Modelagem

Bruna Almeida Osti* e Fabio Fogliarini Brolesi†
Instituto de Computação, Universidade Estadual de Campinas - UNICAMP
Campinas, SP
Email: *b231024@dac.unicamp.br, †brolesi@gmail.com

***Resumo*—Este trabalho busca abordar a modelagem estatística do projeto da disciplina, mostrando possibilidades de uso dos modelos plug-in e *risk minimization*, bem como o uso de risco bayesiano para o fit e a calibração da curva a partir dos resultados obtidos.**

***Index Terms*—Trustworthy, AI, Ethical**

I. Introdução

Os modelos prognósticos usam vários fatores em combinação para prever o risco de resultados clínicos futuros em pacientes. Um bom modelo deve (i) fornecer previsões precisas que informam os pacientes e seus cuidadores, (ii) apoiar a pesquisa clínica e (iii) permitir decisões para melhorar os resultados dos tratamentos aos pacientes, segundo. Um modelo prognóstico tem três fases principais: desenvolvimento do modelo (incluindo validação interna), validação externa e investigações de impacto na prática clínica. Embora muitos modelos prognósticos sejam propostos, poucos são atualmente usados na prática, conforme mostra [1].

Neste trabalho pretende-se estruturar uma automação que seja capaz de predizer quais pacientes sairão da unidade de terapia intensiva dadas características presentes nos datasets fornecidos que tem relação com cirurgias ortopédicas e dados de estadia em unidades de terapia intensiva.

O estudo de [2] utilizou o modelo de regressão logística com máxima verossimilhança em pacientes internados para identificar se após a admissão para a terapia intensiva o paciente evoluiria a óbito em até 14 dias. Para este caso, utilizou-se curvas calibradas e score de Brier, que é uma medida de avaliação usada para verificar a qualidade de um valor de probabilidade previsto. É semelhante ao erro quadrático médio, mas aplicado apenas para scores de probabilidade de predição, cujos valores variam entre $[0, 1]$. A partir do modelo obtido, este foi testado em outro hospital para a validação.

De mesmo modo [3] utiliza de regressão logística para identificar pacientes com lesão que evoluirão a óbito a partir de critérios do *New Injury Severity Score* (NISS). o NISS, segundo o autor é um preditor melhor que outros levantados no estudo para o caso de óbito e análises de pacientes com problemas ortopédicos. Para resultado independentes, neste caso, os valores de *Injury Severity Score* ISS e NISS além de idade eram fatores que influenciavam no resultado do modelo final.

O estudo de [4] também se utiliza de regressão logística para análise de mortalidade em UTI, utilizando regularização L1, mas também utilizou o Least Absolute Shrinkage and Selection Operator (LASSO), regressão logística com uma regularização L2 e árvores de decisão de aumento de gradiente (gradient boosting). Para este caso, o gradient boosting foi o método mais adequado, uma vez que o interesse da pesquisa era utilizar as informações do modelo em tempo real.

Em comum, todos os estudos utilizaram métodos de aprendizado supervisionado. Segundo [5] existem duas abordagens para o aprendizado supervisionado:

- *plug-in*: estimar as funções de probabilidade e probabilidades anteriores do treinamento dos dados e conectá-los ao teste de razão de verossimilhança de Bayes ou
- *risk minimization*: otimizar um classificador sobre uma aproximação empírica da taxa de erro calculada nas amostras de dados de treinamento.

II. Pré processamento

Para o pré processamento, utilizamos os conjuntos de dados `patients`, `surgeries` e `icu` presentes na pasta do Google Drive da disciplina[1]. Realizamos a limpeza, em termos do dataset de cirurgia, de dados como CID, tipo de cirurgia, fila e também um tratamento para segregar as alas dos pacientes, ignorando localização exata (sala); para internações, tratamos dados de morte e de doação de órgãos e tecidos como morte, caso contrário, não morte. O caso positivo (*true values*) foi definido então como paciente em óbito na unidade de tratamento intensivo. Para os outros dados tanto pacientes, quando cirurgias quanto internações, aplicou-se tipos de data e hora, categoria e texto, conforme a semântica da coluna dentro do contexto.

Com relação a feature engineering, foram tratados os tipos de dados de acordo com a natureza deles (data, hora, categoria, numérico, e outros).

Para o presente trabalho, utilizou-se das seguintes features:

- `patients_STAY DURATION`: tempo de estadia entre entrada e saída do paciente
- `patients_ICU DURATION`: tempo de estadia na unidade de terapia intensiva
- `icu_CID`: CID identificada no conjunto de dados de unidade de terapia intensiva (apenas o prefixo neste caso)
- `patients_ORIGIN UNIT`: unidade de origem do paciente
- `patients_DESTINATION UNIT`: unidade de destino do paciente

[1] https://drive.google.com/drive/folders/1M9k_FwADLcD_fl8yMjwmwQuXgwVhWGI3

- `icu_IS DEAD`: identificação de óbito ou não do paciente (1 para óbito, 0 para não óbito)

Como resultado, identificou-se na *feature* `REASON FOR DISCHARGE` do dataset `ICU` para marcar como 1 para casos de óbito ou doação de órgãos após o óbito, e 0 caso contrário.

Para a integração dos datasets, utilizou-se campos de **nome**, **data de admissão** e **data de liberação**, além de **data da cirurgia** e **data de primeira cirurgia**.

Os campos identificados como sensíveis para o dataset resultante são:

- `surgeries.VALUE`: valor gasto na cirurgia
- `icu.AGE`: idade do paciente
- `icu.SEX`: sexo do paciente

Para as features `icu_CID`, `patients_ORIGIN UNIT`, `patients_DESTINATION UNIT`, `icu.SEX` , aplicamos o método `get_dummies` que transforma os dados categóricos em 1 e 0 aumentando a dimensionalidade para os seguintes campos:

III. Falsos positivos e falsos negativos

Para o presente trabalho, entendendo o contexto médico, entende-se que os falsos negativos (não dizer que o paciente irá evoluir a óbito quando, de fato, ele irá), são mais danosos que os falsos positivos.

Desta forma, será dado pesos maior aos falsos negativos, uma vez que esse peso é uma punição pelo erro. Para isso, colocou-se pesos para ambos como segue:

- falsos positivos: 700
- falsos negativos: 10000

Os valores foram colocados a partir da análise de parâmetros e resultados identificados após execuções de funções que retornaram o resultado mais adequado e definidos afim de fazer com que houvesse um ajuste considerando a necessidade de pesos maiores para falsos negativos.

IV. Métricas de avaliação

A métrica de avaliação escolhida para nosso problema é a sensibilidade (recall), essa métrica avalia a capacidade do método de detectar a proporção de positivos classificados corretamente. Ela pode ser obtida pela equação:

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

No qual, TP (Verdadeiro Positivo) e FN (Falso Negativo).

V. Modelos

A. Modelos Plug-in

1) Linear Discriminant Analysis (LDA): Conforme [6], o LDA é um classificador paramétrico com um limite de decisão linear, como o próprio nome sugere, que é gerado ajustando densidades condicionais de classe aos dados e usando a regra de Bayes. O modelo ajusta uma densidade gaussiana para cada classe, assumindo que todas as classes compartilham a mesma matriz de covariância. Além disso, pode ser usado para redução de dimensionalidade, projetando os dados de entrada para um subespaço linear que consiste nas direções que maximizam a separação entre as classes.

Segundo [7], o LDA derivado de modelos probabilísticos mais simples que modelam a distribuição condicional da classe de dados $P(X|y = k)$ para cada k. Podemos utilizar a regra de Bayes, para cada amostra de treinamento e selecionamos a classe k que maximiza essa probabilidade posterior. Em específico para o LDA assume-se que as gaussianas para cada classe compartilhem a mesma matriz de covariância: $\sum_k = \sum$ para todo k. Isso reduz o log para:

$$\log P(y = k|x) = -\frac{1}{2}(x-\mu_k)^t \sum^{-1}(x-\mu_k) + \log P(y = k) + Cst \tag{2}$$

2) K-Nearest Neighbors (KNNs): O K-Nearest Neighbors é um método de aprendizado supervisionado não paramétrico, usando aprendizado não generalizante, ou seja, simplesmente armazena instância dos dados de treinamento. A classificação é calculada através de votação majoritária simples dos vizinhos mais próximos de cada ponto, portanto é atribuído a classe de dados que tem mais representantes dentro dos vizinhos mais próximos do ponto.

Segundo [6], uma métrica de distância é necessária para medir se os pontos estão próximos ou afastados do ponto, geralmente é utilizada a distância Euclidiana, entretanto, outros tipos de distância podem ser melhores para problemas específicos.

Por fim, a entrada x é atribuída a classe com a maior probabilidade (equação 3), no qual K é o número de componentes, e i são cada uma das amostras. A valor ótimo do k é altamente dependente dos dados, em geral um k maior diminui os efeitos do ruído, mas torna os limites da classificação menos distintos. Além disso, o algoritmo usa pesos uniformes, apesar de permitir utilizar ponderação.

$$P(y = j|X = x) = \frac{1}{K}\sum_{i \in A} I(y_i = j) \tag{3}$$

Esse método funciona melhor que outros classificadores quando não há muita sobreposição entre as classes e a decisão é dividida entre muitos componentes.

B. Modelos Risk Minimization

1) Decision Trees: Conforme indica [8], as árvores de decisão são usadas para classificação e regressão, através da aprendizagem de regras de decisão simples que são inferidas através dos dados. Quanto mais profunda a árvore, mais complexas são as regras de decisão e mais adequado o modelo.

As árvores são muito utilizadas pela sua facilidade de interpretação, além disso é possível validar um modelo usando testes estatísticos, isso torna possível explicar a confiabilidade do modelo. Entretanto, elas podem ser instáveis pois pequenas variações nos dados podem resultar na geração de uma

árvore completamente diferente, sendo necessário o uso de um ensemble para mitigar o problema. Além disso, elas não são boas em extrapolação, visto que as predições são aproximações constantes por partes.

Segundo [7], a árvore de decisão particiona recursivamente o espaço de features de forma que as amostras com os mesmos rótulos sejam agrupadas.

A qualidade do candidato é computada usando a função de impureza (função de loss) H, a escolha depende do tipo de task (classificação ou regressão). A medida mais comum utilizada para medir a impureza é a Gini, mas também podemos utilizar a função de entropia.

2) Neural Networks (MLP): O multi-layer perceptron é um método de aprendizado supervisionado que aprende uma função não linear que descreve os dados. As features de entrada são representadas por $x = [x_1, x_2, x_3, ..., x_n]$, que são multiplicados pelos respectivos pesos sinápticos, que são elementos do vetor $w = [w_1, w_2, w_3, ..., w_n]$, gerando o valor z, que é denominado como potencial de ativação, e o termo b que não é afetado pela entrada e corresponde ao "bias", como descrito na equação 4.

$$z = \sum_{i=l}^{N} x_i W_i + b \tag{4}$$

Esse valor z passa por uma função responsável por limitar tal valor a um dado intervalo (função de ativação), produzindo o valor final da saída do y neurônio. Os valores de saída da camada inicial estão ligados às entradas das camadas seguintes, consequentemente até a rede atingir a ultima camada fornecendo como resultado de saída, quanto mais profunda a rede, mais capaz de aprender relações complexas o modelo é.

O MLP treina seus pesos usando o Stochastic Gradient Descent (SGD) [9] e atualiza os parâmetros usando o gradiente da perda em relação ao parâmetro que precisa de adaptação [10], [11]. No qual, tem uma taxa de aprendizado que controla a busca no espaço de parâmetros e uma a função de perda usada como avaliação para a rede.

VI. Resultados

Para encontrar-se a melhor versão de cada modelo foi utilizado o Grid Search para busca exaustiva dos melhores parâmetros, combinado com a validação cruzada de 10 folds que previne o modelo de alcançar o indesejável overfitting, isso porque é um método de re-amostragem que usa diferentes partes dos dados para avaliar e treinar um modelo em diferentes iterações.

Os parâmetros levados em consideração para cada modelo foram:

Linear Discriminant Analysis:

- solver: O solucionador para otimização de peso.
  - svd: Decomposição de valor singular (padrão). Não calcula a matriz de covariância, portanto este solver é recomendado para dados com um grande número de características.
  - lsqr: Solução de mínimos quadrados. Pode ser combinado com o estimador de encolhimento ou de covariância personalizado.
  - eigen: Decomposição de valores próprios. Pode ser combinado com o estimador de encolhimento ou de covariância personalizado.

K-nearest neighbors:

- n_neighbors: quantidade de vizinhos considerados
- weights:
  - uniform: pesos uniformes. Todos os pontos em cada vizinhança são ponderados igualmente.
  - distance: pontos de peso pelo inverso de sua distância. neste caso, os vizinhos mais próximos de um ponto de consulta terão uma influência maior do que os vizinhos mais distantes.
- p: Quando p = 1, isso é equivalente a usar distância_manhattan (l1) e distância_euclidiana (l2) para p = 2. Para p arbitrário, é usado distância_minkowski (l_p).

Decision Tree:

- criterion: Função para medir a qualidade da separação, exemplo: gini, entropia.
- max_depth:A profundidade máxima da árvore.

MLP:

- hidden_layer_sizes: O i-ésimo elemento representa o número de neurônios na i-ésima camada oculta
- activation: Função de ativação para a camada oculta, exemplo: tanh (tangente hiperbólica), relu (unidade linear retificada)
- solver: O solucionador para otimização de peso, exemplo: sgd, adam.
- alpha: Força do prazo de regularização L2. O termo de regularização L2 é dividido pelo tamanho da amostra quando somado à perda.
- learning_rate: Cronograma de taxa de aprendizado para atualizações de peso, exemplo: constant, adaptive.

Os parâmetros levados em conta para cada modelo, a melhor configuração encontrada e os resultados no dataset de validação estão descritos na Tabela I.

Para a fase de teste foram utilizados três diferentes limiares para comparação: 1) Limiar padrão de 0.5, 2) Limiar levando em consideração que a quantidade de positivos no teste, seria a mesma quantidade de positivos da predição, 3) Limiar levando em conta o Risco Baeysiano. Gerando os respectivos resultados contidos na Tabela II.

Comparando-se os modelos (LDA, KNN, Decision Trees e MLP) para os diferentes limiares, podemos observar que o melhor modelo obtido é o Decision Tree, apesar de o modelo

Nome do modelo	Parâmetros	Melhor Modelo	Recall (\pm std)
Linear Discriminant Analysis	{"solver": ["svd", "lsqr", "eigen"]}	{'solver': 'svd'}	0.819 (± 0.050)
K-nearest neighbors (KNNs)	{"n_neighbors":[1,2,..., 20], "weights": ["uniform", "distance"], "p": [1,2,3]}	{'n_neighbors': 3, 'weights': 'uniform', 'p': 1}	0.816 (± 0.036)
Decision Tree	{'criterion':['gini','entropy'], 'max_depth': [3,4...,15]}	{'criterion': 'gini', 'max_depth': 4}	0.867 (± 0.048)
MLP	param_grid_mlp = { 'hidden_layer_sizes': [(50,50,50), (100,50,100), (100,)], 'activation': ['tanh', 'relu'], 'solver': ['sgd', 'adam'], 'alpha': [0.0001, 0.05], 'learning_rate': ['constant','adaptive']}	{'hidden_layer_sizes': (50, 50, 50), 'activation': 'relu', 'solver': 'adam','alpha': 0.05, 'learning_rate': 'adaptive'}	0.843 (± 0.286)

Tabela I
MELHORES MODELOS ENCONTRADOS PELA BUSCA EXAUSTIVA DE PARÂMETROS (GRIDSEARCH) COM 10-FOLDS

Nome do modelo	Padrão (0.5)	Mesma quantidade de positivos (threshold)	Risco Bayesiano (threshold)	Matriz de confusão
linear discriminant analysis	**0.83**	0.40 (t = 0.9445)	0.50 (t = 0.9049 671127239737)	[[4685 1015] [70 349]]
KNNs	**0.84**	None	0.59 (t= 0.9049 671127239737)	[[4464 1236] [69 350]]
Decision Tree	**0.88**	None	0.45 (t=0.9049 671127239737)	[[4421, 1279] [51, 368]]
MLPs	**1.00**	None	**1.00 (t=0.9049 671127239737)**	[[2, 5698] [0, 419]]

Tabela II
MÉTRICAS PARA CADA UM DOS MODELOS SELECIONADOS

MLP ter obtido a métrica de recall maior. Isso porque podemos visualizar pela matriz de confusão que ele acertou todas as amostras positivas, entretanto, errou a maioria das amostras negativas. Portanto, está prevendo a maioria das amostras como a classe positiva de forma equivocada.

Por outro lado, o modelo Decision Tree além de obter recall de 88% também acertou consideravelmente a classe negativa, tendo uma performance interessante. Além disso, para os modelos KNNs, Decision Tree e MLPs não foi possível obter um valor de limitar que obtivesse a quantidade de predições positivas igual a quantidade original de amostras positivas das amostras de teste originais ou entre uma borda maior de 10 amostras.

VII. CALIBRAÇÃO

A calibração, conforme [7], permite ajustar melhor as probabilidades de um determinado modelo ou adicionar suporte para previsão de probabilidade.

Classificadores bem calibrados são classificadores probabilísticos para os quais a saída do método que retorna estimativas de probabilidade para cada classe pode ser interpretada diretamente como um nível de confiança. Por exemplo, um classificador binário bem calibrado deve classificar as amostras de tal forma que entre as amostras para as quais retornou um valor que retorna estimativas de probabilidade para cada classe próximo a $0,8$, aproximadamente 80% realmente pertencem à classe positiva.

As curvas de calibração (também conhecidas como diagramas de confiabilidade) comparam quão bem as previsões probabilísticas de um classificador binário são calibradas. Ele plota a frequência real do rótulo positivo em relação à probabilidade prevista, para previsões agrupadas. O eixo x representa a probabilidade média prevista em cada bin. O eixo y é a fração de positivos , ou seja, a proporção de amostras cuja classe é a classe positiva (em cada bin).

Calibrar um classificador consiste em ajustar um regressor (chamado de calibrador) que mapeia a saída do classificador para uma probabilidade calibrada em $[0, 1]$. Denotando a saída do classificador para uma determinada amostra por f_i, o calibrador tenta prever $p(y_i = 1|f_i)$ [2].

As amostras usadas para ajustar o calibrador não devem ser as mesmas amostras usadas para ajustar o classificador, pois isso introduziria viés. Isso ocorre porque o desempenho do classificador em seus dados de treinamento seria melhor do que para dados novos. Usar a saída do classificador de dados de treinamento para ajustar o calibrador resultaria em um calibrador tendencioso que mapeia para probabilidades mais próximas de 0 e 1 do que deveria.

Para a modelagem, realizou-se a calibração do modelo Linear Discriminant Analysis utilizando regressão logística e foram colocados pesos para as classes, com classe 0 com peso $0,03$ e classe 1 com peso $0,97$. O resultado dos dados com classes não calibrado estão em 1 e o resultado da calibração está presente na figura 2. Nota-se que com os pesos associados, a curva calibrada fica próxima à curva ideal de calibração, com pequenos desvios próximos de 0 e próximos de 1.

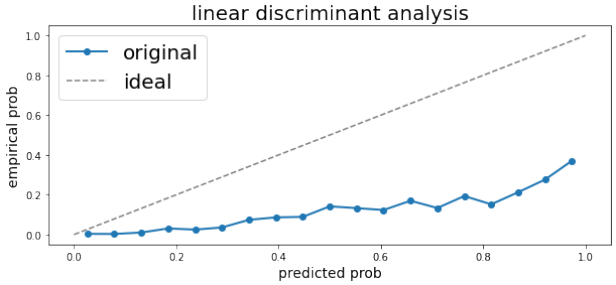

Figura 1. Valores originais dos retornos de probabilidade de classes para o problema

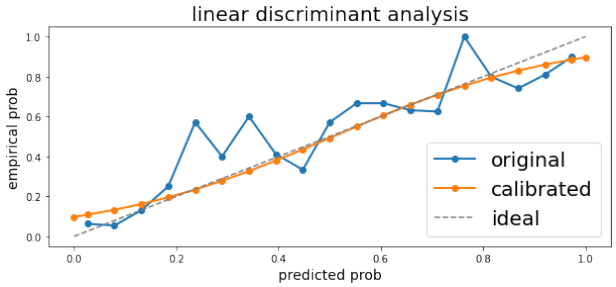

Figura 2. Valores calibrados a partir de regressão logística dos retornos de probabilidade de classes para o problema

REFERÊNCIAS

[1] J. C. Patino, Cecilia Maria e Ferreira, "Prognostic studies for health care decision making," 2019.

[2] https://scikit-learn.org/stable/modules/calibration.html

[2] Y. Allenbach, D. Saadoun, G. Maalouf, M. Vieira, A. Hellio, J. Boddaert, H. Gros, J. E. Salem, M. R. Rigon, C. Menyssa, L. Biard, O. Benveniste, and P. C. and, "Development of a multivariate prediction model of intensive care unit transfer or death: A french prospective cohort study of hospitalized COVID-19 patients," *PLOS ONE*, vol. 15, p. e0240711, Oct. 2020.

[3] Z. J. Balogh, E. Varga, J. Tomka, G. Süveges, L. Tóth, and J. A. Simonka, "The new injury severity score is a better predictor of extended hospitalization and intensive care unit admission than the injury severity score in patients with multiple orthopaedic injuries," *Journal of orthopaedic trauma*, vol. 17, no. 7, pp. 508–512, 2003.

[4] A. E. Johnson and R. G. Mark, "Real-time mortality prediction in the intensive care unit," in *AMIA Annual Symposium Proceedings*, vol. 2017, p. 994, American Medical Informatics Association, 2017.

[5] K. R. Varshney, *Trustworthy Machine Learning*. Chappaqua, NY, USA: Independently Published, 2022.

[6] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[9] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*, pp. 421–436, Springer, 2012.

[10] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48, Springer, 2012.

[11] A. Ng, J. Ngiam, C. Y. Foo, and Y. Mai, "Deep learning," *CS229 Lecture Notes*, pp. 1–30, 2014.