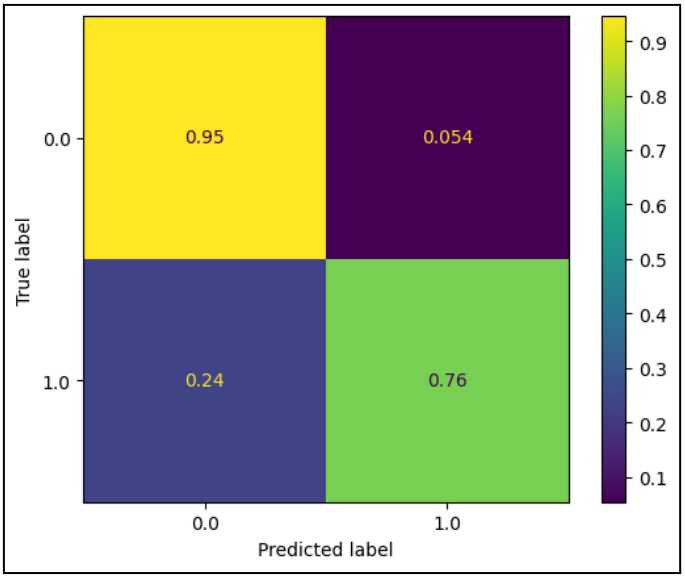
Já na segunda versão, os resultados foram:



Nessa versão, o melhor modelo foi o RandomForest com *score* de 0.877, ficando logo a frente do HistGradientBoosting com 0.876. A melhor configuração do RandomForest foi: 'criterion': 'entropy', 'min_samples_split': 16, 'n_estimators': 70. Na próxima imagem, é possível ver o *score* para cada configuração do RandomForest:

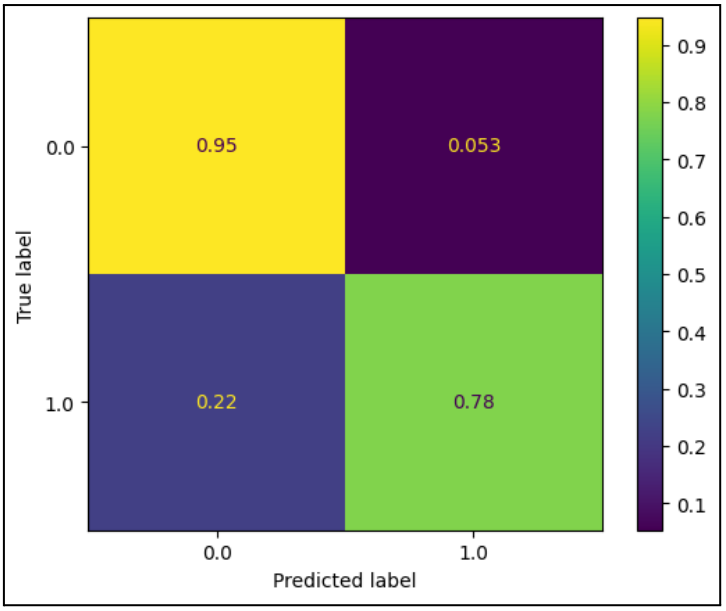


Já para os dados de teste, na primeira versão, temos a seguinte matriz de confusão:



Pode-se ver que o modelo é bem melhor em identificar bebês que não morreram do que morreram. Entretanto, isso já era esperado devido a grande diferença de proporção de dados já mencionados.

Na segunda versão, temos a seguinte matriz de confusão:



Pode-se perceber que, nessa versão, tivemos uma pequena melhora de 2% na identificação de bebês que vieram a óbito.

8 - Conclusão

Este trabalho tinha o objetivo de investigar a aplicação da causalidade para ajudar a prever se um bebê viria a óbito dadas as suas características. Para tal, primeiramente, era necessário desenvolver um modelo que conseguisse prever, satisfatoriamente, dadas as *features* do bebê, se ele viria a óbito ou não. Inicialmente, o objetivo era usar o modelo NICE para gerar exemplos contrafactuais do dataset para aumentar a explicabilidade dos fatos. No entanto, ficamos interessados na parte dos dados faltantes e tivemos alguns problemas com a biblioteca que iríamos utilizar, portanto resolvemos pivotar o rumo do trabalho na direção mencionada. Continuamos a testar o NICE independentemente, mas voltamos o foco para o uso dos NaN's, uma vez que essa área talvez tenha maior aplicabilidade em uma maior gama de modelos.

Utilizando de causalidade, com ajuda da biblioteca causalnex, foi possível chegar a um conjunto final de 31 *features* para serem levadas em consideração no modelo. Ao testar 4 modelos diferentes e esbarrar em questões como o problema da classe rara, chegamos a um modelo final que prevê, com acurácia média de 0.945, se um bebê virá a óbito ou não.

Com isso, esse modelo poderia ajudar a auxiliar a equipe médica na tomada de decisões para que, ao identificar o risco de o bebê vir a óbito, ajam para o objetivo final de a criança vir a crescer de forma saudável, diminuindo a mortalidade infantil.

Ainda conseguimos adotar um modelo de aumento de informação do dataset através do uso da flag de dados faltando, que se mostrou eficaz aumentando a precisão do modelo previamente pensado. Esse tipo de prática pode ser útil na

situação apresentada, em que as informações restantes podem não ser suficientes para inferimos causalmente os valores dos dados faltantes, além de ser um método mais simples. Além disso, ele pode ser útil em uma análise inicial para se ter uma ideia se a flag pode realmente se tornar uma feature, pois nem sempre ela trará informações significativas.

Link do github para o trabalho:
https://github.com/Pendulun/TWINSCausalidadeML

9 - Referências

[1]https://github.com/AMLab-Amsterdam/CEVAE/blob/master/datasets/TWINS/.

[2]Guo, R., Cheng, L., Li, J., Hahn, P. R., and Liu, H. A survey of learning causality with data: Problems and methods. 2018.

[3]Cheng, L., Guo, R., Moraffah, R., Candan, K. S., Raglin, A., and Liu, H. A practical data repository for causal learning with big data. In Benchmarking, Measuring, and Optimizing, Lecture notes in computer science, pp. 234–248. Springer International Publishing, Cham, 2020.

[4]Mfateneza, E., Rutayisire, P. C., Biracyaza, E., Musafiri, S., and Mpabuka, W. G. Application of machine learning methods for predicting infant mortality in rwanda: analysis of rwanda demographic health survey 2014-15 dataset. BMC Pregnancy Childbirth, 22(1):388, May 2022.

[5]Brahma, D. and Mukherjee, D. Using machine learning to target neonatal and infant mortality.https://www.ideasforindia.in/topics/macroeconomics/using-machine-learning-to-target-neonatal-and-infant-mortality.html, 2022. Acessado: 26-09-2022.

[6]Batista, A. F. M., Diniz, C. S. G., Bonilha, E. A., Kawachi, I., and Chiavegatto Filho, A. D. P. Neonatal mortality prediction with routinely collected data: a machine learning approach. BMC Pediatr., 21(1):322, July 2021.

[7]Sheikhtaheri, A., Zarkesh, M. R., Moradi, R., and Kermani, F. Prediction of neonatal deaths in NICUs: development and validation of machine learning models. BMC Med. Inform. Decis. Mak., 21(1):131, April 2021.

[8]Mangold, C., Zoretic, S., Thallapureddy, K., Moreira, A., Chorath, K., and Moreira, A. Machine learning models for predicting neonatal mortality: A systematic review. Neonatology, 118(4):394–405, July 2021.

[9]Feng, J., Lee, J., Vesoulis, Z. A., and Li, F. Predicting mortality risk for preterm infants using deep learning models with time-series vital sign data. NPJ Digit. Med., 4(1):108, July 2021.

[10]Brughmans, D., Leyman, P., and Martens, D. NICE: An algorithm for nearest instance counterfactual explanations. 2021.

[11]Beaumont, P., Horsburgh, B., Pilgerstorfer, P., Droth, A., Oentaryo, R., Ler, S., Nguyen, H., Ferreira, G. A., Patel, Z., & Leong, W. (2021). CausalNex [Computer software]. https://github.com/quantumblacklabs/causalnex

[12]Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 2011