Previsão de tempo de uso de leitos hospitalares com Aprendizado de Máquina

Larissa Magistrali, Nikolas Lacerda e Andrea Aparecida Konzen
Escola Politécnica - Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre - RS - Brasil
{larissa.magistrali, nikolas.santos}@edu.pucrs.br, andrea.konzen@pucrs.br

***Resumo*—The length of hospital stays of patients is a fundamental factor for the planning and effective management of hospital resources. With the advancement of machine learning and the increasing availability of data in healthcare, there is an opportunity and an important interest in predicting the length of hospital stays of patients in order to improve the quality of patient care, decrease costs operations, increase service efficiency and help with discharge planning. In this work, a proposal for a customized model will be presented to determine the length of hospital stays.**

I. Introdução

Os leitos hospitalares são um dos recursos essenciais em todos os hospitais, e leitos geralmente são usados como um importante indicador de qualidade do serviço hospitalar [1]. Devido ao seu número limitado nos hospitais, o tempo de permanência de pacientes está diretamente relacionado ao custo financeiro da internação [2]. Logo, encurtar o tempo de ocupação pode não apenas aumentar a rotatividade de pacientes internados, mas também reduzir o custo dos recursos. Melhorar a alocação de leitos e amenizar sua escassez são grandes problemas enfrentados pelos hospitais, mas que em tempos pandêmicos se tornaram ainda mais importante de serem resolvidos. Segundo o relatório da OMS (Organização Mundial da Saúde), o tempo de internação de um paciente é considerado um dos mais importantes fatores de monitoramento e desempenho nos hospitais [3].

O tempo de internação de um paciente é definido em número de dias desde a data inicial da internação até a data em que o paciente recebe alta hospitalar [4]. O tempo de internação é uma variável complexa e pode haver uma variação significativa em diferentes contextos sociais, instalações, condições e especialidades de doenças, mesmo dentro do mesmo sistema hospitalar [5], dificultando o acompanhamento do fluxo de pacientes. A incerteza no tempo de permanência hospitalar do paciente é um grande impedimento para um agendamento eficaz.

O presente trabalho tem como objetivo implementar algoritmos de Aprendizado de Máquina para predizer o tempo de internação de pacientes.

Este trabalho está organizado na seguinte ordem: Na Seção II apresenta-se os principais trabalhos relacionados. Seção III, metodologia de trabalho e etapas do desenvolvimento. E por fim, na Seção IV apresenta-se os resultados obtidos e as contribuições com o algoritmo implementado.

II. Trabalhos relacionados

Estudos sobre o uso da tecnologia e seus algoritmos de Aprendizado de Máquina na previsão o tempo de internação do paciente em hospitais têm sido cada vez mais abordados na literatura. Nesta seção discute-se quatro relevantes pesquisas relacionadas ao problema descrito acima.

- No trabalho de M. Rachda Naila [6], foram aplicados os algoritmos de *Random Forest* e *Gradient Boosting*, com o objetivo de predizer o número de dias de internação do paciente. No estudo é utilizado uma base de dados disponibilizada pela Microsoft com os dados da internação de pacientes. Após a etapa de avaliação, o algoritmo de *Random Forest* teve um melhor desempenho.
- Um estudo de S. G. Gurazada [7], aplica algoritmos de Aprendizado de Máquina para previsão do tempo de permanência de pacientes nos departamentos de emergência australianos. Nesta análise são utilizados os algoritmos de *Random Forest, Decision Tree, Zeror, Naive Bayes, K-Nearest Neighbor* (KNN) e *Logistic Regression* para prever se um paciente será internado por mais ou por menos de 4 horas. Para a aplicação dos algoritmos foi utilizada uma ferramenta chamada *Weka Software*. Na análise de performance, é demonstrado que os modelos *K-Nearest Neighbor* e *Random Forest* tiveram um desempenho melhor na base de dados. Os autores também buscam explicar o modelo fazendo uma análise dos atributos. Os atributos *idade*, *categoria de triagem* e *modo de chegada* tiveram um impacto maior na previsão do modelo.
- Aplicando técnicas de Aprendizado de Máquina em uma base de dados, [8] também realiza um estudo para prever o tempo de internação do paciente. Neste, o objetivo é prever se o paciente ficará internado menos que 3 dias, entre 3 e 5 dias ou mais que 6 dias. Para isso, são utilizados os modelos *Bayesian classification, K-Nearest Neighbor* e *Decision Tree*, sendo *Decision Tree* o modelo com melhor desempenho. Também há uma explicação do modelo neste trabalho, onde foram identificadas as variáveis tipo de *cirurgia*, *número médio de visitas por dia* e o *número de consultas médicas* como as variáveis mais relevantes para predição do modelo.
- Com foco no algoritmo de *XGBoost*, Yong Chen [9] aplica esse modelo para prever o tempo de internação

do paciente. Neste, o objetivo é comparar o algoritmo de *XGBoost* com outros, como *Decision Tree, K-Nearest Neighbor e Support vector machine* (SVM), e avaliar como o XGBoost se comporta em comparação a estes. Assim como os estudos anteriores, também há uma busca para explicar o modelo neste trabalho, onde foram identificadas as variáveis *número de operações*, *status de transferência* e *idade* como as variáveis mais relevantes para predição do modelo.

A partir do estudo dos trabalhos relacionados, nota-se que entre os algoritmos de Aprendizado de Máquina utilizados nestes estudos, destaca-se o uso de algoritmos de aprendizado supervisionado, dos quais *Decision Tree* [10], *Random Forest* [11] e *K-Nearest Neighbor* [12] são os algoritmos mais frequentes.

III. Metodologia e Desenvolvimento

Neste capítulo são abordadas as diferentes etapas realizadas para a resolução do problema proposto, como a apresentação e construção da base de dados usada, pré-processamento dos dados, divisão da base em base de teste e de treinamento, algoritmos de treinamento e análise dos resultados, como é mostrado no diagrama na Figura 1 e disponibilizado no GitHub [13].

A. Base de dados

Neste trabalho foi utilizada a base de dados Brateca (Brazilian Tertiary Care Dataset) [14]. Brateca é uma base de dados clínicos de hospitais brasileiros, hospedada na plataforma de compartilhamento de dados da área da saúde Physionet [15] para pesquisadores mediante realização de curso sobre o uso ético dos dados, que foi anonimizada para garantir a segurança e privacidade dos envolvidos e disponibilizada de forma gratuita para ser usada por outras pesquisas. Esta base de dados reúne uma grande quantidade de dados estruturados na língua portuguesa sobre internações de pacientes, notas clínicas, descrições de pacientes, prescrições e exames realizados em 10 diferentes hospitais do Brasil. Por serem dados que retratam a realidade da saúde brasileira, as pesquisas realizadas com eles podem ter um grande impacto positivo na saúde do país que retrata. O Brateca é composto por 5 conjuntos de dados:

- Admissão - conjunto de dados de cada admissão individual do paciente.
- Exame - conjunto de dados de exames e seus respectivos resultados em cada admissão.
- Prescrição - conjunto de dados de cabeçalhos de prescrição, que inclui informações do paciente, avaliações farmacêuticas, data da prescrição, data de validade, informações da enfermaria e se a prescrição inclui medicamentos especiais.
- Item de prescrição - conjunto de dados de medicamentos prescritos, como nome, dosagem e informações sobre como o medicamento deve ser administrado.
- Nota clínica - conjunto de dados de notas clínicas em texto livre sobre detalhes da estadia e do tratamento do paciente.

Os conjuntos de dados de *Item de prescrição* e *Notas clínicas* são conjuntos de dados que possuem dados não estruturados, logo não serão utilizados neste trabalho, sendo possível explora-los em pesquisas futuras.

O conjunto de exames, possui dados de inúmeros exames nos quais muitos aparecem poucas vezes, dessa forma, foram selecionados os cinco exames mais comuns, pelo fato de haver dados suficientes para o treinamento do modelo, sendo estes os exames de Leucócitos, Plaquetas, Creatinina, Eosinófilos e Potássio.

Um outro tratamento realizado, foi a filtragem de apenas a primeira consulta de cada paciente, pois como o objetivo é prever o tempo de internação, não faz sentido usarmos dados depois do primeiro dia de internação.

Por fim, após a concatenação dos conjuntos de dados, o conjunto final ficou com 28 atributos, 18 destes categóricos e 9 numéricos, conforme descrito na Tabela 1.

Atributo	Tipo de dado
Admission_ID	ID da Consulta
Allergy	Numerical (0–2), mean ± SD (0.001 ± 0.0427)
Alerts	Numerical (0–35), mean ± SD (1.05 ± 1.93)
Prescription_Score	Numerical (0–52), mean ± SD (1.66 ± 2.77)
Score_One	Numerical (0–6), mean ± SD (0.23 ± 0.50)
Score_Two	Numerical (0–5), mean ± SD (0.21 ± 0.51)
Antibiotics	Numerical (0–29), mean ± SD (1.79 ± 3.57)
High_Alert	Numerical (0–16), mean ± SD (1.14 ± 1.46)
Controlled	Numerical (0–14), mean ± SD (0.88 ± 1.11)
Not_Default	Numerical (0–0), mean ± SD (0.03 ± 0.21)
Tube	Numerical (0-2), mean ± SD (0.002 ± 0.057)
Different_Drugs	Numerical (0–37), mean ± SD (5.50 ± 4.54)
Interventions	Numerical (0–12), mean ± SD (0.040 ± 0.338)
Complications	Numerical (0–114), mean ± SD (0.47 ± 2.75)
Leucócitos	Numerical (0–486550), mean ± SD (9356 ± 5354)
Plaquetas	Numerical (0–2.33), mean ± SD (2.00 ± 9.59)
Creatinina	Numerical (0.01–126.6), mean ± SD (1.019 ± 1.113)
Eosinófilos	Numerical (0–486550), mean ± SD (9356 ± 5354)
Potássio	Numerical (15–94), mean ± SD (4.30 ± 0.43)
Public	Categorical (0, no; 1, yes)
Surgical	Categorical (0, no; 1, yes)
IC	Categorical (0, no; 1, yes)
Obstetrics	Categorical (0, no; 1, yes)
Emergency	Categorical (0, no; 1, yes)
Ambulatory	Categorical (0, no; 1, yes)
COVID_19	Categorical (0, no; 1, yes)
Pharmacy_Assessment	Categorical (0, no; 1, yes)
Sex	Categorical (F, female; M, male)

Tabela I
Características da base de dados (72081 instâncias na base)

B. Pré-processamento

O pré-processamento de dados é a primeira etapa a ser realizada no processo de Aprendizado de Máquina. E esta consiste principalmente na organização e estruturação dos dados que serão utilizados pelo algoritmo. Para este trabalho, o pré-processamento do Brateca se resume a eliminação de fatores em variáveis categóricas, eliminação de erros de introdução, eliminação de valores nulos e outliers e normalização dos valores numéricos.

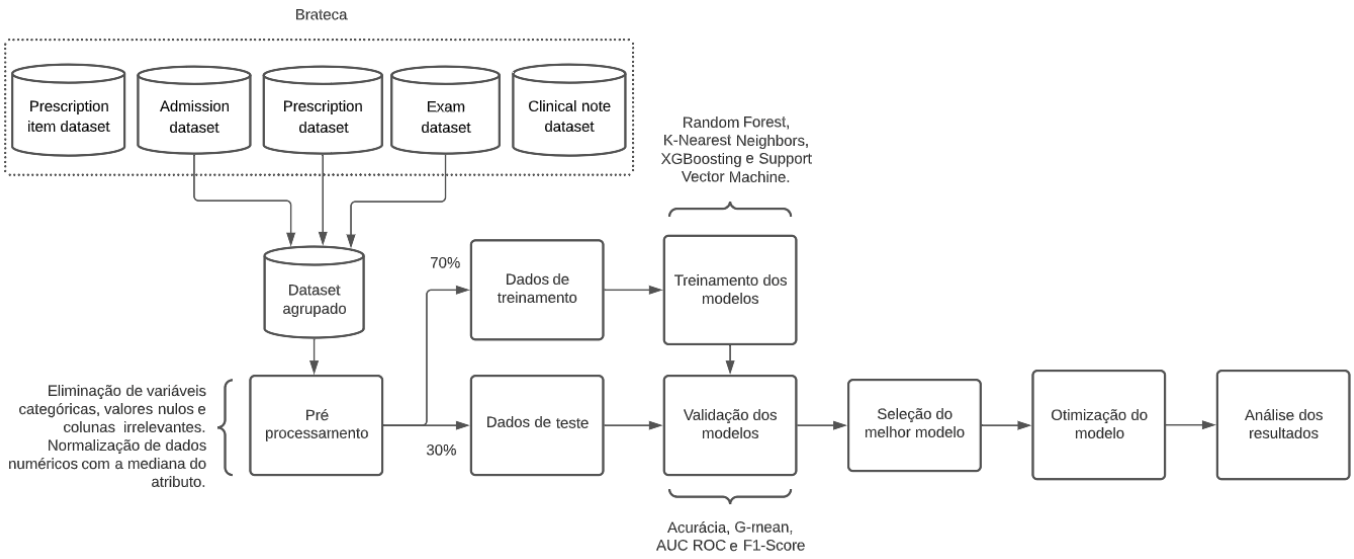

Figura 1. Diagrama de Modelo

1) Eliminação de variáveis categóricas: Dado que os algoritmos de Aprendizado de Máquina do pacote *scikit-learn* da linguagem de programação *Python* apenas aceitam entradas numéricas, as variáveis categóricas foram submetidas a transformações.

No conjunto de dados, o único atributo categórico que foi trabalhado é o *Sex*. Para isso, foi aplicado um codificador para transformar *Female* em 0 e *Male* em 1, e para remover qualquer tipo de hierarquia, transformou-se os atributos em colunas.

2) Eliminação de valores nulos: Os modelos de Aprendizado de Máquina desenvolvidos não aceitam a presença de valores nulos. Existem algumas opções para tratar estes valores nulos: eliminação do atributo, eliminação dos registros que contenham o atributo nulo ou aplicação de alguma estratégia de preenchimento desses valores. Neste caso, os atributos com valores nulos existentes na base de dados são apresentados na Tabela II.

Atributo	Quantidade de valores nulos
Weight	62737
Height	71860
Complications	10421
Leucócitos	34652
Plaquetas	35703
Creatinina	37601
Eosinófilos	41279
Potássio	42005

Tabela II
VALORES NULOS

Dado que as colunas de *Weight* e *Height* possuem mais de 3/4 dos dados nulos, optou-se pela eliminação desses atributos. Para o restante das variáveis recorreu-se ao preenchimento dos valores pelo valor da mediana, com o objetivo de não gerar impacto no modelo.

3) Eliminação de colunas desnecessárias: Antes da compilação dos três conjuntos de dados em um, cada um deles possuía uma coluna de identificação para se ligar às outras tabelas. Como não são mais necessárias, estas colunas foram removidas.

4) Normalização de Dados Numéricos: A normalização é uma técnica para mudar os valores das colunas numéricas no conjunto de dados para usar uma escala comum, sem distorcer as diferenças nos intervalos de valores e nem perder informações. A normalização também é necessária para alguns algoritmos para modelar os dados corretamente.

Como no conjunto de dados de entrada existem colunas com valores variando de 0 a 1 e outras colunas com valores maiores, a diferença na escala dos números pode causar problemas ao tentar combinar os valores durante a modelagem.

A normalização evita esses problemas criando novos valores que mantêm a distribuição geral e as proporções nos dados de origem, mantendo os valores em uma escala aplicada em todas as colunas numéricas usadas no modelo.

Como estratégia foi utilizado o *MinMaxScaler* (fórmula abaixo) para a reescala de dados, seu diferencial se dá uma vez que este age sobre a coluna, ou seja, o cálculo da reescala é feito de forma independente entre cada coluna, de tal forma que a nova escala se dará entre 0 e 1 (ou -1 e 1 se houverem valores negativos no conjunto de dados).

$$MinMaxScaler = \frac{X - Xmin}{Xmax - Xmin}$$

C. Aprendizado de máquina

Aprendizado de máquina é uma área de inteligência artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática [16].

Nesta sub-seção, explora-se o Aprendizado de Máquina Supervisionado, que a partir de raciocínios sobre exemplos de

treinamento fornecidos como entrada, o algoritmo se ajusta para mapear saídas correspondentes ao treino, para predizer o tempo de internação do paciente.

O fluxo de desenvolvimento pode ser observado na Figura 1, onde após o pré processamento, os dados foram separados em dados de treino e teste, que será demonstrado na subseção III-C1, com isso treinou-se os modelos a partir dos dados de treinamento, selecionando o modelo que resulta nas melhores métricas, que será demonstrado na sub-seção III-D. Por fim, otimizou-se esse modelo em busca dos melhores hiperparâmetros, e analisou-se os resultados finais, na subseção III-E.

1) Dados de Treino e Teste: Após a limpeza e o pré-processamento, 72081 registros foram extraídos e obtidos para a tarefa de mineração de dados. Separar os dados em conjuntos de treinamento e teste é uma parte da avaliação de modelos. Nesta fase, foi particionado o conjunto de dados em um conjunto de treinamento e um conjunto de teste; 70% dos dados (50456 registros) foram usados para treinamento, e 30% dos dados (21625 registros) foram usados para teste. O conjunto de treinamento foi usado para ajustar os parâmetros dos modelos, e o conjunto de testes foi usado para avaliar sua capacidade preditiva.

O porcentagem entre conjunto de treinamento e conjunto de destes não possui uma regra fixa, no estudo [17] é demonstrado, baseado em análise estatística, que uma separação com 70% dos dados para treino e 30% dos dados para testes é uma das melhores opções de divisão.

2) Coluna Alvo: O objetivo é a criação de um modelo de Aprendizado de Máquina para identificar o tempo de internação do paciente, logo a variável alvo será o tempo de internação, mais especificamente intervalos de tempo de internação. A Figura 2 mostra a distribuição do número de dias de internação na base Brateca.

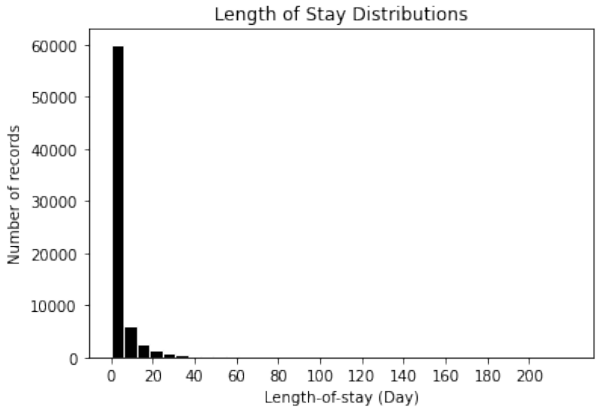

Figura 2. Distribuição do número de dias de internação na base de dados de treinamento

Observa-se que existem mais exemplos com poucos números de dias, diminuindo a quantidade de amostras conforme esse número aumenta. Neste estudo o objetivo foi identificar o tempo de internação dentro do intervalo de até 3 dias de internação, entre 3 e 5 dias e 5 ou mais dias. A distribuição dos dias de internação consta na Figura 3.

A coluna alvo foi gerada diminuindo a data de saída com a data de entrada do paciente, disponíveis no conjunto de dados de *Admission* do Brateca, o número resultando foi então convertido para uma das 3 classes definidas.

Com esse objetivo de coluna alvo, nosso problema passa a ser um problema de multi-classificação.

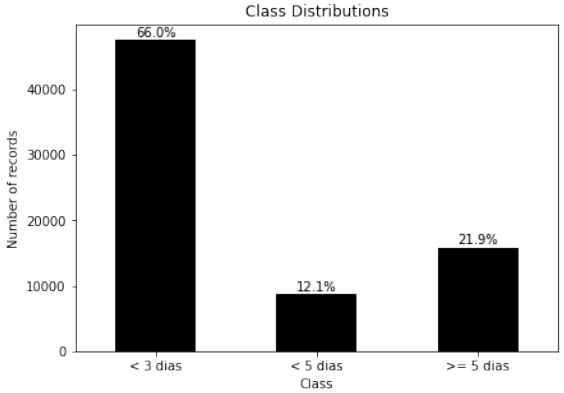

Figura 3. Distribuição agrupada do número de dias de internação na base de dados de treinamento

Nota-se também, que nossa base de dados é desbalanceada, havendo muito mais exemplos de pacientes que ficaram menos que 3 dias, do que exemplos que ficaram mais, deve-se levar isso em consideração e usar estratégias para mitigar esse tipo de problema na hora da criação e treinamento dos modelos.

3) Modelagem: Com a coluna alvo decidida e a base de dados separada em treino e testes, inicia-se a exploração dos algoritmos para resolver esse tipo de problema.

4) Modelo Base: Com o objetivo de perceber qual o mínimo de performance que o modelo preditivo desenvolvido deverá atingir, foi desenvolvido um modelo base simples, sem exigência computacional. Nesse sentido, aplicou-se um modelo de *Decision Tree* [18], sem qualquer ajuste de hiperparâmetro e sem considerar o balanceamento dos dados. Os resultados do modelo base, Tabela III, mostram que a *acurácia balanceada* foi de 63,27%, *G-Mean* de 78,50%, *AUC ROC Balanceado* de 72,22% e *F1-Score Balanceado* de 76,87%. Os parâmetros padrões do modelo base constam na Tabela X do Apêndice A

Modelo	Wt. F1-Score	G-Mean	Wt. AUC ROC	Wt. Acc
Decision Tree	76,87%	78,50%	74,22%	63,27%

Tabela III
RESULTADOS OBTIDOS PELO MODELO BASE

As métricas escolhidas aqui levam em consideração o desbalanceamento de dados, como existe uma classe majoritária, métricas como precisão e acurácia que não levam isso em consideração tendem a serem muito altas e dar a falsa sensação de um modelo bom, pois um modelo que colocasse todos os testes como a classe majoritária teria essas métricas altas,

as métricas de *F1-Score Balanceado*, *G-Mean*, *AUC ROC Balanceado* e *Acurácia balanceada* levam em consideração o desbalanceamento das classes. Em [19] e [20] são discutidas, de uma maneira geral, essas métricas.

A utilização de um modelo base aparenta ser um bom ponto de partida para futura comparação com os modelos desenvolvidos utilizando algoritmos de Aprendizado de Máquina mais complexos.

D. Seleção do modelo

Existem várias opções de algoritmos preditivos que podem ser implementados com o objetivo de prever o tempo de internação do paciente, cada um com as suas vantagens e desvantagens em casos específicos. Foram testados quatro modelos diferentes, com o mesmo conjunto de dados, a fim de identificar qual o melhor modelo tendo em vista o tipo de problema.

Os quatro algoritmos testados foram: *Random Forest* [21], *K-Nearest Neighbors* [22], *XGBoost* [23] e *Support Vector Machine* [24], disponíveis no pacote do *scikit-learn* e *xg-boosting*. Dentro das diversas opções de escolha para modelagem, os algoritmos foram escolhidos, essencialmente, com base na revisão da literatura, sendo utilizados por diversos estudos e com bons resultados apresentados. Quanto aos parâmetros de cada algoritmo, foram utilizados seus valores padrão, pois nesta fase deseja-se apenas perceber qual algoritmo possui maior potencial para resolver o problema em questão. Os parâmetros padrões de cada modelo constam nas Tabelas XII, XIV, XIII, XI do Apêndice A. Os resultados constam na Tabela IV.

Modelo	Wt. F1-Score	G-Mean	Wt. AUC ROC	Wt. Acc
Random Forest	81,35%	82,05%	89,12%	67,69%
KNN	77,98%	76,32%	82,87%	62,36%
XGBoost	80,20%	80,08%	89,45%	64,82%
SVM	77,60%	75,41%	82,48%	61,07%

Tabela IV
RESULTADOS OBTIDOS PELOS QUATRO MODELOS TESTADOS

Ao analisar os resultados da Tabela IV nota-se que em sua maioria, os modelos possuem métricas maiores em comparação ao modelo base, apresentado na seção III-C4, sem otimização dos parâmetros dos algoritmos, porém ainda são relativamente baixas. O fato de ainda não considerarmos o desbalanceamento dos dados interfere na performance dos modelos.

Como discutido anteriormente, nossa base de dados é desbalanceada, logo, é preciso considerar isso na hora da modelagem, há diversas estratégias para lidar com base de dados desbalanceadas como *Oversampling, Undersampling, SMOTE* [25], entre outras. Essas estratégias acabam adicionando ou removendo dados da base a fim de balanceá-la. Uma outra alternativa, é alterar o peso de cada classe para ser levada em consideração na hora do treinamento, dessa forma, não é necessário modificar e adicionar ruídos à base.

Primeiramente foi adicionado o peso nas classes. Para isso é adicionado o atributo *class_weight='balanced'* para todos os modelos. Os resultados constam na Tabela V.

Modelo	Wt. F1-Score	G-Mean	Wt. AUC ROC	Wt. Acc
Random Forest	80,88%	81,70%	88,60%	68,00%
KNN	77,98%	76,32%	82,87%	62,36%
XGBoost	80,77%	82,65%	89,09%	71,92%
SVM	79,00%	80,01%	86,43%	68,88%

Tabela V
RESULTADOS OBTIDOS PELOS QUATRO MODELOS SENSÍVEIS AO CUSTO

Observando a Tabela V, nota-se que com o uso do peso nas classes para o cálculo da função dos modelos gerou resultados melhores para os modelos, dado que agora as classes minoritárias tem um peso maior na hora da convergência. Em termos de predição das classes, a acurácia balanceada foi melhor em todos os casos, com um ganho médio de 3,8%

Como uma segunda estratégia, foi utilizado *sampling*. A biblioteca do *scikit-learn* nos oferece um modelo de *Random Forest* balanceado que utiliza *undersampling* aleatoriamente em cada iteração para equilibrá-la. Alternativamente, também será criado um modelo de Random Forest com o uso da técnica de SMOTE. Os resultados constam na Tabela VI

Modelo	F1-Score	G-Mean	AUC ROC	Wt. Acc
Balanced Random Forest	81,23%	83,27%	89,48%	72,55%
SMOTE Random Forest	87,17%	90,33%	96,54%	87,19%

Tabela VI
RESULTADOS OBTIDOS PELO ESTRATÉGIA DE SAMPLING

A estratégia de *sampling* gerou os modelos com os melhores resultados, conforme demonstrado na Tabela VI.

O modelo *Balanced Random Forest* realiza o *undersampling* a cada iteração para balancear os dados, ou seja, descarta os dados da classe majoritária com o objetivo de equilibrar com as classes minoritárias. Uma consequência do *undersampling* é a perda de informação, dado que ocorre o descarte de amostras, porém os resultados foram promissores, mesmo com a perda de informação.

O modelo com *SMOTE* gera amostras da classe minoritária selecionando exemplos que estão próximos no espaço de recursos de amostras da classe minoritária existentes. Este foi o modelo que obteve as melhores métricas em comparação com os outros, porém ele adiciona ruído à base de dados.

Como o objetivo não é adicionar ruídos a base de dados, será desconsidero o modelo de *Random Forest* com *SMOTE* para os próximos passos do estudo, mas se demonstrou uma ótima técnica que gerou bons resultados dos quais podem ser usados futuramente.

Nesta análise, a métrica escolhida para selecionar o modelo com melhores resultados, a ser utilizado nas próximas etapas, foi a *acurácia balanceada*. Em termos gerais, nota-se pelos resultados obtidos que o algoritmos *XGBoost* e *Balanced Random Forest* foram os modelos que apresentaram os melhores resultados, em termos de *acurácia* e *F1-score*.

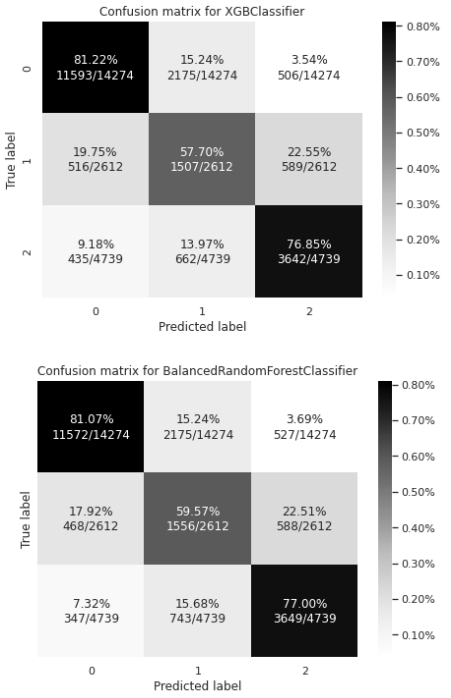

Figura 4. Comparação dos Modelos de Balanced Random Forest e XGBoost

Ao analisar as matrizes de confusão, conforme a Figura 4, observa-se um comportamento comum entre os modelos, pois possuem uma performance um pouco melhor nas classes dos extremos do que na classe intermediaria.

O algoritmo de *Balanced Random Forest* será explorado para as etapas restantes. O objetivo a seguir será otimizar os parâmetros deste modelo para obter o melhor resultado possível.

E. Otimização dos parâmetros

Após selecionado o algoritmo com melhores resultados tendo em conta o conjunto de dados inicial e o problema, existem duas alternativas de otimizar esta performance. A primeira consiste em fazer variar o conjunto de variáveis a incluir no modelo, com base na sua relevância para a previsão, ou incluir novas variáveis, quer através da seleção de outras variáveis disponíveis ou transformação de outras já existentes. A segunda maneira se baseia em otimizar os parâmetros do próprio algoritmo, ou seja, variar os atributos de entrada do algoritmo e selecionar a combinação desses atributos com os melhores resultados. A metodologia seguida nesta etapa será variar os atributos do algoritmo.

Neste momento, foi implementado um dos métodos mais comuns para otimização dos parâmetros de um algoritmo, o *Random Search*. Este método consiste em definir um conjunto de valores possíveis para os parâmetros do modelo e aleatoriamente são testadas diferentes combinações desses valores. Isto permite descobrir combinações de parâmetros não intuitivas, no entanto exige um tempo computacional maior. Para avaliar cada combinação, o método recorre à metodologia de validação cruzada, sendo o conjunto de dados dividido em 3 subconjuntos, escolhendo a melhor combinação com base na métrica definida. Os parâmetros a serem avaliados por este método estão representados na Tabela VII, bem como os valores definidos para cada parâmetro.

Atributo	Valores
min_samples_leaf	[1, 2, 4]
min_samples_split	[2, 5, 10]
max_features	['auto', 'sqrt', 'log2']
max_depth	[4, 5, 10, 50]
criterion	['gini', 'entropy']
n_estimators	[50, 100, 200, 400, 600]

Tabela VII
POSSÍVEIS VALORES DOS HIPERPARÂMETROS

Quanto à otimização dos parâmetros do algoritmo *Random Forest* (RF) com recurso ao método *Random Search*, foi observado que, na sua maioria, este método consegue melhorar ligeiramente a performance do modelo, podendo não ser consideradas melhorias significativas. No entanto, verifica-se em situações pontuais um decréscimo do desempenho uma vez que com este método apenas são testadas algumas combinações aleatórias dos parâmetros, que podem ser piores que os parâmetros base definidos para este algoritmo.

Modelo	Wt. F1-Score	G-Mean	Wt. Acc
Base Model	76,87%	78,50%	63,27%
Balanced RF Default	81,23%	83,27%	72,55%
Balanced RF Random Search	81,55%	83,67%	73,05%

Tabela VIII
COMPARAÇÃO DOS MODELOS OTIMIZADOS

Verifica-se, através da Tabela VIII, uma pequena melhoria na performance dos modelos desenvolvidos para cada especialidade, quando comparados com os resultados apresentados anteriormente. A performance do modelo otimizado apresenta um ganho de 0,5% em termos de acurácia em comparação ao modelo não otimizado. Assim, estes são os modelos finais apresentados como solução para resolver o problema proposto.

IV. RESULTADOS

Nesta seção são analisados os resultados obtidos pelo melhor modelo, assim como explorar a explicabilidade do mesmo.

A. Resultados obtidos

A seguinte matriz de confusão, Figura 5, representa os resultados do melhor modelo de *Balanced Random Forest*

O modelo final consegue classificar corretamente 81.76% dos pacientes que ficaram menos de 3 dias, 60.07% dos pacientes que ficaram menos que 5 dias e 77.32% dos pacientes que ficaram 5 ou mais dias.

Nota-se que para as classes das extremidades, o maior erro ocorre para as classes mais próximas, ou seja, não há tantas ocorrências de erro onde o paciente ficou menos de 3 dias e o modelo classificou como 5 ou mais dias, sendo este resultado

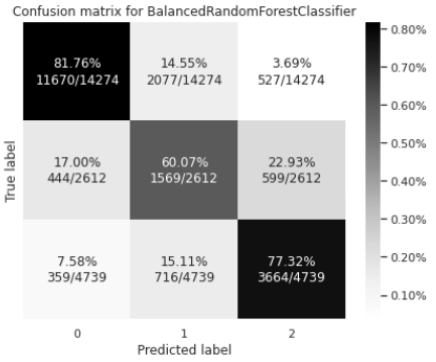

Figura 5. Matriz de Confusão do Modelo Final

3.69%, o contrário também não ocorre muito, sendo 7.58% dos casos.

Na classe intermediária, de menos que 5 dias, nota-se que o modelo não possui uma grande performance, obtendo um acerto de 60.07%, onde do total, 22.93% ele classificou como 5 ou mais dias e 17.00% como menos que 3 dias.

O comportamento da classe intermediária é intuitivo, espera-se que o modelo tenha um desempenho pior na classificação da classe intermediária, dado que esta classe pode ser parecida com qualquer uma de suas vizinhas, consequentemente, o modelo tem mais dificuldade em realizar a predição.

B. Interpretação dos Resultados

Nesta etapa, é realizada uma analise no fator de importância, que refere-se a técnicas que atribuem uma pontuação aos recursos de entrada com base em quão úteis eles são na previsão de uma variável de destino.

As pontuações são úteis e podem ser usadas em uma variedade de situações em um problema de modelagem preditiva, como:

- Melhor compreensão dos dados;
- Melhor compreensão de um modelo;
- Reduzindo o número de recursos de entrada.

A análise do fator de importância é fornecida automaticamente pela biblioteca do *scikit-learn* através do *feature_importances_*, porém, mecanismos comuns para calcular a importância de recursos podem ser tendenciosos, conforme demonstra [26], esses mecanismos tendem a aumentar a importância de variáveis categóricas contínuas ou de alta cardinalidade.

Para análise do fator de importância, será utilizada a *Permutation importance*, introduzida inicialmente por [21]. A Permutation Importance também possui alguns pontos negativos, conforme demonstra [27] essa estratégia pode superestimar a importância das variáveis preditoras correlacionadas.

o resultado da análise do fator de importância com o uso de *Permutation Importance* para o nosso modelo consta na Tabela IX;

Ao analisar a Tabela IX nota-se que o atributo *Emergency* é o atributo que tem mais peso na decisão do modelo ao classificar, com um fator de importância de 17.87%, *Surgical*, *Creatinina* e *Different_Drugs* também são atributos com um bom fator de importância.

Atributo	Fator de Importância
Emergency	17,87%
Surgical	7,75%
Creatinina	2,94%
Different_Drugs	2,78%
Plaquetas	2,35%
Antibiotics	2,21%
Potássio	1,99%
Complications	1,88%
Leucócitos	1,85%
Eosinófilos	1,64%

Tabela IX
OS 10 ATRIBUTOS COM O MAIOR FATOR DE IMPORTÂNCIA

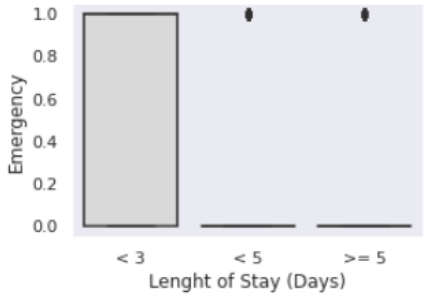

Figura 6. Diagrama de caixa do atributo Emergency

Analisando o atributo de *Emergency* separadamente, através do diagrama de caixa da Figura 6, identifica-se um comportamento no qual não existem instâncias da classe 2 ($<$ 5 dias) ou da classe 3 ($\geq$ 5 dias) que não sejam *Emergency*, ou seja, a variável consegue separar bem a classe 1 ($<$ 3 dias) das outras classes, sendo uma boa candidata para se levar em conta na hora da classificação, justificando o seu valor de importância alto. Este fator também é intuitivo, dado que pacientes em emergência tendem a ficar mais tempo internados.

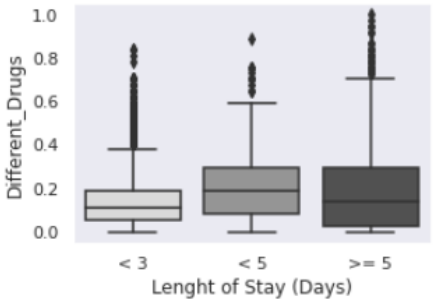

Figura 7. Diagrama de caixa do atributo Different Drugs

Analisando o atributo de *Different_Drugs*, através do diagrama de caixa da Figura 7, nota-se um comportamento um pouco diferente, observa-se que as classes 2 e 3 possuem instâncias com valores maiores de *Different_Drugs*, porém as três classes possuem exemplos com valores menores, ou seja, o atributo ainda é relevante para separar as classes em

altos valores, mas há sobreposição em valores mais baixos, justificando seu fator de importância menor.

Em complemento, analisou-se os valores de SHAP (SHapley Additive exPlanations). A análise de valores de SHAP é um método relativamente novo, baseado na teoria dos jogos, que permite explicar a saída de um modelo de aprendizado de máquina [28]. O resultado gerado são os indicadores que apresentam uma maior correlação com o resultado da classificação de cada classe, representando a importância para a previsão de cada classe específica, conforme demonstrado na Figura 8.

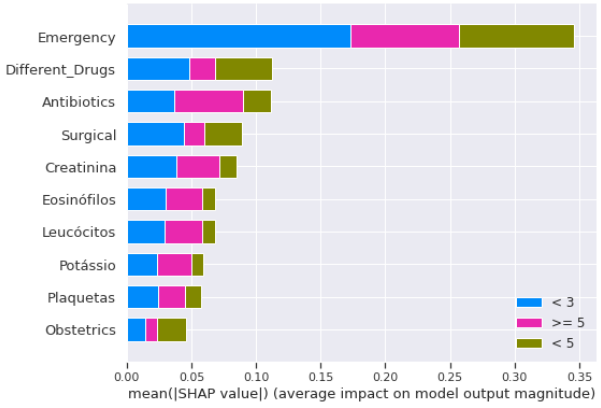

Figura 8. Os 10 atributos com maior fator de importância, segundo a análise do SHAP

A análise de SHAP exige bastante recurso computacional, para a realização desse teste foram utilizadas uma amostra de 2000 instâncias do conjunto de dados de teste.

Nota-se, a partir da Figura 8 que em comparação com a análise anterior, do fator de importância, demonstrado na Tabela IX, 9 dos 10 atributos continuam os mesmos. Observa-se a diferença na alteração do atributo *Complications* pelo *Obstetrics* entre os 10 mais importantes, e também a ordem de importância entre os atributos. Isso ocorre pelo fato da análise de SHAP conceder a cada atributo um valor de importância para uma previsão específica baseado na magnitude das atribuições de recursos [28]. Essa definição de importância difere da definição baseada na diminuição do desempenho do modelo, como no caso da importância do *Permutation Importance*. O fato de 9 dos 10 atributos serem os mesmos considerando diferentes definições de importância, aumenta a confiabilidade na explicabilidade do modelo.

A Figura 8 demonstra uma análise mais detalhada no que diz respeito a importância do atributo para cada uma das classes. Por exemplo, o atributo *Emergency*, que possui o maior *SHAP value* tem uma grande importância para a classe onde o paciente fica menos que 3 dias, porém entre as outras duas classes ela tem uma importância equilibrada, o que vai ao encontro do diagrama de caixa da Figura 6. O atributo *Antibiotics* por sua vez, possui uma maior importância para a terceira classe, onde o paciente fica 5 ou mais dias, e uma importância menos para as outras.

V. Conclusões e trabalhos futuros

Neste trabalho foram apresentadas as etapas de tratamento, desenvolvimento e resultados obtidos na exploração do uso de algoritmos de Aprendizado de Máquina, análise dos modelos e interpretação dos resultados para encontrar os melhores indicadores que caracterizam o tempo de internação de um paciente. Os modelos testados apresentaram uma boa avaliação em relação aos modelos base e uma boa performance na base de dados do Brateca.

Os dados de saúde são, geralmente, não estruturados. Em termos de praticidade, este trabalho limitou-se ao uso de dados estruturados, mas para trabalhos posteriores pode-se explorar o uso de dados não estruturados e considerar os dois conjuntos de dados do Brateca que não foram incluídos. Ainda para trabalhos futuros existem várias possibilidades, com Aprendizado de Máquina, pode-se explorar outros tipos de modelos e otimizações com o objetivo de fazer novas comparações. No que tange a melhora da performance do algoritmo desenvolvido, pode-se buscar a inserção de mais dados na base de dados a fim de balanceá-la organicamente. Uma outra alternativa para trabalhos futuros é explorar outros tipos de estratégias para predição do tempo de internação como Redes Neurais ou Análise de Sobrevivência.

Como contribuição e aproveitamento deste estudo, pode-se utilizar dos indicadores apresentados neste trabalho para comparação com estudos posteriores na base de dados Brateca ou outros e direcionar estudos na área da medicina com Aprendizado de Máquina.

Referências

[1] A. Marshall, C. Vasilakis, and E. El-Darzi, "Length of stay-based patient flow models: Recent developments and future directions," *Health care management science*, vol. 8, pp. 213–20, 09 2005.

[2] M. J. Fine, H. M. Pratt, D. Obrosky, J. R. Lave, L. J. McIntosh, D. E. Singer, C. M. Coley, and W. N. Kapoor, "Relation between length of hospital stay and costs of care for patients with community-acquired pneumonia," *The American Journal of Medicine*, vol. 109, no. 5, pp. 378–385, 2000.

[3] C. Shaw, *How can hospital performance be measured and monitored?* World Health Organization Regional Office for Europe, 2003.

[4] D. A. Huntley, D. W. Cho, J. Christman, and J. G. Csernansky, "Predicting length of stay in an acute psychiatric hospital," *Psychiatric Services*, vol. 49, no. 8, pp. 1049–1053, 1998, pMID: 9712211.

[5] S. Shea, R. Sideli, W. DuMouchel, G. Pulver, R. Arons, and P. Clayton, "Computer-generated informational messages directed to physicians: effect on length of hospital stay," *Journal of the American Medical Informatics Association : JAMIA*, vol. 2, no. 1, p. 58—64, 1995.

[6] M. Rachda Naila, P. Caulier, S. Chaabane, A. Chraibi, and S. Piechowiak, "Using machine learning models to predict the length of stay in a hospital setting," 04 2020.

[7] S. G. Gurazada, S. C. Gao, F. Burstein, and P. Buntine, "Predicting patient length of stay in australian emergency departments using data mining," *Sensors*, vol. 22, no. 13, 2022.

[8] S. Aghajani and M. Kargari, "Determining factors influencing length of stay and predicting length of stay using data mining in the general surgery department," *Hospital Practices and Research*, vol. 1, no. 2, pp. 53–58, 2016.

[9] Y. Chen, "Prediction and analysis of length of stay based on nonlinear weighted xgboost algorithm in hospital," *Journal of Healthcare Engineering*, vol. 2021, pp. 1–9, 12 2021.

[10] O. Z. Maimon and L. Rokach, *Data mining with decision trees: theory and applications*. World scientific, 2014, vol. 81.

[11] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random forests," in *Ensemble machine learning*. Springer, 2012, pp. 157–175.

[12] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.
[13] "Github: length-of-stay-prediction-brateca," https://github.com/larissamagistrali/length-of-stay-prediction-brateca, acesso: 12/11/2022.
[14] B. Consoli, H. Dias, R. Vieira, R. Bordini, and U. Ana, "Brateca (brazilian tertiary care dataset): a clinical information dataset for the portuguese language." LREC, 2022.
[15] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *circulation*, vol. 101, no. 23, pp. e215–e220, 2000.
[16] T. Mitchell, B. Buchanan, G. DeJong, T. Dietterich, P. Rosenbloom, and A. Waibel, "Machine learning," *Annual Review of Computer Science*, vol. 4, no. 1, pp. 417–433, 1990.
[17] Q. Nguyen, H.-B. Ly, H. Lanh, N. Al-Ansari, H. Le, T. Van Quan, I. Prakash, and B. Pham, "Influence of data splitting on performance of machine learning models in prediction of shear strength of soil," *Mathematical Problems in Engineering*, vol. 2021, 02 2021.
[18] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.
[19] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.
[20] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.
[21] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[22] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[23] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
[24] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
[25] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan Kaufmann Publishers, 2012.
[26] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC bioinformatics*, vol. 8, no. 1, pp. 1–21, 2007.
[27] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis, "Conditional variable importance for random forests," *BMC bioinformatics*, vol. 9, no. 1, pp. 1–11, 2008.
[28] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

APÊNDICE

A.1 Parâmetros dos Modelos

Atributo	Valor
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	101
max_leaf_nodes	None
min_impurity_decrease	0.0
class_weight	None
ccp_alpha	0.0 height

Tabela X
DECISION TREE PARÂMETROS

Atributo	Valor
n_neighbors	5
weights	uniform
algorithm	auto
leaf_size	30
p	2
metric	minkowski
metric_params	None
n_jobs	None

Tabela XI
KNN PARÂMETROS

Atributo	Valores
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	101
verbose	0
warm_start	False
class_weight	None
ccp_alpha	0.0
max_samples	None

Tabela XII
RANDOM FOREST PARÂMETROS

Atributo	Valores
C	1.0
kernel	rbf
degree	3
gamma	scale
coef0	0.0
shrinking	True
probability	True
tol	0.001
cache_size	200
class_weight	None
verbose	False
max_iter	-1
decision_function_shape	ovr
break_ties	False
random_state	101

Tabela XIII
SVM PARÂMETROS

Atributo	Valores
booster	gbtree
verbosity	1
validate_parameters	False
disable_default_eval_metric	False
max_depth	6
gamma	0
min_child_weight	1
max_delta_step	0
subsample	1
sampling_method	uniform
scale_pos_weight	1
max_leaves	0
eta	0.3

Tabela XIV
XGBOOST PARÂMETROS