



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Anderson Sales Zambeli

Comparação entre algoritmos de Machine Learning para previsão de mortalidade em UTI: algoritmo HyperFast em relação a redes neurais MLP e o XGBoost

Florianópolis,
2025

Anderson Sales Zambeli

Comparação entre algoritmos de Machine Learning para previsão de mortalidade em UTI: algoritmo HyperFast em relação a redes neurais MLP e o XGBoost

Trabalho de Conclusão de Curso submetido ao Curso de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador(a): André Wust Zibetti

Florianópolis,
2025

Zambeli, Anderson Sales

Comparação entre algoritmos de Machine Learning para previsão de mortalidade em UTI: algoritmo HyperFast em relação a redes neurais MLP e o XGBoost / Anderson Sales Zambeli ; orientador, André Wüst Zibetti, 2025.

105 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2025.

Inclui referências.

1. Ciências da Computação. 2. HyperFast. 3. Machine Learning. 4. Predição. 5. Unidade de Terapia Intensiva. I. Zibetti, André Wüst. II. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. III. Título.

Anderson Sales Zambeli

Comparação entre algoritmos de Machine Learning para previsão de mortalidade em UTI: algoritmo HyperFast em relação a redes neurais MLP e o XGBoost

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Título de Bacharel em Ciências da Computação e aprovado em sua forma final pelo Curso de Graduação em Ciências da Computação do INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico da Universidade Federal de Santa Catarina.

Florianópolis, 02 de dezembro de 2025.

Profa. Lúcia Helena Martins Pacheco, Dra.
Coordenadora do Curso

Banca examinadora:

Prof. André Wust Zibetti, Dr.
Orientador
Universidade Federal de Santa Catarina

Prof. Alexandre Gonçalves Silva, Dr.
Avaliador
Universidade Federal de Santa Catarina

Prof. Jônata Tyska Carvalho, Dr.
Avaliador
Universidade Federal de Santa Catarina

RESUMO

Neste trabalho, é comparado o algoritmo de machine learning HyperFast, um modelo de machine learning que faz uso de meta-learning e hiper redes voltado para dados tabulares, na predição de mortalidade em até sete dias após a primeira entrada na Unidade Terapia Intensiva (UTI), usando dados coletados nas primeiras vinte quatro horas da estadia na UTI, com os algoritmos XGBoost e rede neural MLP (Multi-Layer Perceptron). Os dados foram extraídos do Medical Information Mart for Intensive Care IV (MIMIC-IV). Quatro conjuntos de dados foram criados. O primeiro é um conjunto sem balanceamento de classes e eliminação de medições outliers, o segundo foi retirando as medições outliers, o terceiro fez-se balanceamento de classes e o quarto, balanceamento de classes e eliminação de medições outliers. Os algoritmos foram testados em cada conjunto de dados. Os resultados dos algoritmos foram analisados, discutidos e comparados tendo em vista a robustez a presença de outliers no conjunto de dados e desbalanceamento de classes usando a métrica Área sob a curva Precision Recall (AUPRC) a métrica como principal métrica de avaliação. Especificamente, foi analisado o algoritmo MLP em comparação às métricas e tempo de treinamento/predição com o HyperFast, tendo em vista arquitetura similar e execução em CPU ou GPU. Os resultados mostraram que o algoritmo HyperFast é uma opção viável, pois o tempo de treinamento foi menor do que o tempo de um MLP com arquitetura similar da rede neural gerada pelo HyperFast, quando ambos rodam em CPU e sendo bem mais rápido quando rodado em GPU. Outro ponto é que o tempo de treino do HyperFast não dependeu do tamanho do conjunto de dados de entrada, somente dos hiperparâmetros usados. Além disso, as performances em relação a métrica AUPRC ficaram próximas na maioria dos conjuntos comparando HyperFast com os outros algoritmos. Já o modelo XGBoost superou os outros algoritmos em relação ao tempo com as métricas melhores em todos conjuntos.

Palavras-chave: HyperFast; Machine Learning; Predição; Unidade de Terapia Intensiva.

LISTA DE FIGURAS

Figura 1 – Arquitetura do Algoritmo HyperFast.....	17
Figura 2 - Histogramas da variável anchor_age.....	34
Figura 3 - Histogramas da variável HeartMean_mean.....	34
Figura 4 - Histogramas da variável HeartMean_mean sem extremos.....	35
Figura 5 - Contagem de mortes por dia a partir da entrada na UTI.....	35
.....	36
Figura 6 - Fluxo de exclusão de dados.....	37
Figura 7 - Gráfico da curva Precisão-Recall para os modelos no conjunto D4.....	41
Figura 8 - Gráfico do número de dados vs tempo de Treinamento usando os conjuntos D1 e D3.....	43
Figura 9 - Gráfico do número de dados vs tempo de Treinamento usando os conjuntos D2 e D4.Fonte: Elaboração própria.....	44
Figura 10 - Gráficos dos tempos de treino em relação a quantidade de dados dos modelos HyperFast em CPU sem fine tuning.....	47
Figura 11 - Gráficos dos tempos de treino em relação a quantidade de dados dos modelos HyperFast em CPU com fine tuning usando as quantidades de dados dos conjuntos D1 e D3.....	47

LISTA DE TABELAS

Tabela 1 - Trabalhos Relacionados.....	19
Tabela 2 - Variáveis escolhidas do MIMIC - IV.....	23
Tabela 3 - Consulta da variável Urea Nitrogen no banco.....	27
Tabela 4 - Número médio de medições de cada variável laboratorial nas primeiras 24 horas.....	28
Tabela 5 - Número de valores nulos e não nulos de cada variáveis da tabela gerada pelas consultas.....	29
Tabela 6 - Grids usados para os modelos.....	40
Tabela 7 - Resultados dos modelos no conjunto de teste de D4.....	42
Tabela 8 - Tempo de predição do HyperFast em CPU e Quantidade de dados.....	45
Tabela 9 - Tamanho dos modelos em MegaBytes.....	48
Tabela A5 - Estatísticas descritivas das variáveis numéricas do conjunto D1 após da imputação de valores.....	72
Tabela A6 – Resultados do GridSearchCV em cada conjunto.....	77

LISTA DE ABREVIATURAS E SIGLAS

UTI – Unidade de Terapia Intensiva

MPL- Multi-Layer Perceptron

XGB – XGBoost

ML – Machine Learning

MIMIC - Medical Information Mart for Intensive Care

RF - Random Features

PCA – Principal Component Analysis

IC - Intervalo de Confiança

AUC ROC - Area Under the Receiver Operating Characteristic Curve

AUPRC - Area Under the Precision Recall Curve

Sumário

1	Introdução.....	11
1.1	Objetivos.....	12
1.1.1	Objetivos Específicos.....	12
2	Fundamentação Teórica.....	13
2.1	Machine Learning.....	13
2.1.1	Aprendizado Supervisionado.....	13
2.2	Redes Neurais MLP.....	15
2.3	XGBoost.....	16
2.4	HyperFast.....	16
2.5	Trabalhos Relacionados.....	18
3	Metodologia.....	22
3.1	MIMIC IV.....	22
3.2	Entendimento do Problema.....	25
3.3	Implementação.....	26
3.4	Entendimento dos Dados.....	26
3.4.1	Exploração direto no banco usando consultas SQL.....	26
3.4.2	Extração dos Dados.....	28
3.4.3	Explorações adicionais.....	29
3.5	Pré-processamento dos Dados.....	35
3.5.1	Exclusão de Dados.....	37
3.5.2	Criação dos conjuntos.....	38
3.6	Modelagem.....	39
4	Resultados e Discussões.....	42
5	Conclusões.....	50
5.1	Limitações.....	50
5.2	Trabalhos Futuros.....	50
6	Referências.....	52
7	Apêndices.....	56
A	- Consulta para extrair os exames laboratoriais:.....	56

B - Consulta para extrair as medições de sinais vitais agregadas de mínimo, máximo, média e desvio padrão.....	57
C - Lista de referências para os valores de máximo e mínimo possíveis de cada variável.....	62
D - Tabela das estatísticas descritivas das variáveis numéricas.....	63
E - Tabela das estatísticas descritivas das variáveis categóricas.....	67
F - Tabela das estatísticas descritivas das variáveis numéricas do conjunto D1 antes da imputação de valores.....	67
G - Tabela das estatísticas descritivas das variáveis numéricas do conjunto D1 após da imputação de valores.....	72
H – Resultados do GridSearch CV em cada conjunto.....	77
I – Tabelas com os resultados dos modelos no conjunto de Teste de D1, D2 e D3..	80

1 Introdução

Entender e estudar as ocupações das Unidades de Terapia Intensiva (UTI) é algo de suma importância. Elas são cruciais no tratamento de pessoas em estado gravíssimo provendo equipamento especializado, tratamento priorizado e suporte a funções vitais de paciente, e por isso afetam diretamente o resultado de um tratamento delicado. Porém seu custo operacional é elevado em relação às outras unidades, envolvendo uma infraestrutura complexa, equipamentos de alta tecnologia e equipes multidisciplinares especializadas. Por esse motivo, recomenda-se que as taxas de ocupação se mantenham entre 70% e 75%, de forma a equilibrar eficiência econômica e capacidade de resposta a aumentos inesperados de demanda, como em pandemias ou surtos sazonais (Valentin et al., 2020).

No contexto brasileiro, a expansão dos leitos de terapia intensiva tem avançado na última década, com crescimento aproximado de 52% entre 2014 e 2024, atingindo mais de 73 mil leitos em operação. Apesar dessa ampliação, persistem acentuadas desigualdades regionais, nas quais a oferta é significativamente maior em grandes centros urbanos e regiões mais desenvolvidas, enquanto áreas periféricas e municípios de pequeno porte apresentam baixa cobertura (AMIB, 2024). Estudos do Conselho Federal de Medicina indicam que menos de 10% dos municípios brasileiros dispõem de ao menos um leito de UTI, o que demonstra a forte concentração territorial desses recursos (CFM, 2023).

Com isso em mente, ter maneiras de predizer quem precisa de uma UTI, o desfecho da internação, tempo de estadia e o perfil das pessoas que eventualmente fazem o uso dos leitos de uma UTI, é uma informação interessante para as tomadas de decisão sobre as políticas de saúde que envolvem a expansão, manutenção e logísticas dos leitos de UTI.

Tendo em vista o grande volume de dados coletados pelos sistemas de saúde privados e governamentais, o uso de técnicas de mineração de dados se apresenta como uma maneira interessante de extrair informações e criar aplicações, no âmbito de UTIs, que possam auxiliar na otimização dos recursos, tomadas de decisões rápidas e administração de leitos. Exemplos dessas aplicações são predizer se um paciente vai morrer em até um certo número de dias após sua entrada no hospital (ZHAH et al., 2020), dizer se a estadia de um paciente vai ser de longa duração (WU et al., 2021) ou predizer se o paciente vai desenvolver choque séptico durante sua estadia em uma UTI (NEMAT et al, 2018).

Novos algoritmos de aprendizado de máquina, uma técnica da mineração de dados, têm surgido nos últimos anos, trazendo novas inovações para a área. Logo, avaliá-los no uso da área médica é importante para entender como e onde seu uso traz ganhos, quais suas limitações e vantagens e que cenários eles são melhores indicados.

Este trabalho visa fazer um estudo de métodos de aprendizado de máquina para criar modelos de predição de mortalidade após sete dias da admissão na UTI, baseado nos dados coletados nas primeiras vinte quatro horas após entrada na UTI. A escolha da mortalidade em até 7 dias como desfecho neste trabalho pode ser entendida como uma especialização da mortalidade precoce em UTI, em especial dando atenção ao algoritmo HyperFast, publicado em fevereiro de 2024, um algoritmo que usa meta-learning e hiper redes para dados tabulares. Clinicamente, uma janela de 7 dias é suficiente para captar a maior densidade dos óbitos diretamente relacionados à gravidade inicial e ao manejo intensivo, evitando a influência excessiva de fatores tardios, como infecções hospitalares adquiridas durante a estadia (MARTIN-LOECHES et al, 2018).

1.1 Objetivos

1.1.1 Objetivo Geral

Criar modelos de predição de mortalidade em até sete dias após a primeira entrada na Unidade Terapia Intensiva e compará-los, em especial ao modelo gerado pelo algoritmo HyperFast.

1.1.1 Objetivos Específicos

- Criar modelos de machine learning (Rede Neurais MLP, XGBoost e HyperFast), para predição de mortalidade em até sete dias após a primeira entrada na Unidade Terapia Intensiva, usando os dados coletados nas primeiras vinte quatro horas da estadia.
- Comparar os modelos gerados pelos algoritmos MLP e XGBoost com os modelos gerados pelo HyperFast em relação a presença de outliers, desbalanceamento entre classes e tempo para treinar e testar o modelo.

Nas próximas seções, será mostrada a ideia geral deste trabalho. Na seção 2, Fundamentação Teórica, é apresentado a ideia geral dos algoritmos de machine learning escolhidos e os trabalhos relacionados vistos. A seção 3, Metodologia, mostra o processo usado para gerar os modelos. Na seção 4, os resultados são mostrados e discutidos. A seção 5 traz as considerações finais e possíveis trabalhos futuros.

2 Fundamentação Teórica

Neste trabalho serão abordadas algumas técnicas de ML, e nessa seção vamos explicá-las, assim como o conceito mais geral de ML. Ao final também serão apresentados os trabalhos relacionados.

2.1 Machine Learning

Machine Learning, uma área de inteligência artificial, consiste em fazer uso de algoritmos que aprendem a fazer decisões a partir de um conjunto de dados.

Os modelos podem ser divididos nas seguintes categorias: modelos supervisionados, modelos não supervisionados, modelos semi supervisionados e modelos de aprendizado por reforço.

Os modelos supervisionados usam um conjunto de dados rotulados (que já tem o resultado associado conhecido em relação ao problema em questão) para rotular novos dados. Modelos não supervisionados usam conjuntos de dados que não estão rotulados e buscam padrões e relações no conjunto de dados. Modelos semi supervisionados usam tanto os dados rotulados quanto dados não rotulados buscando também rotular novos dados. No aprendizado por reforço, o modelo aprende numa espécie de tentativa e erro, onde ações corretas são recompensadas e ações erradas punidas.

Neste trabalho o foco está nos modelos supervisionados e uma visão mais profunda é dada a seguir.

2.1.1 Aprendizado Supervisionado

No aprendizado supervisionado, recebemos um conjunto de dados rotulados, para gerar um modelo que faça novas rotulações em um conjunto de dados novos não rotulados.

Para gerar os modelos, os dados são vistos como um grupo de variáveis (X) e seus respectivos rótulos (Y). O objetivo é achar uma função f que mapeia X para Y , $f: X \rightarrow Y$.

O processo para gerar o modelo começa com o pré-processamento dos dados, onde é feita uma série de ações para deixar os dados prontos para serem usados. Essas ações incluem: limpeza de dados com variáveis faltantes, geração de novas variáveis a partir de outras, normalização de variáveis com valores numéricos, codificação de variáveis categóricas, remoção de valores extremos (outliers), entre outras.

Após isso, o conjunto é dividido em 2 subconjuntos, um conjunto de treino usado para treinar os modelos, e um conjunto de teste, usado para avaliar os modelos. O conjunto de treino ainda pode ser dividido em treino e validação, onde o conjunto de validação é usado para descobrir um bom grupo de hiperparâmetros do modelo.

Com os dados divididos, os tipos de modelos são selecionados, e então, treinados usando o conjunto de treino. Se busca também o melhor conjunto de hiperparâmetros, um processo conhecido como ajuste de hiperparâmetros, que pode envolver várias técnicas, como por exemplo a técnica de validação cruzada.

Achado o modelo treinado com melhor conjunto de hiper parâmetros, os dados do conjunto de testes são passados ao modelo que lhes atribuem rótulos. Esses rótulos preditos pelo modelo são comparados com os rótulos reais já conhecidos e várias métricas de avaliação são geradas a partir dessas comparações.

Os modelos podem ser de classificação, onde os rótulos são de um conjunto de classes e o modelo tenta dizer qual classe aquele dado faz parte, ou de regressão, onde o rótulo é um valor numérico e o modelo tenta achar qual valor aquele dado deve ter.

Neste trabalho, os modelos são de classificação binária, onde temos duas classes, e por isso as métricas usadas serão a acurácia balanceada, precisão, recall e AUC ROC e AUPRC.

A acurácia é definida como a porcentagem de rótulos corretos ditos pelos modelos. Sua fórmula é $\frac{TP+TN}{TP+TN+FP+FN}$ onde TP é o número de classificação verdadeiras positivas, TN o número de classificações verdadeiras negativas, FP o número de classificações falso positivas e FN o número de classificações falso negativas.

Precisão é a porcentagem de todas as classificações corretas de uma classe feitas pelo modelo em relação a todas as classificações feitas para aquela classe. Sua fórmula é $\frac{TP}{TP+FP}$.

Recall ou sensibilidade é a porcentagem de classificações corretas de uma classe feitas pelo modelo em relação ao total de valores reais daquela classe. Sua fórmula é $\frac{TP}{TP+FN}$.

A *AUC ROC (Area Under the Receiver Operating Characteristic Curve)* é uma métrica que avalia a capacidade de distinção correta de classe do modelo,

calculando a área sob a curva *ROC* (*Receiver Operating Characteristic*). Quanto maior o valor, melhor o modelo consegue diferenciar uma classe da outra.

A *AUCPR* (*Area Under Precision Recall Curve*) representa a área sob a curva no gráfico com o eixo Y representando a Precisão do modelo para a classe positiva e no eixo X o Recall do modelo para classe positiva, para vários pontos de corte (Threshold) e é mais apropriada que a AUC para casos onde existe forte desbalanceamento entre as classes positivas e negativas (Saito e Rehmsmeier, 2025).

2.2 Redes Neurais MLP

Redes neurais artificiais são algoritmos de ML inspirados no funcionamento dos neurônios do cérebro humano. As redes neurais Multilayer Perceptron (MLP) são uma classe de redes neurais compostas por múltiplas camadas de neurônios, sendo amplamente utilizadas para tarefas de classificação e regressão. Elas estão dentro da classe de aprendizado supervisionado.

São consideradas *feedforward*, significando que um dado é fornecido à camada de entrada e vai sendo passado para as camadas seguintes, e na saída da camada final, se tem o resultado.

Sua estrutura pode ser decomposta em 3 partes:

- Camada de entrada: aquela que recebe os dados.
- Camadas ocultas: são as camadas conectadas com a de entrada que fazem o processamento dos dados. Elas aplicam alguma função de ativação na informação que chega na suas entradas e passa o resultado para os neurônios da próxima camada.
- Camada de saída: conectada com a última camada oculta, produz o resultado final.

Na rede MLP, os neurônios de camadas adjacentes estão totalmente conectados com os da camada anterior. Cada neurônio possui uma função de ativação, que processa as informações recebidas para gerar a saída daquele neurônio. Outro ponto é que todas as conexões entre os neurônios possuem um peso associado a elas. São esses pesos que vão ser ajustados pelo treinamento da rede para que o dado fornecido na entrada produza o resultado esperado na saída.

O treinamento de uma MLP usa o algoritmo de retropropagação (backpropagation), que calcula o erro entre a saída prevista pela rede e a saída esperada e propaga para as camadas anteriores, ajustando os pesos das conexões entre os neurônios. Esse treinamento usa um conjunto de dados rotulados e o objetivo é minimizar a diferença entre a saída prevista pela rede e a saída esperada. O ajuste dos pesos é feito gradualmente de acordo com um hiperparâmetro chamado de *learning rate*.

O treinamento envolve várias passagens pelo conjunto de dados de treinamento até chegar em um resultado satisfatório. Cada passagem é chamada de *epoch*. Dependendo do tamanho do conjunto de dados e sua complexidade, esse processo pode levar um tempo significativo.

2.3 XGBoost

O XGBoost (Extreme Gradient Boosting), abreviado com XGB neste trabalho, é um algoritmo de aprendizado de máquina utilizado para resolver problemas de classificação e regressão, sendo uma implementação com melhorias de Gradient Boosting, um método de ensemble que constroi modelos fortes a partir de modelos fracos (tipicamente árvores de decisão).

Ele foi apresentado por Chen e Guestrin (2016).

O método de Gradient Boosting consiste em criar um modelo composto por vários modelos fracos, que são treinados de forma sequencial, fazendo com que cada novo modelo corrija os erros do anterior. Para isso, se ajusta às previsões no novo modelo para reduzir o gradiente da função de perda. A função de perda quantifica o erro entre o resultado esperado e o previsto.

Em relação ao Gradient Boosting, o algoritmo XGB inclui dois tipos de regularização a função de perda :

L1 (Lasso): Ajuda a reduzir o número de variáveis.

L2 (Ridge): Controla o impacto de cada variável.

Essas adições a função de perda penalizam modelos mais complexos gerados, assim minimizando o overfitting do modelo final.

Além disso, o algoritmo apresenta otimizações para fazer uso de computação paralela, aumentando a velocidade e eficiência computacional em relação ao Gradient Boosting.

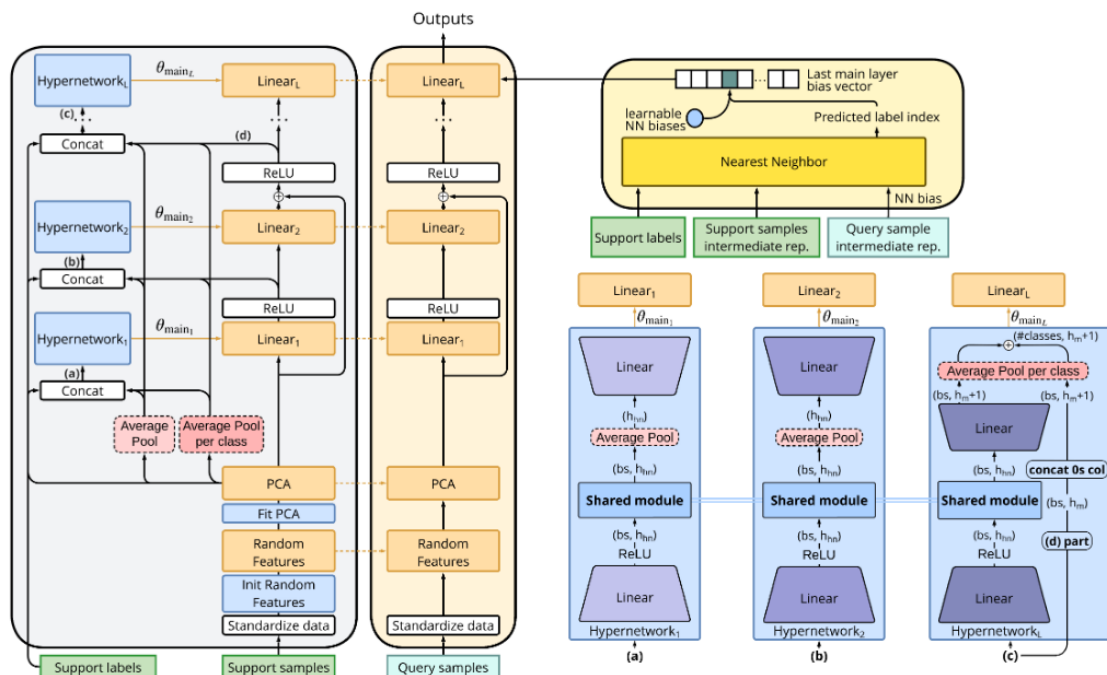
2.4 HyperFast

O HyperFast, apresentado no artigo *HyperFast: Instant Classification for Tabular Data* (BONET, 2024), introduz uma nova maneira de classificação de dados tabulares usando uma combinação de meta-aprendizado e hiper redes. Esta abordagem substitui o treinamento tradicional de redes neurais, gerando os pesos de uma rede neural principal em uma única etapa de passagem nos dados, eliminando a necessidade de treinamento e ajuste de hiper parâmetros para novos conjuntos de dados.

Uma hiper rede é rede neural treinada para achar os pesos de uma outra rede neural (HA; DAI; LE, 2016), enquanto meta-aprendizado envolve usar o conhecimento adquirido do treinamento de um modelo em várias tarefas diferentes para gerar novos modelos em outras tarefas, e com isso ter uma alta capacidade de se adaptar a novas tarefas.

O HyperFast, gera um grupo de hiper redes a partir da criação de várias redes neurais treinadas em um conjunto diversificado de tarefas de classificação, englobando datasets com diferentes distribuições, tamanhos e características. Para cada novo dataset, o HyperFast recebe um conjunto de suporte, que é parte dos dados de treinamento, e, com uma única passagem pelas hiper redes, gera os pesos de uma rede principal. Além disso, o processo de otimização de hiperparâmetros é opcional, dando a possibilidade de ser uma solução de uso imediato (*off the shelf*).

Figura 1 – Arquitetura do Algoritmo HyperFast



Fonte: Bonet et al. (2024)

De maneira simplificada, o algoritmo faz o treinamento das hiper redes da seguinte maneira: uma parte dos dados de entrada, chamada de *support set*, é extraída dos dados de treinamento. As entradas passam por um módulo que faz Random Features (RF) e Principal Component Analysis (PCA) para que fiquem projetados em um espaço de características de tamanho fixo e invariante à permutação. Os dados projetados entram numa sequência de módulos de hiper redes indo de 1 a L , onde L é o número de camadas da rede principal, que geram os pesos da rede principal correspondentes a camada de seu número. Os módulos de hiper redes possuem, além de seus parâmetros específicos, parâmetros compartilhados. A partir da segunda camada, cada módulo de hiper rede recebe a representação intermediária dos dados após passar pela camada anterior da rede principal junto com as classes correspondentes, depois da aplicação de conexões residuais e não-linearidades. Para os pesos da última camada, é feito um average pooling por classe dos resultados da representação intermediária que sai da hiper rede da camada final. Por fim, um módulo que usa Nearest Neighbours é usado para adicionar parâmetros de aprendizado ao vetor de bias da camada de classificação da rede principal. Durante o treino, uma parte dos dados chamada de *query set* é usada para gerar os resultados das classificações da rede principal e depois fazer os ajustes dos parâmetros dos módulos de hiper redes.

O treino, ou meta treino, como os autores chamam, é feito usando vários conjuntos de dados distintos, com características diferentes, como número de variáveis e classes diferentes. Após o treino, o algoritmo com os pesos gerados pode ser usado para gerar uma rede neural principal a partir de conjunto de dados, passando uma vez pelos dados de treinamento desse conjunto, chamada pelos autores de meta inferência. Esse algoritmo com os pesos gerados é o disponibilizado pelos autores e o método de *fit* é a passagem dos dados pelo algoritmo para gerar a rede principal.

No artigo, os autores argumentam que o novo algoritmo apresenta vantagens sobre os outros em relação a velocidade, pois elimina a o processo iterativo de treinamento tradicional. Além disso é altamente adaptável para novos conjuntos de dados, podendo gerar classificações com rapidez e eficiência.

2.5 Trabalhos Relacionados

A seguir, apresenta-se um levantamento de estudos que abordam predição de mortalidade em diferentes contextos clínicos, variando quanto ao tipo de população, momento da coleta das variáveis e algoritmos utilizados. Esses trabalhos ilustram a diversidade de abordagens e métricas empregadas na área, evidenciando como distintos conjuntos de dados e métodos de aprendizado de máquina têm sido aplicados para antecipar desfechos críticos em ambientes hospitalares. A Tabela 1 sintetiza os principais estudos identificados, destacando seu foco, os modelos utilizados e o melhor desempenho obtido em termos de AUC.

Tabela 1 - Trabalhos Relacionados

Autor	Resumo	Melhor modelo em relação a métrica AUC
Saadatmand et al. (2022)	Predição de Mortalidade em pacientes com COVID-19	XGB, 0.928
Taylor et al. (2016)	Predição de morte em pacientes diagnosticados com sepse na admissão	Random Forest, 0.860
Raita et al.(2019)	Predição de admissão em UTI ou morte durante hospitalização	Rede neural Profunda, 0.86
Zhai et al.(2020)	Predição de morte em até 7 dias após admissão em UTIs de emergência	XGB, 0.849
Choi et al.(2022)	Predição de mortalidade durante a hospitalização	XGB e LightGBT, 0.977
Schmidt et al.(2018)	Predição de mortalidade durante a internação em uma UTI usando os primeiros dados coletados de cada variável após a entrada na UTI	Rede neural Convolucional, 0.83
Veith e Steele (2018)	Predição de morte no momento de admissão usando variáveis administrativas	LazyStar, 0.751

Fonte: Elaboração Própria

Saadatmand et al. (2022) desenvolveu modelos de predição de mortalidade em pacientes com COVID-19. Saadatmand usou dados de dois hospitais iranianos com pacientes que testaram positivo para COVID-19 para prever a morte na hospitalização. Foram 963 pacientes incluídos, sendo coletados de cada os sinais vitais, laboratoriais, sintomas, comorbidades e informações demográficas para serem usados no trabalho. Os modelos usados foram Random Forest, XGB, LogitBoost, Bagged CART e k-NearestNeighbors. Entre as métricas usadas, estão a acurácia, recall, especificidade e AUC. O modelo que apresentou melhor resultado foi o XGB com acurácia de 0.78, recall de 0.90, especificidade de 0.58 e AUC de 0.928. O modelo Random Forest teve 0.78, 1.00, 0.41, 0.868.

Taylor et al. (2016) fez modelos de previsão de morte em pacientes diagnosticados com sepse na admissão. Os dados coletados foram de pacientes de 4 hospitais por um período de 12 meses, tendo um total de 4676 pacientes. A variável a ser predita era mortalidade em 28 dias após admissão. Os modelos usados foram a árvore de decisão CART, regressão logística e RF. Além disso, os modelos foram comparados também com ferramentas de predição de mortalidade já conhecidas. O modelo Random Forest teve a melhor performance geral, com o valor AUC de 0.860.

Raita et al. (2019) fez modelos de predição de admissão em UTI ou morte durante hospitalização no momento da triagem no departamento de emergência com as variáveis comumente disponíveis nesse contexto e predição de hospitalização, com dados coletados de vários hospitais nos Estados Unidos coletados no período de 2007 a 2005 pelo National Hospital and Ambulatory Medical Care Survey. Os modelos usados foram regressão logística com regularização Lasso, Random Forest, Gradient Boosted Decision Tree (GBDT) e rede neural profunda. Para primeira predição o modelo de rede neural profunda teve AUC de 0.86, recall de 0.80 e especificidade de 0.76. O modelo de Random Forest teve AUC de 0.85, recall de 0.86 e especificidade de 0.68.

Zhai et al. (2020) gerou modelos de predição de morte em até 7 dias após admissão em UTIs de emergência. Os modelos usados foram Support Vector Machines, GBDT, XGB e regressão logística. As AUC para os respectivos modelos foram 0.794, 0.840, 0.849 e 0.822.

Choi et al. (2022) fez modelos de predição de mortalidade durante a hospitalização em UTIs com dados de dois hospitais universitários coletados entre 2006 e 2020. Usando variáveis demográficas e os piores valores de variáveis de sinais vitais e laboratoriais, vários modelos foram treinados com algumas configurações diferentes dos dados. Os melhores resultados foram obtidos com os modelos XGB, LightGBM e RF com AUCs de 0.977, 0.977 e 0.972.

Schmidt et al. (2018) fez modelos de predição de mortalidade durante a internação em uma UTI usando os primeiros dados coletados de cada variável após a entrada na UTI. Os modelos foram Redes Neurais Convolucionais usando dados do MIMIC III. A AUC do melhor modelo foi 0.83, com recall de 0.697, especificidade 0.959 e acurácia de 0.916.

Veith e Steele (2018) fizeram modelos de predição de morte no momento de admissão usando variáveis administrativas também com os dados do MIMIC III. Usando algumas configurações diferentes dos dados e os modelos baseados nos algoritmos LazyKStar, Logistic, Simple Logistic, NaïveBayes e BayesNet do software WEKA, conseguiram valores de AUC entre 0.751 e 0.689, sendo 0.751 o do algoritmo LazyKstar.

Os trabalhos apresentados compartilham o objetivo de aprimorar a predição de mortalidade em diferentes cenários clínicos, mas diferem quanto ao contexto de aplicação, às variáveis disponíveis e à complexidade dos modelos utilizados. Em comum, destacam-se a adoção de algoritmos de aprendizado de máquina que superam métodos tradicionais, como os escores de gravidade, em termos de desempenho preditivo, sobretudo quando incorporam grande volume de informações clínicas, como sinais vitais, exames laboratoriais e comorbidades. Entre os pontos

fortes, ressaltam-se a capacidade desses modelos de capturar interações não lineares entre variáveis, o uso de bases extensas como MIMIC-III e EHRs hospitalares, e a altos valores de AUC ROC alcançados pelos melhores modelos de cada trabalho. Por outro lado, algumas limitações são recorrentes: falta de detalhamento de fatores importantes como balanceamento entre as classes e foco na métrica AUC ROC na avaliação dos modelos.

Este trabalho além de usar dois tradicionais de aprendizado de máquina (XGboost e Redes Neurais MLP), usa um algoritmo recente (HyperFast). Também é feito uma análise de tempos e tamanhos dos algoritmos no treinamento e predição, algo que não foi feito pelos trabalhos citados.

3 Metodologia

Nesta seção será mostrada a metodologia usada para criar os modelos. Primeiro é apresentado o banco de dados de onde foram retirados os dados. Depois, com base nas 2 primeiras fases do CRISP-DM, entendimento do problema e entendimento de dados, será relatado o que foi estudado para compreensão do problema que queremos modelar (previsão de mortalidade) e a exploração dos dados a ser feita. Por fim, será mostrado o processo utilizado na parte de pré-processamento dos dados e como a modelagem será feita.

3.1 MIMIC IV

O Medical Information Mart for Intensive Care IV (MIMIC IV) é um conjunto de dados de pacientes críticos anonimizado com informações coletadas no hospital Beth Israel Deaconess Medical Center na cidade de Boston em Massachusetts nos Estados Unidos durante o período de 2008 até 2019 e disponível na Physionet, uma plataforma que mantém bancos de dados da área médica e mantido por membros do Laboratório para Fisiologia Computacional do Massachusetts Institute of Technology.

Os dados vieram de pacientes de mais de 18 anos que ou deram entrada no departamento de emergência do hospital, ou que deram entrada na UTI dele. Após essa coleta, os dados passaram por um processo de anonimização para estar em conformidade com a Health Insurance Portability and Accountability Act (HIPAA), que define as regras e normas para garantir a privacidade e segurança em dados eletrônicos de saúde de pacientes.

O banco é dividido em dois módulos: *hosp* e *icu*. O módulo *hosp* contém os dados coletados durante a estadia de um paciente no hospital após ser admitido do departamento de emergência. O módulo *icu* contém os dados do paciente durante sua estadia nas UTIs do hospital.

Ao todo, na sua versão 2.2, que será usada neste trabalho, o banco contém 431231 hospitalizações únicas de 299712 indivíduos no módulo *hosp* e 73181 estadias em UTI no módulo *icu*.

A tabela de eventos laboratoriais (*labevents*) do módulo *hosp* traz todos os resultados laboratoriais de um paciente durante sua estadia no hospital. O tipo do exame feito é identificado pelo campo *itemid*, e sua definição fica na tabela *d_labitems*, onde é descrito o exame (*label*), sua categoria (*category*) e fluido de coleta (*fluid*). A unidade de medição fica direto na tabela *labevents* (*valueuom*).

A tabela de eventos gerais (*chartevents*) do modulo *icu* traz todos os eventos que ocorreram com um paciente durante sua estadia em uma UTI. Isso inclui medições de sinais vitais, exames laboratoriais, mudanças de medicamento etc. Basicamente qualquer evento é registrado. O tipo do evento é identificado pelo campo *itemid*, e sua definição fica na tabela *d_items*, onde é descrito o evento (*label*), sua categoria (*category*), unidade de medição quando é aplicável (*valueuom*) e algumas outras informações.

Neste trabalho são usadas variáveis dos pacientes coletadas nas primeiras 24 horas da primeira estadia na UTI de uma admissão no hospital. Essas variáveis são os sinais vitais e exames laboratoriais, além da idade, gênero, os valores mínimos de cada componente do Glasgow Coma Score e o escore usando estes valores, uso de ventilação mecânica nas primeiras 24 horas, uso de ventilação mecânica invasiva nas primeiras 24 horas e o tipo de admissão na UTI. Elas foram escolhidas a partir dos escores de gravidade e de outros trabalhos relevantes, além de explorações no banco de dados. As variáveis selecionadas do banco estão na Tabela 2, junto com os limites máximos e mínimos dos valores que são possíveis segundo. As variáveis com *itemid* igual a None foram criadas a partir de outras no banco. As variáveis de sinais vitais serão agregadas e usadas as médias, os máximos, os mínimos e os desvios padrões das medições. As variáveis laboratoriais somente médias, os máximos e os mínimos.

As variáveis de sinais vitais foram: batimentos cardíacos, frequência respiratória, pressão arterial sistólica, pressão arterial diastólica, pressão arterial média, saturação de oxigênio medida por oximetria de pulso, temperatura corporal, peso na admissão.

As variáveis de exames laboratoriais as variáveis foram: hematócritos, hemoglobina, contagem de células brancas, albumina, proteína, proteína c-reativa, hiato aniônico, bicarbonato, cálcio total, cálcio livre, creatinina, cloreto, glicose, potássio, sódio, nitrogênio ureico no sangue, magnésio, fosfato, lactato, pO2 (pressão parcial de oxigênio no sangue), pCO2 (pressão parcial de dióxido de carbono no sangue) e bilirrubina total.

Os valores mínimos e máximos possíveis das variáveis foram decididos a partir de pesquisas sobre cada uma delas.

Tabela 2 - Variáveis escolhidas do MIMIC - IV

itemid	Variável	Unidade	Mínimo Possível	Máximo Possível	Tipo da variável
51221	Hematocrit	%	4.0	66.0	numérica
51222	Hemoglobin	g/dL	1.2	30.0	numérica
51301	WhiteBloodCells	K/uL	0.1	1000.0	numérica

50862	Albumin	g/dL	2.0	7.0	numérica
51492	Protein	mg/dL	0.0	700.0	numérica
50889	C-Reactive Protein	mg/L	0.0	700.0	numérica
50868	AnionGap	mEq/L	-10.0	60.0	numérica
50882	Bicarbonate	mEq/L	1.0	55.0	numérica
50893	CalciumTotal	mg/dL	1.0	60.0	numérica
50808	FreeCalcium	mmol/L	0.1	45.0	numérica
50912	Creatinine	mg/dL	0.01	73.8	numérica
50902	Chloride	mEq/L	40.0	160.0	numérica
50931	Glucose	mg/dL	3.0	2656.0	numérica
50971	Potassium	mEq/L	0.7	14.0	numérica
50983	Sodium	mEq/L	78.0	200.0	numérica
51006	UreaNitrogen	mg/dL	1.0	300.0	numérica
50960	Magnesium	mg/dL	0.1	30.0	numérica
50970	Phosphate	mg/dL	0.1	60.0	numérica
50813	Lactate	mmol/L	0.0	50.0	numérica
50821	pO2	mm Hg	30.0	700.0	numérica
50818	pCO2	mm Hg	10.0	150.0	numérica
50885	BilirubinTotal	mg/dL	0.01	80.0	numérica
51265	PlateletCount	K/uL	1.0	3000.0	numérica
220045	HeartRate	bpm	25.0	700.0	numérica
220210	RespiratoryRate	insp/min	1.0	70.0	numérica
220050	ArterialBloodPressureSystolic	mmHg	30.0	400.0	numérica
220179	NonInvasiveBloodPressureSystolic	mmHg	30.0	400.0	numérica
220051	ArterialBloodPressureDiastolic	mmHg	10.0	380.0	numérica
220180	NonInvasiveBloodPressureDiastolic	mmHg	10.0	380.0	numérica
220052	ArterialBloodPressureMean	mmHg	20.0	390.0	numérica
220181	NonInvasiveBloodPressureMean	mmHg	20.0	390.0	numérica
220277	O2SaturationPulseoxymetry	%	40.0	100.0	numérica
223761	TemperatureFahrenheit	°F	50.0	116.0	numérica
223762	TemperatureCelsius	°C	10.0	46.7	numérica
None	age	years	18.0	92.0	numérica
226512	Weight_adm	kg	2.00	600.0	numérica

None	gender	None	0	1	categórica
220739	gcs_eye_min	None	1.0	4.0	numérica
223900	gcs_verbal_min	None	1.0	5.0	numérica
223901	gcs_motor_min	None	1.0	6.0	numérica
None	GCS_total_min	<u>None</u>	3.0	15.0	numérica
None	any_vent_first_24h	None	0	1	categórica
None	invasive_vent_first_24h	None	0	1	categórica
None	admission_type	None	0	8	categórica

Fonte: Elaboração própria

3.2 Entendimento do Problema

Para ter uma maior compreensão do problema proposto para a modelagem, além dos estudos dos artigos já mencionados na seção de Trabalhos Relacionados, outros trabalhos, artigos e documentos foram estudados a fim compreender melhor o contexto do problema. Também foram feitas conversas com pessoas inseridas na área de UTIs, como enfermeiros e médicos intensivistas.

Além de estudos específicos de predição de mortalidade, estudos que fazem outros tipos de previsão dentro do contexto de UTI e departamento de emergência foram lidos, como estudos para prever a necessidade de uma UTI ou admissão em hospital, tempo de estadia do paciente na UTI e desenvolvimento de quadros clínicos específicos.

Estudos que usam técnicas de ML não supervisionadas como o feito por Vranas et al. (2017) em que se é usado o algoritmo de *Cluster Consensus* para identificar subgrupos com características similares em pacientes de UTI, ajudaram a entender um pouco melhor as possíveis variáveis a serem usadas.

Outro tópico importante são os índices de gravidade do paciente. Estes índices são valores e classificações que avaliam o quão grave o paciente está e são muito utilizados na decisão de mandar um paciente para um leito de UTI, além de serem usados para fazer a predição de morte (KEEGAN; SOARES, 2016) e tempo de estadia em UTI. Entre eles, se destacam o Simplified Acute Physiology Score (SAPS), o Acute Physiology and Chronic Health Disease Classification System (APACHE) e o Sequential Organ Failure Assessment (SOFA). Cada um deles tem seu próprio grupo de variáveis usadas para gerar o índice, que também vão ajudar a fazer a escolha das variáveis a serem usadas nos modelos.

Durante o entendimento do problema, foram pré selecionadas as variáveis de sinais vitais e laboratoriais de interesse.

Também foi feito um estudo buscando definir quais são os intervalos possíveis de que essas variáveis podem ter. Essas informações são mostradas na Tabela A nos apêndices.

3.3 Implementação

O código foi desenvolvido na linguagem Python versão 3.11 usando o ambiente Jupyter Notebook. Para a manipulação das tabelas de dados foi usado a biblioteca Pandas. A biblioteca Scikit-learn versão 1.14 de ML também foi usada para fazer as análises e gerar o modelo MLP. A biblioteca Imbalanced-learn foi usada para fazer balanceamento. O modelo XGB usou a biblioteca XGBoost e o HyperFast, o código disponibilizado pelos autores no GitHub disponível no link <https://github.com/AI-sandbox/HyperFast>. O código disponibilizado gera uma rede neural de 3 camadas com 784 perceptrons cada e possui parâmetros que fazem ensemble de redes geradas pelo algoritmo, além de parâmetros para fazer *fine tuning* da rede gerada. Neste trabalho não foi considerado o uso do parâmetro para fazer várias redes e agrupá-las usando ensemble. Para as consultas no banco de dados, foi usado a plataforma BigQuery.

A máquina usada foi uma máquina virtual do Google Colab, com processador Intel(R) Xeon(R) CPU @ 2.20GHz com 12 cores, GPU Nvidia L4 com 24 GB de memória e memória RAM de 53 GB.

3.4 Entendimento dos Dados

No processo de entendimento de dados, foram feitas várias análises estatísticas e gráficas a fim de ter uma melhor compreensão de como os dados estão em relação a várias questões. A seguir são mostradas algumas das análises que foram consideradas importantes.

3.4.1 Exploração direto no banco usando consultas SQL

No primeiro momento, foi feito uma exploração das tabelas de eventos laboratoriais e eventos gerais do paciente durante sua estadia na UTI, para identificar as variáveis usando consultas SQL direto no banco usando a plataforma.

Por vezes mais de um *itemid* aparece para as variáveis escolhidas, onde pode haver mudança entre algum outro campo, como a categoria do exame ou fluido. Este fato pode ser visto na Tabela 3.

Tabela 3 - Consulta da variável Urea Nitrogen no banco

itemid	label	fluid	category	valueuom	Contagem de registros na tabela de labevents
51951	Urea Nitrogen, Stool	Stool	Chemistry	Null	1
51104	Urea Nitrogen, Urine	Urine	Chemistry	mg/dL	34033
51825	Urea Nitrogen, Joint Fluid	Joint Fluid	Chemistry	Null	1
51804	Urea Nitrogen, CSF	Cerebrospinal Fluid	Chemistry	Null	1
50851	Urea Nitrogen, Ascites	Ascites	Chemistry	mg/dL	21
51842	Bun	Other Body Fluid	Chemistry	Null	1
51922	Urea Nitrogen, Pleural	Pleural	Chemistry	Null	1
51006	Urea Nitrogen	Blood	Chemistry	mg/dL	3189474
51045	Urea Nitrogen, Body Fluid	Other Body Fluid	Chemistry	mg/dL	51
52647	Urea Nitrogen	Blood	Chemistry	mg/dL	369

Fonte: Elaboração Própria

Nesse exemplo podemos ver que para a variável *Nitrogênio da ureia* (Urea Nitrogen) temos vários registros, que em alguns casos são idênticos, como os itemids de valor 51006 e 52647, e várias variações onde o nome muda um pouco. Dado que o banco não possui uma explicação mais aprofundada do exame, onde podemos saber se os exames são equivalentes, optou-se por usar aquele que possui maior quantidade de registros, ou seja, maior número de coletas.

De maneira similar foi feito o mesmo para os sinais vitais. Aqui se optou-se por pegar medição de pressão arterial e pressão arterial através de método não invasivo e temperatura em Celsius e Fahrenheit.

Observou-se também por outras consultas que o banco não possui estadias de pacientes com idade inferior a 18 anos ou maior que 91. O fato de não haver maior que 91 anos é pelo processo de anonimização do MIMIC-IV, que atribui a todo paciente com idade maior que 91 essa idade.

3.4.2 Extração dos Dados

Para extração dos dados laboratoriais do banco, foi usada a consulta em SQL mostrada no Apêndice A.

Nesta consulta, registros de estadia onde a idade é menor que de 18 anos já são descartados pela consulta.

Usou-se essa tabela para ver a média do número de medições de cada exame laboratorial no intervalo das primeiras 24 horas da primeira UTI (Tabela 4).

Tabela 4 - Número médio de medições de cada variável laboratorial nas primeiras 24 horas

Variável	Número de medições
Hematócritos	2,344
Hemoglobina	2,029
Contagem de células brancas	1,999
Albumina	0,326
Proteínas	0,239
Proteínas C-Reativas	0,024
Hiato aniônico	2,000
Bicarbonato	2,022
Cálcio total	1,741
Cálcio livre	1,184
Creatinina	2,033
Cloreto	2,113
Glicose	1,903
Potássio	2,121
Sódio	2,082

Nitrogênio ureico no sangue	2,028
Magnésio	1,852
Fósforo	1,748
Lactato	1,499
pO2 (pressão parcial do oxigênio no sangue)	2,133
pCO2 (pressão parcial do dióxido de carbono no sangue)	2,132
Bilirrubina total	0,591
Contagem de Plaquetas	2,041

Fonte: Elaboração própria

A tabela da consulta depois foi transformada em uma tabela com valores agregados de cada estadia, trazendo o máximo, mínimo e média de cada variável por estadia. O desvio padrão não é coletado pois como visto, os exames são coletados e no máximo em torno de duas vezes nas primeiras 24 horas.

Em virtude do tamanho da tabela de dados e a quantidade de dados processados na tabela *chart_events*, onde estão os valores de sinais vitais, a consulta para extrair os sinais vitais já traz os dados agregados de cada estadia por valor mínimo, máximo, média e desvio padrão. A consulta está disponível no Apêndice B. Além disso, as medições que estão nas seis horas que antecedem o óbito de um paciente foram desconsideradas.

As duas tabelas foram unificadas pelo campo *stay id*.

3.4.3 Explorações adicionais

As informações de número de campos preenchidos da tabela com as agregações de todas as variáveis de cada estadia, além de outras variáveis importantes que foram usadas para criar novas variáveis, usando o comando *info()* do Pandas estão na Tabela 5.

Tabela 5 - Número de valores nulos e não nulos de cada variáveis da tabela gerada pelas consultas

Coluna	Valores não nulos	Valores nulos	dtype
subject_id	65756	0	int64
hadm_id	65756	0	int64
stay_id	65756	0	int64
icu_intime	65756	0	datetime64[ns]

icu_outtime	65756	0	object
hosp_outtime	65756	0	object
deathtime	6543	59213	datetime64[ns]
admission_type	65756	0	object
hospital_expire_flag	65756	0	int64
rn	65756	0	int64
gender	65756	0	object
anchor_age	65756	0	int64
anchor_year	65756	0	int64
dod	24305	41451	object
HeartRate_min	65748	8	float64
HeartRate_max	65748	8	float64
HeartRate_std	65656	100	float64
HeartRate_mean	65748	8	float64
RespiratoryRate_min	65652	104	float64
RespiratoryRate_max	65652	104	float64
RespiratoryRate_std	65522	234	float64
RespiratoryRate_mean	65652	104	float64
ArterialBloodPressureSystolic_min	22580	43176	float64
ArterialBloodPressureSystolic_max	22580	43176	float64
ArterialBloodPressureSystolic_std	22391	43365	float64
ArterialBloodPressureSystolic_mean	22580	43176	float64
NonInvasiveBloodPressureSystolic_min	60441	5315	float64
NonInvasiveBloodPressureSystolic_max	60441	5315	float64
NonInvasiveBloodPressureSystolic_std	58887	6869	float64
NonInvasiveBloodPressureSystolic_mean	60441	5315	float64
ArterialBloodPressureDiastolic_min	22592	43164	float64
ArterialBloodPressureDiastolic_max	22592	43164	float64
ArterialBloodPressureDiastolic_std	22392	43364	float64
ArterialBloodPressureDiastolic_mean	22592	43164	float64
NonInvasiveBloodPressureDiastolic_min	60433	5323	float64
NonInvasiveBloodPressureDiastolic_max	60433	5323	float64
NonInvasiveBloodPressureDiastolic_std	58879	6877	float64
NonInvasiveBloodPressureDiastolic_mean	60433	5323	float64
ArterialBloodPressureMean_min	22661	43095	float64

ArterialBloodPressureMean_max	22661	43095	float64
ArterialBloodPressureMean_std	22446	43310	float64
ArterialBloodPressureMean_mean	22661	43095	float64
NonInvasiveBloodPressureMean_min	60424	5332	float64
NonInvasiveBloodPressureMean_max	60424	5332	float64
NonInvasiveBloodPressureMean_std	58865	6891	float64
NonInvasiveBloodPressureMean_mean	60424	5332	float64
O2SaturationPulseoxymetry_min	65707	49	float64
O2SaturationPulseoxymetry_max	65707	49	float64
O2SaturationPulseoxymetry_std	65602	154	float64
O2SaturationPulseoxymetry_mean	65707	49	float64
TemperatureFahrenheit_min	61612	4144	float64
TemperatureFahrenheit_max	61612	4144	float64
TemperatureFahrenheit_std	59503	6253	float64
TemperatureFahrenheit_mean	61612	4144	float64
TemperatureCelsius_min	5040	60716	float64
TemperatureCelsius_max	5040	60716	float64
TemperatureCelsius_std	4721	61035	float64
TemperatureCelsius_mean	5040	60716	float64
Weight_adm	1870	63886	float64
gcs_eye_min	65519	237	float64
gcs_verbal_min	65498	258	float64
gcs_motor_min	65491	265	float64
GCS_total_min	65464	292	float64
any_vent_first_24h	65756	0	int64
invasive_vent_first_24h	65756	0	int64
Hematocrit_min	63744	2012	float64
Hematocrit_max	63744	2012	float64
Hematocrit_mean	63744	2012	float64
Hemoglobin_min	63541	2215	float64
Hemoglobin_max	63541	2215	float64
Hemoglobin_mean	63541	2215	float64
WhiteBloodCells_min	63549	2207	float64
WhiteBloodCells_max	63549	2207	float64

WhiteBloodCells_mean	63549	2207	float64
Albumin_min	17286	48470	float64
Albumin_max	17286	48470	float64
Albumin_mean	17286	48470	float64
Protein_min	7172	58584	float64
Protein_max	7172	58584	float64
Protein_mean	7172	58584	float64
C-ReactiveProtein_min	1343	64413	float64
C-ReactiveProtein_max	1343	64413	float64
C-ReactiveProtein_mean	1343	64413	float64
AnionGap_min	63883	1873	float64
AnionGap_max	63883	1873	float64
AnionGap_mean	63883	1873	float64
Bicarbonate_min	63950	1806	float64
Bicarbonate_max	63950	1806	float64
Bicarbonate_mean	63950	1806	float64
CalciumTotal_min	59075	6681	float64
CalciumTotal_max	59075	6681	float64
CalciumTotal_mean	59075	6681	float64
FreeCalcium_min	23647	42109	float64
FreeCalcium_max	23647	42109	float64
FreeCalcium_mean	23647	42109	float64
Creatinine_min	63966	1790	float64
Creatinine_max	63966	1790	float64
Creatinine_mean	63966	1790	float64
Chloride_min	63992	1764	float64
Chloride_max	63992	1764	float64
Chloride_mean	63992	1764	float64
Glucose_min	63713	2043	float64
Glucose_max	63713	2043	float64
Glucose_mean	63713	2043	float64
Potassium_min	63967	1789	float64
Potassium_max	63967	1789	float64
Potassium_mean	63967	1789	float64
Sodium_min	63990	1766	float64
Sodium_max	63990	1766	float64
Sodium_mean	63990	1766	float64
UreaNitrogen_min	63947	1809	float64
UreaNitrogen_max	63947	1809	float64
UreaNitrogen_mean	63947	1809	float64
Magnesium_min	61602	4154	float64
Magnesium_max	61602	4154	float64
Magnesium_mean	61602	4154	float64

Phosphate_min	59220	6536	float64
Phosphate_max	59220	6536	float64
Phosphate_mean	59220	6536	float64
Lactate_min	35048	30708	float64
Lactate_max	35048	30708	float64
Lactate_mean	35048	30708	float64
pO2_min	34708	31048	float64
pO2_max	34708	31048	float64
pO2_mean	34708	31048	float64
pCO2_min	34704	31052	float64
pCO2_max	34704	31052	float64
pCO2_mean	34704	31052	float64
BilirubinTotal_min	26849	38907	float64
BilirubinTotal_max	26849	38907	float64
BilirubinTotal_mean	26849	38907	float64
PlateletCount_min	63565	2191	float64
PlateletCount_max	63565	2191	float64
PlateletCount_mean	63565	2191	float64

Fonte: Elaboração própria

Explorações gráficas foram feitas, como histogramas para ver as distribuições das variáveis para todos os registros, registros onde a morte ocorreu na admissão e onde a morte não ocorreu. Os gráficos para a variável idade (*anchor_age*) são mostrados na Figura 2 como exemplo.

Um problema encontrado ao fazer esses gráficos é quando a variável possui valores muito extremos, que faz com que os intervalos dos histogramas fiquem muito grandes e os valores muito concentrados em um ou dois intervalos. Isso já indica que há presença de valores outliers nos dados. A variável da média de batimentos cardíacos (*HeartRate_mean*) é um exemplo desse fenômeno, como pode ser visto na Figura 3.

A fim de fazer uma análise que mostre melhor as distribuições, foi usado método interquartil para retirar os valores extremos para fazer mais gráficos. Na figura 4 temos os mesmos gráficos para a mesma variável Figura 3.

Outra análise interessante foi a da contagem de mortes por dia a partir da entrada na UTI. Podemos ver pela Figura 5 que a grande parte das mortes ocorreu na primeira semana. No total, a quantidade pessoas que morreram após a primeira entrada na UTI no hospital é 6543, que morreram após sete dias contados a partir da primeira entrada na UTI no hospital é 2520 e que morreram entre a entrada e os primeiros sete dias é 4023.

Figura 2 - Histogramas da variável anchor_age

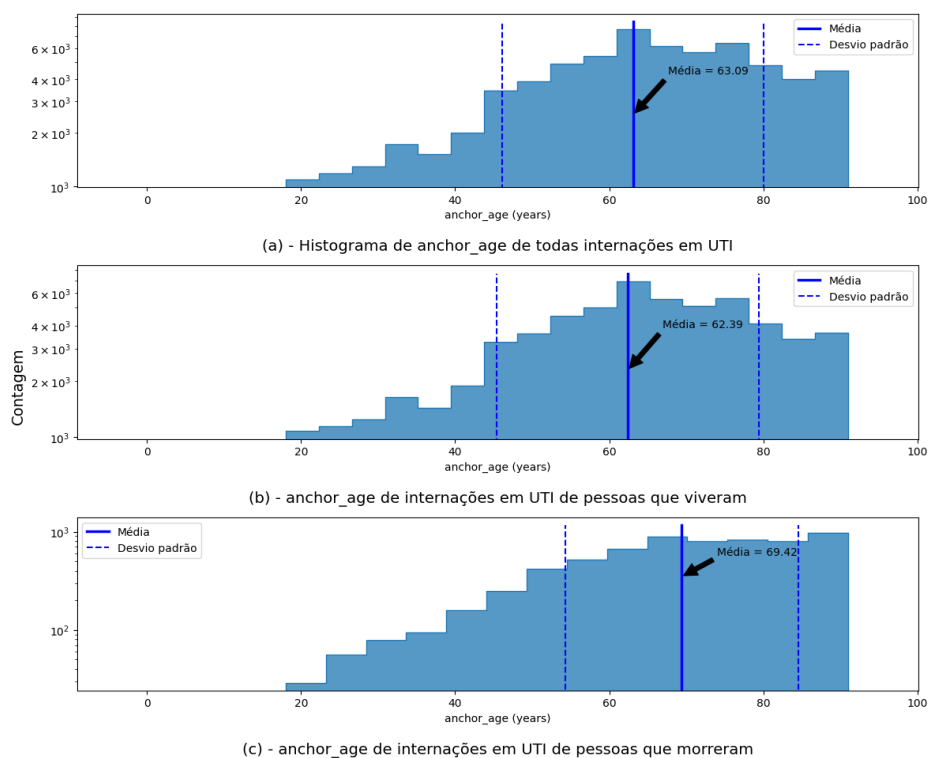


Figura 3 - Histogramas da variável HeartMean_mean

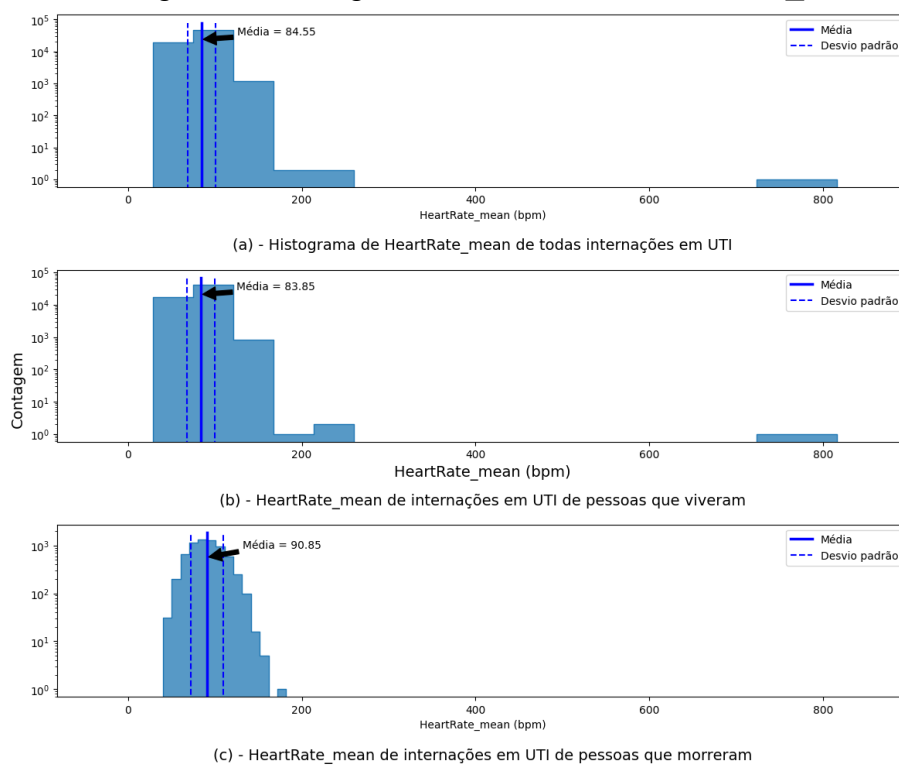
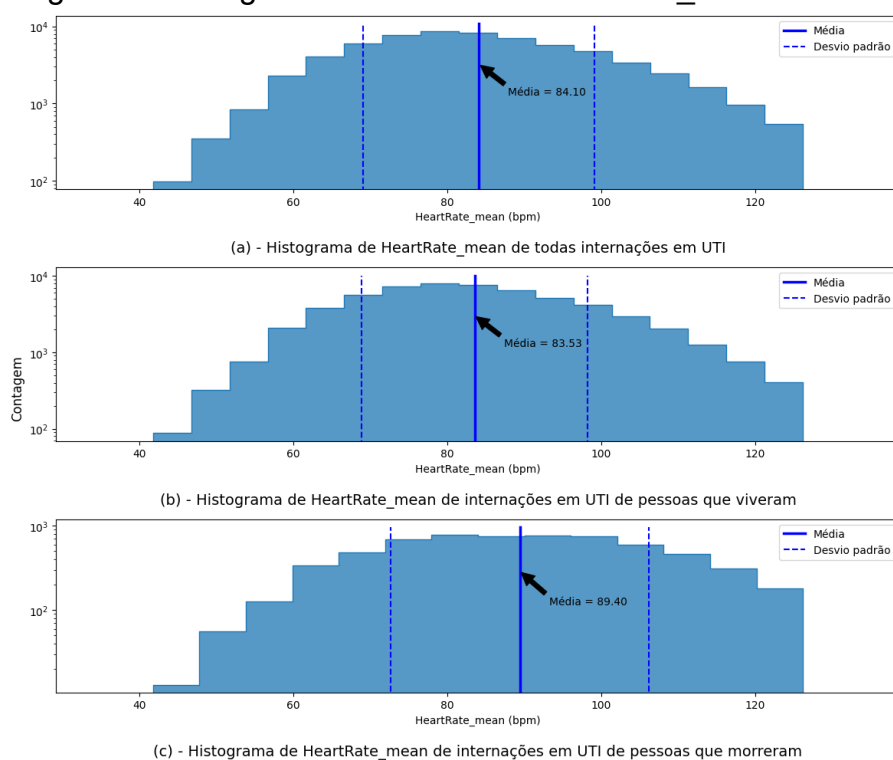
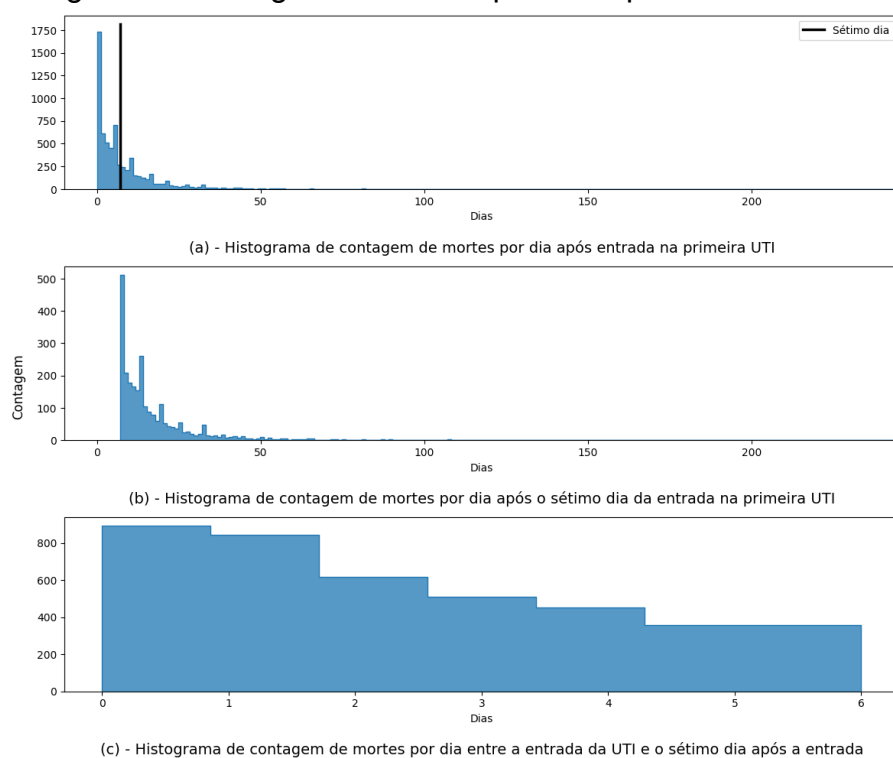


Figura 4 - Histogramas da variável HeartMean_mean sem extremos



Fonte: Elaboração Própria

Figura 5 - Contagem de mortes por dia a partir da entrada na UTI



Fonte: Elaboração própria

3.5 Pré-processamento dos Dados

Nesta parte mostraremos o processo usado para preparar os dados para serem usados nos modelos escolhidos.

Primeiro foi criado uma coluna única para as variáveis de pressão, onde quando há medição de pressão não invasiva, esse valor é usado, na falta deste, a medição invasiva é usada. A razão é que a pressão não invasiva tem um número maior de preenchimentos, que pode ser visto na Tabela 5. Para temperatura uma única coluna foi feita e usando a medição em Fahrenheit que é a mais preenchida, e na falta, a temperatura em Celsius convertida para Fahrenheit.

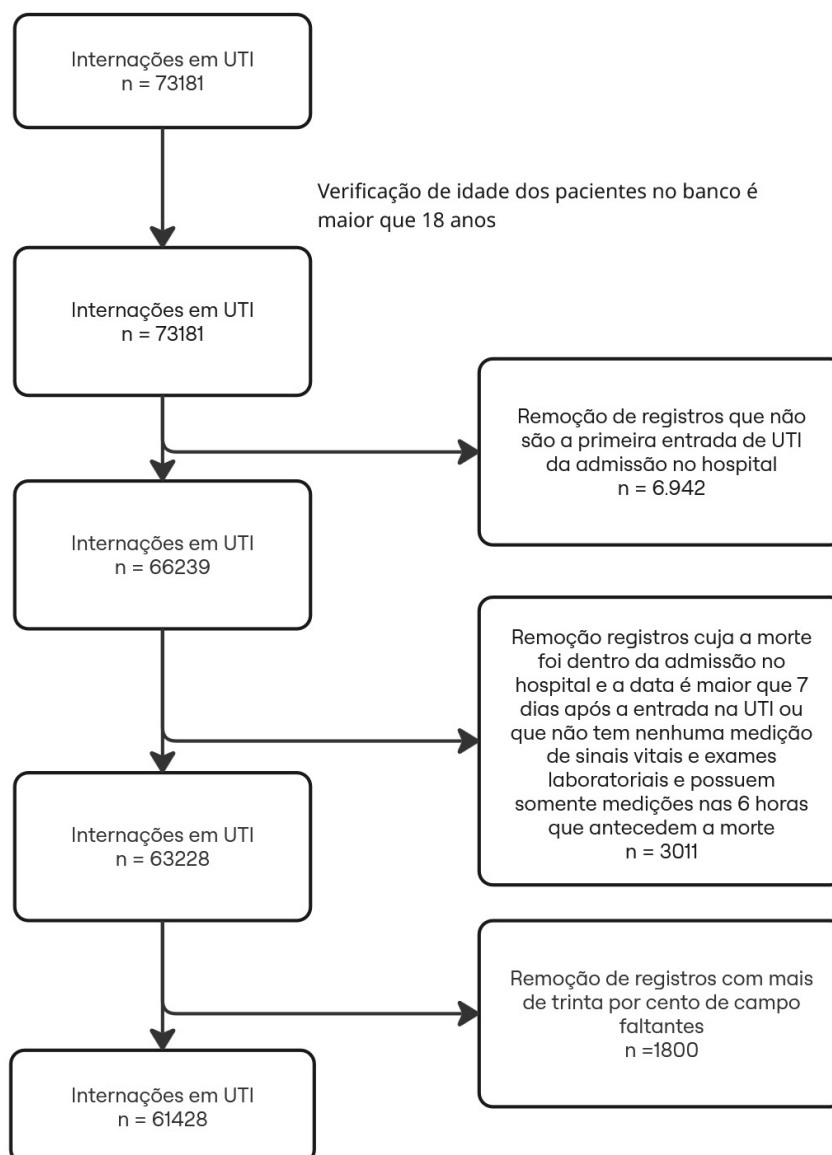
Por fim, para os registros onde o campo de desvio padrão é nulo, mas temos média, máximo e mínimo preenchido, o que significa que houve somente uma medição daquela variável, o desvio foi colocado como zero.

A idade do paciente no momento da admissão foi extraída usando os campos *anchor_age*, *anchor_year* e *icu_intime*.

Após isso, a variável de interesse, morte em até 7 dias dentro do hospital após a primeira entrada na UTI, foi extraída usando os campos *deathtime*, *hospital_expire_flag* e *icu_intime*.

As variáveis que estavam vazias para mais de cinquenta por cento dos pacientes foram descartadas. Outros campos como *subjec_id* que não são pertinentes para os modelos foram retirados. Com isso o grupo de variáveis finais foram mostrados nas tabelas A2 e A3 nos apêndices D e E respectivamente, com a análise estatística de cada variável no conjunto.

Figura 6 - Fluxo de exclusão de dados



Fonte: Elaboração própria

3.5.1 Exclusão de Dados

O processo de exclusão de dados seguiu o seguinte fluxo (Figura 6): retiradas de registros com idade menor que 18 anos, retirada de registros que não são da primeira entrada de UTI da admissão, retirada de registros cuja morte foi dentro da admissão no hospital e a data é maior que 7 dias após a entrada na UTI ou que não possui nenhuma medida laboratorial ou de sinais vitais, e retirada de registros com mais de trinta por cento de campos faltantes. Além disso, estamos desconsiderando medidas que foram coletadas nas 6 horas que antecedem a morte de um paciente, os chamados dados agonais. Embora não exista um consenso sobre o período agonal, a remoção de dados entre 4 a 12 horas anteriores ao evento é usada na literatura de ML clínico (BOUVAREL, CARRAT, LAPIDUS; 2023).

3.5.2 Criação dos conjuntos

Para as análises, quatro conjuntos de dados foram criados. O primeiro conjunto, chamado D1, são os dados da maneira como estão, sem eliminação de outliers e sem fazer balanceamento de classes nos dados de treino. Valores nulos foram imputados usando o *SimpleImputer* do scikit-learn, um imputador que preenche os dados faltantes com o valor definido pelo usuário, que neste trabalho foi a mediana da coluna. Os dados foram separados em treino e teste aqui, seguindo a proporção 70% para treino e 30% para teste e também garantindo que a proporção entre as classes foi a mesma do dataset inteiro nos dois conjuntos. A proporção após a eliminação dos registros ficou em 6,02% para a classe positiva e 93,98% para a classe negativa.

O segundo conjunto, D2, passou pela remoção de medições com valores outliers para calcular máximos, mínimos, médias e desvios. Esses outliers são os que ficam fora do intervalos de máximos e mínimos possíveis. Também foi garantido que todas as estadias em D1 sejam as mesmas em D2, porém com os valores das variáveis diferentes em virtude da retirada de medições outliers. Depois foi feita a imputação de valores faltantes usando o *SimpleImputer* usando a mediana da coluna como valor imputado para as variáveis numéricas, visto que as variáveis categóricas estavam completamente preenchidas. Nesse estudo, os valores limites para dizer se um valor é ou não outlier, foram escolhidos manualmente a partir do que se julga como um valor possível ou não daquela variável, de acordo com a literatura. Após a exclusão e das medidas outliers, foram retirados 252 registros por ficarem com mais 30% de campos nulos.

O conjunto D3, que vem de D1, não terá a eliminação de outliers e o balanceamento de classes no conjunto de treino será feito. O balanceamento é feito usando a técnica de *undersampling*, onde retiramos valores das classes majoritárias até atingir uma proporção desejada entre as classes. Aqui essa proporção escolhida foi de 1 para 1.

O conjunto D4, feito a partir de D2, passa pela eliminação de outliers e balanceamento de classes no conjunto de treino.

As variáveis categóricas foram transformadas em numéricas usando o *get_dummies* do Pandas que faz *one hot encoding* delas e os dados de cada conjunto foi dividido em treino e teste para seguindo a proporção 70 por cento de treino e trinta por cento para teste, mantendo a proporção de classes dos dados originais em cada conjunto de treino e teste. Foi garantido que um dado que aparece no conjunto de treino de um conjunto não apareça no conjunto de teste de outro. As variáveis numéricas foram normalizadas usando o *MinMaxScaler* do scikit-learn usando como base os dados de treino para ajustar o normalizador após as imputações.

Visto que a falta de uma variável, em especial as variáveis laboratoriais que são coletadas no em torno de 2 vezes no máximo nas primeiras 24 horas, pode não ser um valor faltante aleatório (*Missing Completely at Random* ou *Missing at Random*) e ter chance de possuir um significado clínico, para cada variável foi criada uma coluna que indica se para aquele registro, determinada variável estava faltante e foi imputada. Por exemplo, um exame de Lactato, que avalia doença aguda, não ser feito pode indicar quadro estável do paciente.

A imputação de valores gerou diferença mínima nas médias e desvios padrões das variáveis como podemos ver olhando as tabelas A4 e A5 dos apêndices F e G respectivamente.

3.6 Modelagem

Para a modelagem, foi usado o método do scikit-learn *GridSearchCV*, que recebe grupos de vários parâmetros ou grids de um determinado modelo, cada um com opções de valores para serem usados, e faz validação cruzada para cada combinação dos diferentes parâmetros do modelo que será treinado. Ao final ele traz os tempos de treino em cada fold, as médias dos tempos das validações, valor das métricas passadas ao método de cada treino e as médias da validação para cada combinação dos parâmetros. O número de folds escolhidos para a validação cruzada do *GridSearchCV* foi 5. A fim de evitar *data leakage*, os conjuntos não foram passados para o *GridSearchCV*. Foram criados 2 *pipelines*, um que faz imputação e normalização, o outro imputação + normalização + undersampling. Os conjuntos D1 e D2 sem imputação e normalização, foram então passados para cada *pipeline*, com o intuito de achar os melhores hiperparâmetros para cada algoritmo em cada conjunto correspondente. Por exemplo, usar D2 no segundo *pipeline* para gerar achar os hiperparâmetros de do conjunto D4. Foram utilizados dois pipelines porque cada método de modelagem requer um fluxo de pré-processamento distinto: um pipeline realizando apenas imputação e normalização, e outro adicionando a etapa de *undersampling*. Incorporar essas etapas dentro dos pipelines garante que todas as transformações sejam executadas somente sobre os folds de treino dentro da validação cruzada.

Com o intuito de avaliar melhor o algoritmo MLP em relação ao HyperFast, fizemos um grid de parâmetros com redes menores para o MLP e um grid com uma arquitetura similar em relação ao número de parâmetros da rede gerada pelo HyperFast(MLP-Hyper). Para o HyperFast usamos dois grids com os mesmos parâmetros, um para rodar em CPU igual ao algoritmo MLP do scikit-learn e o outro em GPU. No HyperFast, o parâmetro '*optimization*' tem valor True como padrão. Isso faz com que após a geração da rede principal, ela passe por *fine tuning* com passos definidos pelo parâmetro '*optimization_steps*'. Por isso, um segundo grid para o

HyperFast em cada tipo de processador foi usado com '*optimization*' como None, o que possibilita avaliar a rede principal gerada sem nenhuma modificação posterior. Os grids usados estão mostrados na Tabela 6.

Tabela 6 - Grids usados para os modelos

Modelo	Grid
XGBoost	<code>'max_depth': [3, 7, 5],</code> <code>'eta': [0.3, 0.1],</code> <code>'subsample': [0.5, 0.7, 1]</code> <code>'n_estimators': [50, 100, 200, 500]</code>
MLP	<code>'hidden_layer_sizes': [(100, 100,), (50, 100, 50),],</code> <code>'max_iter': [20, 50, 100],</code> <code>'learning_rate_init': [0.01, 0.001, 0.0001],</code> <code>'early_stopping': [True]</code>
MLP com mesma arquitetura da rede gerado pelo HyperFast(MLP-Hyper)	<code>hidden_layer_sizes': [(1024, 1024)],</code> <code>'max_iter': [50, 100],</code> <code>'learning_rate_init': [0.01, 0.001, 0.0001],</code> <code>'early_stopping': [True]</code>
HyperFast em CPU	<code>'n_ensemble': [1],</code> <code>'batch_size': [1024, 2048],</code> <code>'optimize_steps': [1, 4, 8, 16]</code>
HyperFast em CPU sem fine tuning	<code>'n_ensemble': [1],</code> <code>'batch_size': [1024, 2048],</code> <code>'optimization': [None]</code>
HyperFast em GPU	<code>'n_ensemble': [1]</code> <code>'batch_size': [1024, 2048],</code> <code>'optimize_steps': [1, 4, 8, 16]</code>
HyperFast em GPU sem fine tuning	<code>'n_ensemble': [1],</code> <code>'batch_size': [1024, 2048],</code> <code>'optimization': [None]</code>

Fonte: Elaboração Própria

Após o uso do *GridSearchCV* os melhores hiperparâmetros de cada grid relacionado a cada conjunto foram selecionados, usando a métrica AUPRC, para serem treinados em todos os dados de treino de cada conjunto e serem avaliados usando os dados de teste de cada conjunto. Foi usado o método de *Bootstrap com repetição* 2000 vezes para gerar o intervalo de confiança das métricas geradas pelos modelos no conjunto de teste.

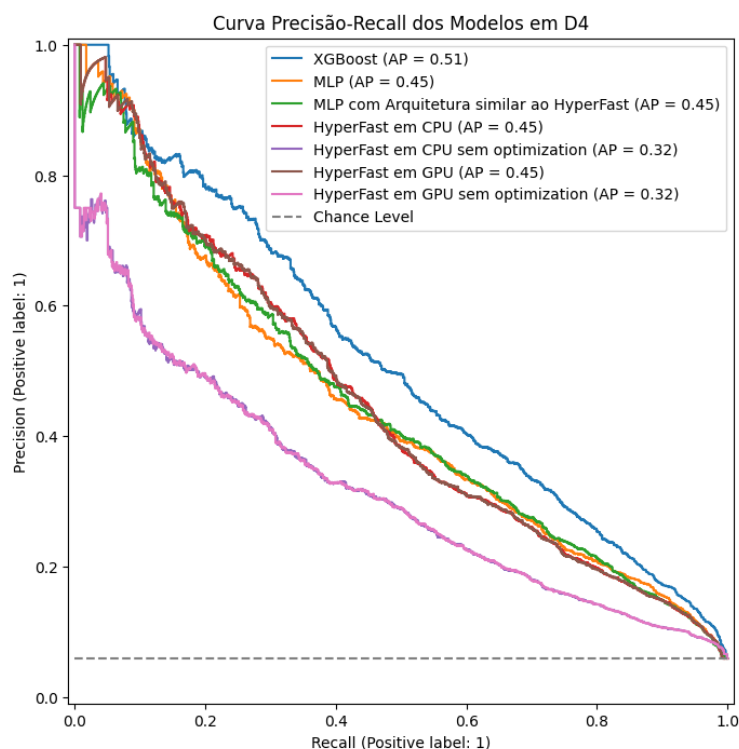
4 Resultados e Discussões

Os melhores hiperparâmetros achados pelo *GridSearchCV*, os resultados de AUPRC, junto com o intervalo de confiança de 95% calculado usando dois desvios padrões estão apresentados na tabela A6 no apêndice H. Os melhores parâmetros foram usados, então, para treinar os modelos e depois usados nos conjuntos de teste.

O conjunto de dados de treinamento com melhor resultado, tendo como ênfase a AUPRC e acurácia balanceada (*Balanced Accuracy*), que é equivalente ao Recall Macro, foi o conjunto D4. Os resultados podem ser vistos na Tabela 7 enquanto os resultados para os outros conjuntos estão nas tabelas A7, A8 e A9 no apêndice I. Levando em consideração os resultados dos HyperFast em todos os conjuntos, é possível notar que o HyperFast foi sensível ao processo de remoção dos valores outliers e ao balanceamento das classes no momento de treinar o modelo.

Foi feito também, para cada modelo em cada conjunto, o gráfico da curva Precision Recall, cujo somente do HyperFast com otimização em D4 será mostrado como exemplo (Figura 7).

Figura 7 - Gráfico da curva Precisão-Recall para os modelos no conjunto D4



Fonte: Elaboração própria

Tabela 7 - Resultados dos modelos no conjunto de teste de D4

Modelo	AUC ROC	AUPRC	Balanced Accuracy	Precision Macro
XGBoost	0,9094 (IC95%: 0,9001 – 0,9176)	0,5134 (IC95%: 0,4811 – 0,5438)	0,8292 (IC95%: 0,8172 – 0,8410)	0,6097 (IC95%: 0,6028 – 0,6168)
MLP	0,8832 (IC95%: 0,8725 – 0,8920)	0,4421 (IC95%: 0,4117 – 0,4718)	0,7972 (IC95%: 0,7847 – 0,8084)	0,5927 (IC95%: 0,5866 – 0,5984)
MLP com arquitetura similar ao HyperFast	0,8860 (IC95%: 0,8752 – 0,8953)	0,4392 (IC95%: 0,4080 – 0,4730)	0,8022 (IC95%: 0,7889 – 0,8151)	0,6063 (IC95%: 0,5992 – 0,6134)
HyperFast em CPU	0,8815 (IC95%: 0,8717 – 0,8910)	0,4496 (IC95%: 0,4196 – 0,4796)	0,7967 (IC95%: 0,7845 – 0,8088)	0,5931 (IC95%: 0,5868 – 0,5992)
HyperFast em CPU sem optimization	0,8368 (IC95%: 0,8254 – 0,8482)	0,3175 (IC95%: 0,2898 – 0,3468)	0,7487 (IC95%: 0,7351 – 0,7618)	0,5694 (IC95%: 0,5642 – 0,5745)
HyperFast em GPU	0,8814 (IC95%: 0,8716 – 0,8910)	0,4492 (IC95%: 0,4193 – 0,4791)	0,7956 (IC95%: 0,7838 – 0,8077)	0,5927 (IC95%: 0,5864 – 0,5988)
HyperFast em GPU sem optimization	0,8368 (IC95%: 0,8254 – 0,8484)	0,3176 (IC95%: 0,2899 – 0,3468)	0,7503 (IC95%: 0,7370 – 0,7631)	0,5696 (IC95%: 0,5645 – 0,5748)

Fonte: Elaboração própria

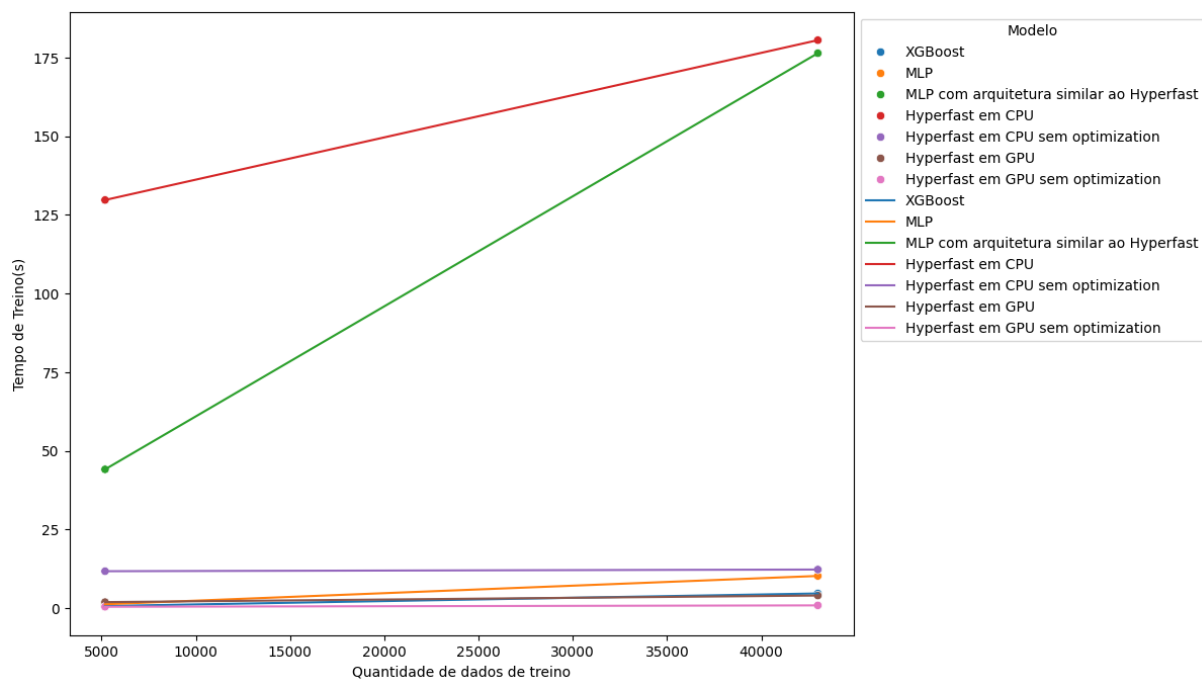
Com base na Tabela 7, vemos que o XGBoost obteve o melhor desempenho em todas as métricas avaliadas, destacando-se com uma AUC ROC de 0,9094 (IC95%: 0,9001 – 0,9176), acurácia balanceada de 0,8292 (IC95%: 0,8172 – 0,8410) e, crucialmente, uma AUPRC de 0,5134 (IC95%: 0,4811 – 0,5438). Os demais modelos baseados em redes neurais — MLP padrão, MLP com arquitetura similar ao HyperFast os HyperFast com optimization (CPU e GPU) — apresentaram desempenho intermediário e relativamente próximo entre si. As AUCs encontraram-se na faixa de 0,881 a 0,886, os valores de AUPRC situaram-se entre 0,439 e 0,449 e as acurácias balanceada em torno de 0,8. Esses resultados, embora expressivos, foram consistentemente inferiores aos alcançados pelo XGBoost. Os HyperFast sem otimização apresentaram desempenho consideravelmente inferior, com destaque para as reduções marcantes em AUPRC ($\approx 0,318$) e acurácia balanceada ($\approx 0,75$). Esses resultados indicam que o algoritmo depende fortemente de ajustes finos para convergir para uma representação significativa dos dados, e que a ausência dessas otimizações compromete severamente sua capacidade preditiva.

Olhando a Figura 7, observamos que a curva Precisão-Recall do XGBoost se mantém consistentemente acima das demais ao longo de praticamente todo o espectro de *recall*, evidenciando melhor capacidade de identificar verdadeiros positivos sem sacrificar excessivamente a precisão. As curvas dos MLPs e dos HyperFast com optimization ficaram bem parecidas, indicando que tem desempenho preditivos similares.

Outro ponto a ser observado é o valor das precisões que ficaram 0,6. Isso sugere

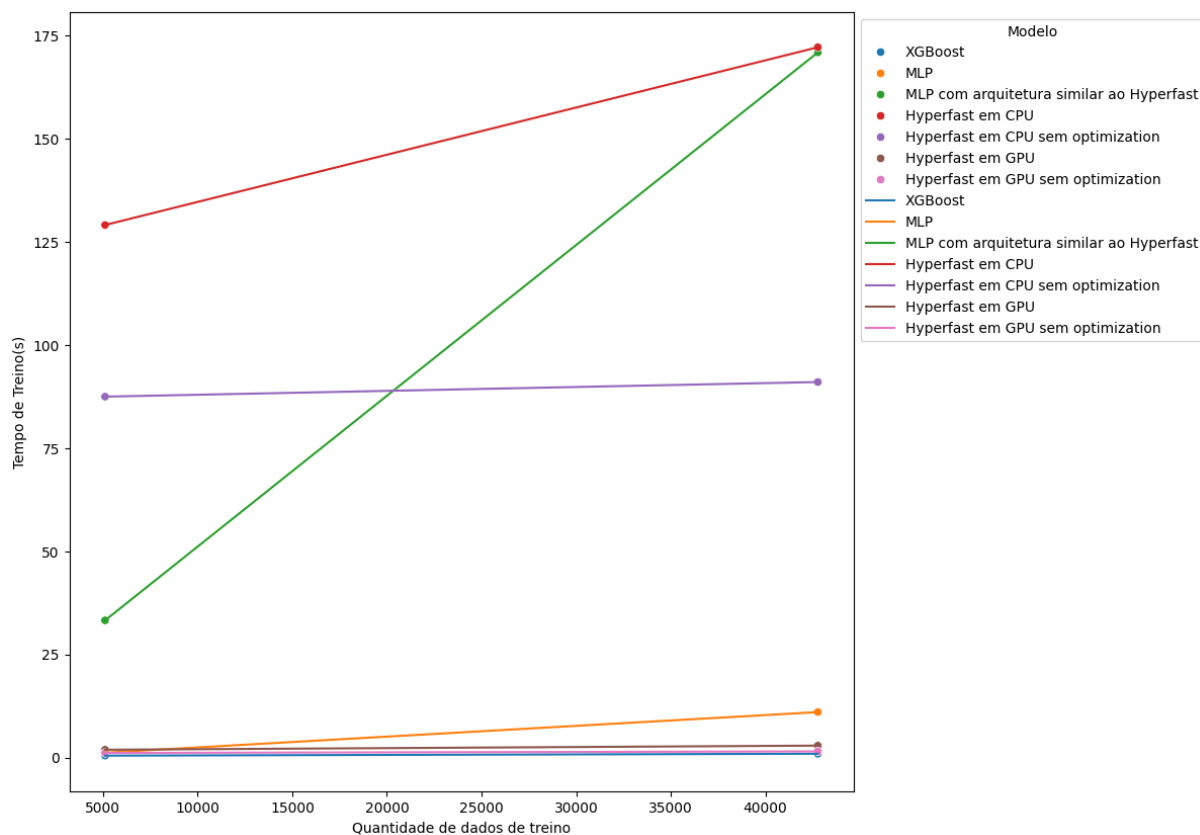
que os modelos, podem ser calibrados para melhorar aumentar a precisão sacrificando um pouco do recall, algo que pode ser observado no gráfico das curvas.

Figura 8 - Gráfico do número de dados vs tempo de Treinamento usando os conjuntos D1 e D3.



Fonte: Elaboração própria

Figura 9 - Gráfico do número de dados vs tempo de Treinamento usando os conjuntos D2 e D4.



Fonte: Elaboração própria

Em relação aos tempos para treino, como podemos ver pelos gráficos das figuras 8 e 9, o Modelo MPL com arquitetura similar ao HyperFast, cresceu mais em relação aos dados que o HyperFast sendo rodado em CPU. O HyperFast sem otimização não apresentou aumento de tempo. Quando olhamos para o hiperparâmetros do HyperFast com otimização, vemos que no conjunto D1 o parâmetro *optimize_steps* é 16 e 8 em D3. O HyperFast sem otimização fica com tempo de treinamento igual, independentemente da quantidade de dados. O mesmo se repete para D2 e D4, com a diferença que o HyperFast sem otimização usou *batch_size* de 2048 em D2 e D4 mas 1024 em D2 e D1.

De fato, ao pegar as tabelas geradas pela validação cruzada, e fazer o gráfico do tempo de treino nos modelos sem *optimize_steps* (Figura 10) e depois fixando o *batch_size* nos modelos de D1 e D3 com *fine tuning* (Figura 11), podemos observar a quantidade de dados não influencia no tempo de treinamento, somente os hiperparâmetros usados.

Outro ponto é que o tempo de predição foi maior que os outros modelos no HyperFast rodando em CPU. Observando a tabela 8, vemos que o HyperFast tem o maior tempo de predição, em torno de 12 a 13 segundos. Isso se deve ao fato de que o modelo sempre precisa fazer o processo de PCA e RF para deixar as entradas com a mesma dimensão da rede principal gerada.

Tabela 8 - Tempo de predição do HyperFast em CPU e Quantidade de dados

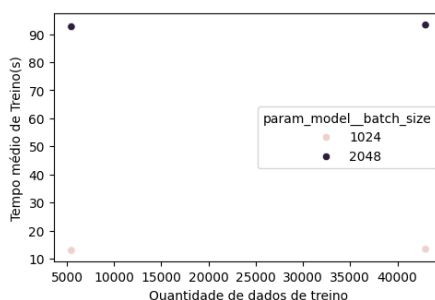
Modelo	Tempo de Predição(s)	Quantidade de dados de test
D1		
XGBoost	0,0449	18429
MLP	0,0321	18429
MLP com arquitetura similar ao HyperFast	0,3990	18429
HyperFast em CPU	12,8867	18429
HyperFast em CPU sem optimization	12,9607	18429
HyperFast em GPU	1,4701	18429
HyperFast em GPU sem optimization	2,3546	18429
D2		
XGBoost	0,0217	18327
MLP	0,0253	18327
MLP com arquitetura similar ao HyperFast	0,3201	18327
HyperFast em CPU	12,6081	18327
HyperFast em CPU sem optimization	12,6515	18327
HyperFast em GPU	1,4630	18327
HyperFast em GPU sem optimization	1,3649	18327
D3		
XGBoost	0,0223	18429
MLP	0,0244	18429
MLP com arquitetura similar ao HyperFast	0,3060	18429
HyperFast em CPU	12,5967	18429
HyperFast em CPU sem optimization	12,9028	18429
HyperFast em GPU	1,4146	18429

HyperFast em GPU sem optimization	2,3227	18429
D4		
XGBoost	0,0233	18327
MLP	0,0246	18327
MLP com arquitetura similar ao HyperFast	0,3321	18327
HyperFast em CPU	12,6556	18327
HyperFast em CPU sem optimization	12,5860	18327
HyperFast em GPU	1,4196	18327
HyperFast em GPU sem optimization	1,3441	18327

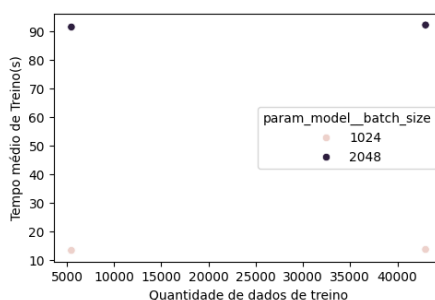
Fonte: Elaboração própria

Por fim, com base na Tabela 9, observamos que os modelos gerados pelo HyperFast possuem tamanho em memória muito maior que os modelos gerados pelos outros algoritmos. Especificamente em relação a MLP com a arquitetura similar do HyperFast que tem segundo maior tamanho (~28Mb), os modelos HyperFast tem tamanho em torno de 181 vezes maior(~5000Mb) e aproximadamente 10000 vezes maior que o XGBoost, que foi o menor em todos os conjntos.

Figura 10 - Gráficos dos tempos de treino em relação a quantidade de dados dos modelos HyperFast em CPU sem fine tuning



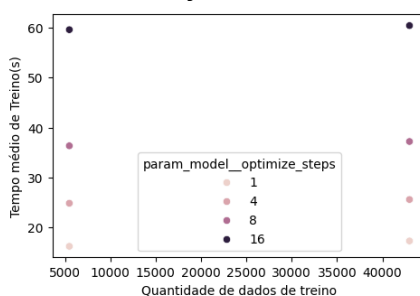
(a) - Tempo médio de treino por quantidade de dados por batch_size usando as quantidades dos conjunto D1 e D3



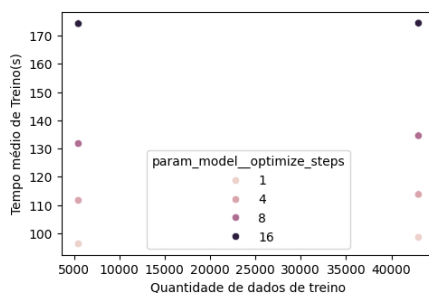
(b) - Tempo médio de treino por quantidade de dados por batch_size usando as quantidades dos conjunto D2 e D4

Fonte: Elaboração própria

Figura 11 - Gráficos dos tempos de treino em relação a quantidade de dados dos modelos HyperFast em CPU com fine tuning usando as quantidades de dados dos conjuntos D1 e D3



(a) - Tempo médio de treino por quantidade de dados por optimize_step fixando batch_size em 1024



(b) - Tempo médio de treino por quantidade de dados por optimize_step fixando batch_size em 2048

Fonte: Elaboração própria

Tabela 9 - Tamanho dos modelos em MegaBytes

Conjunto	Modelo	Tamanho do modelo(megabytes)
D1	XGBoost	0,565
D1	MLP	0,671
D1	MLP com arquitetura similar ao HyperFast	28,440
D1	HyperFast em CPU	5148,404
D1	HyperFast em CPU sem optimization	5019,670
D1	HyperFast em GPU	4985,703
D1	HyperFast em GPU sem optimization	4984,969
D2	XGBoost	0,234
D2	MLP	0,671
D2	MLP com arquitetura similar ao HyperFast	28,440
D2	HyperFast em CPU	5148,404
D2	HyperFast em CPU sem optimization	5148,404
D2	HyperFast em GPU	4985,703
D2	HyperFast em GPU sem optimization	4985,703
D3	XGBoost	0,229
D3	MLP	0,458
D3	MLP com arquitetura similar ao HyperFast	28,441
D3	HyperFast em CPU	5148,405
D3	HyperFast em CPU sem optimization	5019,670
D3	HyperFast em GPU	4985,703
D3	HyperFast em GPU sem optimization	4984,969
D4	XGBoost	0,232
D4	MLP	0,458

D4	MLP com arquitetura similar ao HyperFast	28,440
D4	HyperFast em CPU	5148,404
D4	HyperFast em CPU sem optimization	5148,404
D4	HyperFast em GPU	4985,703
D4	HyperFast em GPU sem optimization	4985,703

Fonte: Elaboração própria

5 Conclusões

Este trabalho fez modelos de predição de mortalidade em até sete dias após a primeira entrada na UTI usando dados coletados nas primeiras 24 horas após a entradas na UTI usando os modelos XGBoost, MLP e HyperFast e comparando-os quanto suas performances preditivas, tamanhos e tempos.

Levando em consideração os resultados vistos na seção anterior, conclui-se que o algoritmo que teve melhor desempenho foi XGBoost, sendo melhor não só no conjunto D4, mas também nos outros conjuntos. HyperFast se mostrou mais competitivo com balanceamento e eliminação de outliers sendo comparável aos modelos sendo comparável MLP. O HyperFast sem otimização só foi melhor em relação a acurácia balanceada nos conjuntos onde não foi feito balanceamento, o que pode ser explicado pelo processo de otimização (*fine tuning*) sendo enviesado pela maior proporção da classe positiva.

Pensando no cenário complexo de mortalidade em UTIs e hospitais em geral, que exigem decisões rápidas e que buscam ter um Recall alto, o modelo XGBoost mostrou-se o mais adequado pelo desempenho, tamanho e velocidade de treino e predição. Os resultados de tamanho e velocidade do HyperFast, que teve resultados de performance preditiva razoáveis e comparáveis as redes neurais no melhor cenário, são um fator limitante, por exigir alta memória e a necessidade de ter GPU para fazer predições rápidas, impossibilitando por exemplo a implementação dentro de um dispositivo móvel que possa ser usada pela equipe médica.

5.1 Limitações

Neste trabalho, o número de modelos e hiperparâmetros usados nos grids foi limitado pelo tempo disponível de uso da máquina virtual do Google Colab.

5.2 Trabalhos Futuros

Como trabalhos futuros, podemos avaliar o uso do HyperFast compará-lo com outros algoritmos que não foram usados aqui. Outro trabalho é fazer uma comparação com uma maior quantidade de conjuntos de dados e com quantidade de dados diferentes para ter uma avaliação melhor da influência da quantidade de dados em relação ao *batch size*, considerando a influência que terá na performance geral do modelo. Nesse mesmo contexto avaliar como os outros hiperparâmetros, por exemplo *n_ensemble*, influenciam na performance e no tempo de treinamento e predição.

Outro possível ponto de pesquisa, é avaliar os modelos ao longo do tempo, ou seja, como os modelos que são treinados com dados de um período específico de tempo vão performar com dados de outros períodos.

Por fim, gerar os parâmetros das hiper redes usando conjuntos de dados deste contexto, e avaliar as redes geradas por esse HyperFast, também é uma ideia interessante para um próximo trabalho, porém a atual implementação não permite fazer esse treinamento da hiper rede.

6 Referências

VRANAS, K. C. et al. Identifying Distinct Subgroups of ICU Patients. **Critical Care Medicine**, v. 45, n. 10, p. 1607–1615, out. 2017. DOI: <https://doi.org/10.1097/CCM.0000000000002548>.

CASTELA FORTE, J. et al. Identifying and characterizing high-risk clusters in a heterogeneous ICU population with deep embedded clustering. **Scientific Reports**, v. 11, n. 1, 8 jun. 2021. DOI: <https://doi.org/10.1038/s41598-021-91297-x>.

SAADATMAND, S. et al. Using machine learning in prediction of ICU admission, mortality, and length of stay in the early stage of admission of COVID-19 patients. **Annals of Operations Research**, 29 set. 2022. DOI : <https://doi.org/10.1007/s10479-022-04984-x>.

FENN, A. et al. Development and Validation of Machine Learning Models to Predict Admission From Emergency Department to Inpatient and Intensive Care Units. **Annals of Emergency Medicine**, v. 78, n. 2, p. 290–302, ago. 2021. DOI: <https://doi.org/10.1016/j.annemergmed.2021.02.029>.

HONG, W. S.; HAIMOVICH, A. D.; TAYLOR, R. A. Predicting hospital admission at emergency department triage using machine learning. **PLOS ONE**, v. 13, n. 7, p. e0201016, 20 jul. 2018. DOI: <https://doi.org/10.1371/journal.pone.0201016>.

SUBUDHI, S. et al. Comparing machine learning algorithms for predicting ICU admission and mortality in COVID-19. **npj Digital Medicine**, v. 4, n. 1, 21 maio 2021. DOI: <https://doi.org/10.1038/s41746-021-00456-x>.

WU, J. et al. Predicting Prolonged Length of ICU Stay through Machine Learning. **Diagnostics**, v. 11, n. 12, p. 2242, 1 dez. 2021. DOI: <https://doi.org/10.3390/diagnostics11122242>.

MCWILLIAMS, C. J. et al. Towards a decision support tool for intensive care discharge: machine learning algorithm development using electronic healthcare data from MIMIC-III and Bristol, UK. **BMJ Open**, v. 9, n. 3, p. e025925, 1 mar. 2019. DOI: <https://doi.org/10.1136/bmjopen-2018-025925>.

ZHAI, Q. et al. Using machine learning tools to predict outcomes for emergency department intensive care unit patients. **Scientific Reports**, v. 10, n. 1, dez. 2020. DOI: <https://doi.org/10.1038/s41598-020-77548-3>.

RAITA, Y. et al. Emergency department triage prediction of clinical outcomes using machine learning models. **Critical Care**, v. 23, n. 1, 22 fev. 2019. DOI: <https://doi.org/10.1186/s13054-019-2351-7>

HAYKIN, Simon S. *Neural Networks: A Comprehensive Foundation*. 2. ed. Upper Saddle River: Prentice Hall, 1999.

BURKOV, Andriy. *The Hundred-Page Machine Learning Book*. 1. ed. Québec City: Andriy Burkov, 2019.

CHEN, T.; GUESTRIN, C. XGBoost: a Scalable Tree Boosting System. **KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 785–794, 2016. DOI: <https://doi.org/10.1145/2939672.2939785>.

XGBOOST DEVELOPERS. **XGBoost Documentation — xgboost 1.5.1 documentation**. Disponível em: <<https://xgboost.readthedocs.io/en/stable/>>.

BONET, D. et al. HyperFast: Instant Classification for Tabular Data. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 38, n. 10, p. 11114–11123, 24 mar. 2024. DOI: <https://doi.org/10.48550/arXiv.2402.14335>.

KNAUER, Ricardo; RODNER, Erik. Squeezing Lemons with Hammers: An Evaluation of AutoML and Tabular Deep Learning for Data-Scarce Classification Applications. **arXiv preprint arXiv:2405.07662**, 2024. DOI: <https://doi.org/10.48550/arXiv.2405.07662>

LEE, Chongmin; KIM, Jihie. Biomarker based Cancer Classification using an Ensemble with Pre-trained Models. **arXiv preprint arXiv:2406.10087**, 2024. DOI: <https://doi.org/10.48550/arXiv.2406.10087>

HA, David; DAI, Andrew; LE, Quoc V. Hypernetworks. **arXiv preprint arXiv:1609.09106**, 2016. DOI: <https://doi.org/10.48550/arXiv.1609.09106>

CABALLAR, R.; STRYKER, C. **Meta learning**. Disponível em: <<https://www.ibm.com/think/topics/meta-learning>>.

AI-SANDBOX. **GitHub - AI-sandbox/HyperFast**. Disponível em: <<https://github.com/AI-sandbox/HyperFast>>. Acesso em: 4 dez. 2024.

BREIMAN, L. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001. DOI: <https://doi.org/10.1023/A:1010933404324>

SCIKIT-LEARN. **scikit-learn: machine learning in Python**. Disponível em: <<https://scikit-learn.org/stable/>>.

TAYLOR, R. A. et al. Prediction of In-hospital Mortality in Emergency Department Patients With Sepsis: A Local Big Data-Driven, Machine Learning Approach. **Academic Emergency Medicine**, v. 23, n. 3, p. 269–278, 13 fev. 2016.

SCHMIDT, Diogo; DA SILVA, Denise Bandeira; DA COSTA, Cristiano André; RIGHI, Rodrigo da Rosa. Um Modelo de Predição de Mortalidade em Unidades de Terapia Intensiva Baseado em Deep Learning. *In*: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO APLICADA À SAÚDE (SBCAS), 18., 2018, Natal. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2018. ISSN 2763-8952. DOI: <https://doi.org/10.5753/sbcas.2018.3677>.

CHOI, M. H. et al. Mortality prediction of patients in intensive care units using machine learning algorithms based on electronic health records. **Scientific Reports**, v. 12, n. 1, 3 maio 2022. DOI: <https://doi.org/10.1038/s41598-022-11226-4>.

KEEGAN, Mark T.; SOARES, Marcio. What every intensivist should know about prognostic scoring systems and risk-adjusted mortality. *Revista Brasileira de Terapia Intensiva*, v. 28, n. 3, p. 264-269, Jan. 2016.

VEITH, N.; STEELE, R. Machine Learning-based Prediction of ICU Patient Mortality at Time of Admission. **Proceedings of the 2nd International Conference on Information System and Data Mining - ICISDM '18**, 2018.

NEMATI, S. et al. An Interpretable Machine Learning Model for Accurate Prediction of Sepsis in the ICU. **Critical Care Medicine**, v. 46, n. 4, p. 547–553, 1 abr. 2018.

CHIA, A. H. T. et al. Explainable machine learning prediction of ICU mortality. **Informatics in Medicine Unlocked**, v. 25, p. 100674, 2021. DOI: <https://doi.org/10.1016/j.imu.2021.100674>.

MARTIN-LOECHES, Ignacio *et al.* Determinants of time to death in hospital in critically ill patients around the world. **Intensive Care Medicine**, v. 42, n. 9, p. 1454-1460, set. 2016. DOI: 10.1007/s00134-016-4479-0. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/27518322/>.

CHAPMAN, Pete et al. CRISP-DM 1.0. **CRISP-DM Consortium**, v. 76, n. 3, 2000.

Classificação: precisão, recall, precisão e métricas relacionadas. Disponível em: <<https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=pt-br>>. Acesso em: 22 nov. 2024.

JOHNSON, A. E. W. et al. MIMIC-IV, a freely accessible electronic health record dataset. **Scientific Data**, v. 10, n. 1, 3 jan. 2023.

JOHNSON, A. et al. **MIMIC-IV**. 2024. Disponível em: <<https://physionet.org/content/mimiciv/3.1/>>. Acesso em: 25 nov. 2024.

SAITO, T.; REHMSMEIER, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. **PLOS ONE**, v. 10, n. 3, p. e0118432, 4 mar. 2015. DOI: <https://doi.org/10.1371>

BERTRAND BOUVAREL; FABRICE CARRAT; LAPIDUS, N. Updating mortality risk estimation in intensive care units from high-dimensional electronic health records with incomplete data. **BMC Medical Informatics and Decision Making**, v. 23, n. 1, 30 ago. 2023.

7 Apêndices

A - Consulta para extrair os exames laboratoriais:

```

with entradas as (select ic.subject_id, ic.hadm_id, ic.stay_id, first_careunit as
care_unit, ic.intime as icu_intime, ic.outtime as icu_outtime, ic.los as icu_los,

ad.admittime as hosp_intime, ad.disctime as hosp_outtime, ad.deathtime,
ad.admission_type, ad.hospital_expire_flag

from physionet-data.mimiciv_2_2_icu.icustays as ic inner join physionet-
data.mimiciv_2_2_hosp.admissions as ad on ic.hadm_id = ad.hadm_id

where ic.intime in (

select min(ic.intime)

from physionet-data.mimiciv_2_2_icu.icustays as ic

group by ic.hadm_id)),

labs as (select labev.hadm_id, labev.itemid, labev.charttime, labev.storetime,
labev.valuenum, labev.valueuom

from physionet-data.mimiciv_2_2_hosp.labevents as labev

where labev.itemid in(

-- LABS

51221, -- Hematocrit | Blood | Hematology | % | 325700

51222, -- Hemoglobin | Blood | Hematology | g/dL | 3188835

51301, -- White Blood Cells | Blood | Hematology | K/uL | 3171798

50862, -- Albumin | Blood | Chemistry | g/dL | 749944

51492, -- Protein | Urine | Hematology | mg/dL | 682918

50889, -- C-Reactive Protein | Blood | Chemistry | mg/L | 119711

50868, -- Anion Gap | Blood | Chemistry | mEq/L | 2965452

50882, -- Bicarbonate | Blood | Chemistry | mEq/L | 2972827

50893, -- Calcium, Total | Blood | Chemistry | mg/dL | 2190816

50808, -- Free Calcium | Blood | Blood Gas | mmol/L | 281355

50912, -- Creatinine | Blood | Chemistry | mg/dL | 3282278

50902, -- Chloride | Blood | Chemistry | mEq/L | 3083705

50931, -- Glucose | Blood | Chemistry | mg/dL | 2749562

```

```

50971, -- Potassium | Blood | Chemistry | mEq/L | 3150261
50983, -- Sodium | Blood | Chemistry | mEq/L | 3119778
51006, -- Urea Nitrogen | Blood | Chemistry | mg/dL | 3189474
50960, -- Magnesium | Blood | Chemistry | mg/dL | 2154209
50970, -- Phosphate | Blood | Chemistry | mg/dL | 2066052
50813, -- Lactate | Blood | Blood Gas | mmol/L | 506778
50821, -- pO2 | Blood | Blood Gas | mm Hg | 511938
50818, -- pCO2 | Blood | Blood Gas | mm Hg | 511713
50885, -- Bilirubin, Total | Blood | Chemistry | mg/dL |
1174903
51464, -- Bilirubin | Urine | Hematology | mg/dL | 532715
51265 -- Platelet Count | Blood | Hematology | K/uL | 3216656
)
),
patients as( select subject_id, gender, anchor_age
from physionet-data.mimiciv_2_2_hosp.patients
where anchor_age >= 18)
select *
from (entradas inner join patients on entradas.subject_id = patients.subject_id)
inner join labs on entradas.hadm_id = labs.hadm_id
where labs.charttime >= entradas.icu_intime and labs.charttime <=
DATETIME_ADD(entradas.icu_intime, INTERVAL '1' DAY)

```

B - Consulta para extrair as medições de sinais vitais agregadas de mínimo, máximo, média e desvio padrão

```

WITH first_stay AS (
SELECT
    ic.subject_id,
    ic.hadm_id,
    ic.stay_id,
    ic.first_careunit AS care_unit,
    ic.intime AS icu_intime,
    ic.outtime AS icu_outtime,

```

```

        ic.los AS icu_los,
        ad.admittime AS hosp_intime,
        ad.dischtime AS hosp_outtime,
        ad.deathtime,
        ad.admission_type,
        ad.hospital_expire_flag,
        ROW_NUMBER() OVER (PARTITION BY ic.hadm_id ORDER BY ic.intime) AS rn
    FROM `physionet-data.mimiciv_2_2_icu.icustays` ic
    INNER JOIN `physionet-data.mimiciv_2_2_hosp.admissions` ad
        ON ic.hadm_id = ad.hadm_id
),
entradas AS (
    SELECT *
    FROM first_stay
    WHERE rn = 1
),

first_vent AS (
    SELECT
        ce.stay_id,
        MIN(ce.charttime) AS first_vent_time,
        MIN(CASE WHEN ce.value IN ('Tracheostomy tube', 'Endotracheal tube') THEN
ce.charttime END) AS first_invasive_vent_time
    FROM `physionet-data.mimiciv_2_2_icu.chartevents` ce
    JOIN entradas e ON ce.stay_id = e.stay_id
    WHERE ce.itemid = 226732
        AND ce.charttime >= e.icu_intime
        AND ce.charttime <= DATETIME_ADD(e.icu_intime, INTERVAL 1 DAY)
        AND (e.deathtime IS NULL OR ce.charttime < DATETIME_SUB(e.deathtime, INTERVAL 6
HOUR))
    GROUP BY ce.stay_id
),

chart_events as (select c_events.stay_id,

MIN(CASE WHEN itemid = 220045 and c_events.valuenum >= 25 and c_events.valuenum <=
700 THEN c_events.valuenum ELSE NULL END) as HeartRate_min,
MAX(CASE WHEN itemid = 220045 and c_events.valuenum >= 25 and c_events.valuenum <=
700 THEN c_events.valuenum ELSE NULL END) as HeartRate_max,
STDDEV(CASE WHEN itemid = 220045 and c_events.valuenum >= 25 and c_events.valuenum
<= 700 THEN c_events.valuenum ELSE NULL END) as HeartRate_std,

```

```

AVG(CASE WHEN itemid = 220045 and c_events.valuenum >= 25 and c_events.valuenum <=
700 THEN c_events.valuenum ELSE NULL END) as HeartRate_mean,

MIN(CASE WHEN itemid = 220210 and c_events.valuenum >= 1 and c_events.valuenum <= 70
THEN c_events.valuenum ELSE NULL END) as RespiratoryRate_min,
MAX(CASE WHEN itemid = 220210 and c_events.valuenum >= 1 and c_events.valuenum <= 70
THEN c_events.valuenum ELSE NULL END) as RespiratoryRate_max,
STDDEV(CASE WHEN itemid = 220210 and c_events.valuenum >= 1 and c_events.valuenum <=
70 THEN c_events.valuenum ELSE NULL END) as RespiratoryRate_std,
AVG(CASE WHEN itemid = 220210 and c_events.valuenum >= 1 and c_events.valuenum <= 70
THEN c_events.valuenum ELSE NULL END) as RespiratoryRate_mean,

MIN(CASE WHEN itemid = 220050 and c_events.valuenum >= 30 and c_events.valuenum <=
400 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureSystolic_min,
MAX(CASE WHEN itemid = 220050 and c_events.valuenum >= 30 and c_events.valuenum <=
400 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureSystolic_max,
STDDEV(CASE WHEN itemid = 220050 and c_events.valuenum >= 30 and c_events.valuenum
<= 400 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureSystolic_std,
AVG(CASE WHEN itemid = 220050 and c_events.valuenum >= 30 and c_events.valuenum <=
400 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureSystolic_mean,

MIN(CASE WHEN itemid = 220179 and c_events.valuenum >= 30 and c_events.valuenum <=
400 THEN c_events.valuenum ELSE NULL END) as NonInvasiveBloodPressureSystolic_min,
MAX(CASE WHEN itemid = 220179 and c_events.valuenum >= 30 and c_events.valuenum <=
400 THEN c_events.valuenum ELSE NULL END) as NonInvasiveBloodPressureSystolic_max,
STDDEV(CASE WHEN itemid = 220179 and c_events.valuenum >= 30 and c_events.valuenum
<= 400 THEN c_events.valuenum ELSE NULL END) as
NonInvasiveBloodPressureSystolic_std,
AVG(CASE WHEN itemid = 220179 and c_events.valuenum >= 30 and c_events.valuenum <=
400 THEN c_events.valuenum ELSE NULL END) as NonInvasiveBloodPressureSystolic_mean,

MIN(CASE WHEN itemid = 220051 and c_events.valuenum >= 10 and c_events.valuenum <=
380 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureDiastolic_min,
MAX(CASE WHEN itemid = 220051 and c_events.valuenum >= 10 and c_events.valuenum <=
380 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureDiastolic_max,
STDDEV(CASE WHEN itemid = 220051 and c_events.valuenum >= 10 and c_events.valuenum
<= 380 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureDiastolic_std,
AVG(CASE WHEN itemid = 220051 and c_events.valuenum >= 10 and c_events.valuenum <=
380 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureDiastolic_mean,

MIN(CASE WHEN itemid = 220180 and c_events.valuenum >= 10 and c_events.valuenum <=
380 THEN c_events.valuenum ELSE NULL END) as NonInvasiveBloodPressureDiastolic_min,
MAX(CASE WHEN itemid = 220180 and c_events.valuenum >= 10 and c_events.valuenum <=
380 THEN c_events.valuenum ELSE NULL END) as NonInvasiveBloodPressureDiastolic_max,

```

```

STDDEV(CASE WHEN itemid = 220180 and c_events.valuenum >= 10 and c_events.valuenum
<= 380 THEN c_events.valuenum ELSE NULL END) as
NonInvasiveBloodPressureDiastolic_std,
AVG(CASE WHEN itemid = 220180 and c_events.valuenum >= 10 and c_events.valuenum <=
380 THEN c_events.valuenum ELSE NULL END) as
NonInvasiveBloodPressureDiastolic_mean,

MIN(CASE WHEN itemid = 220052 and c_events.valuenum >= 20 and c_events.valuenum <=
390 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureMean_min,
MAX(CASE WHEN itemid = 220052 and c_events.valuenum >= 20 and c_events.valuenum <=
390 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureMean_max,
STDDEV(CASE WHEN itemid = 220052 and c_events.valuenum >= 20 and c_events.valuenum
<= 390 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureMean_std,
AVG(CASE WHEN itemid = 220052 and c_events.valuenum >= 20 and c_events.valuenum <=
390 THEN c_events.valuenum ELSE NULL END) as ArterialBloodPressureMean_mean,

MIN(CASE WHEN itemid = 220181 and c_events.valuenum >= 20 and c_events.valuenum <=
390 THEN c_events.valuenum ELSE NULL END) as NonInvasiveBloodPressureMean_min,
MAX(CASE WHEN itemid = 220181 and c_events.valuenum >= 20 and c_events.valuenum <=
390 THEN c_events.valuenum ELSE NULL END) as NonInvasiveBloodPressureMean_max,
STDDEV(CASE WHEN itemid = 220181 and c_events.valuenum >= 20 and c_events.valuenum
<= 390 THEN c_events.valuenum ELSE NULL END) as NonInvasiveBloodPressureMean_std,
AVG(CASE WHEN itemid = 220181 and c_events.valuenum >= 20 and c_events.valuenum <=
390 THEN c_events.valuenum ELSE NULL END) as NonInvasiveBloodPressureMean_mean,

MIN(CASE WHEN itemid = 220277 and c_events.valuenum >= 40 and c_events.valuenum <=
100 THEN c_events.valuenum ELSE NULL END) as O2SaturationPulseoxymetry_min,
MAX(CASE WHEN itemid = 220277 and c_events.valuenum >= 40 and c_events.valuenum <=
100 THEN c_events.valuenum ELSE NULL END) as O2SaturationPulseoxymetry_max,
STDDEV(CASE WHEN itemid = 220277 and c_events.valuenum >= 40 and c_events.valuenum
<= 100 THEN c_events.valuenum ELSE NULL END) as O2SaturationPulseoxymetry_std,
AVG(CASE WHEN itemid = 220277 and c_events.valuenum >= 40 and c_events.valuenum <=
100 THEN c_events.valuenum ELSE NULL END) as O2SaturationPulseoxymetry_mean,

MIN(CASE WHEN itemid = 223761 and c_events.valuenum >= 50 and c_events.valuenum <=
116 THEN c_events.valuenum ELSE NULL END) as TemperatureFahrenheit_min,
MAX(CASE WHEN itemid = 223761 and c_events.valuenum >= 50 and c_events.valuenum <=
116 THEN c_events.valuenum ELSE NULL END) as TemperatureFahrenheit_max,
STDDEV(CASE WHEN itemid = 223761 and c_events.valuenum >= 50 and c_events.valuenum
<= 116 THEN c_events.valuenum ELSE NULL END) as TemperatureFahrenheit_std,
AVG(CASE WHEN itemid = 223761 and c_events.valuenum >= 50 and c_events.valuenum <=
116 THEN c_events.valuenum ELSE NULL END) as TemperatureFahrenheit_mean,

```

```

MIN(CASE WHEN itemid = 223762 and c_events.valuenum >= 10 and c_events.valuenum <=
46.7 THEN c_events.valuenum ELSE NULL END) as TemperatureCelsius_min,
MAX(CASE WHEN itemid = 223762 and c_events.valuenum >= 10 and c_events.valuenum <=
46.7 THEN c_events.valuenum ELSE NULL END) as TemperatureCelsius_max,
STDDEV(CASE WHEN itemid = 223762 and c_events.valuenum >= 10 and c_events.valuenum
<= 46.7 THEN c_events.valuenum ELSE NULL END) as TemperatureCelsius_std,
AVG(CASE WHEN itemid = 223762 and c_events.valuenum >= 10 and c_events.valuenum <=
46.7 THEN c_events.valuenum ELSE NULL END) as TemperatureCelsius_mean,

MAX(CASE WHEN itemid = 226512 and c_events.valuenum >= 2 and c_events.valuenum <=
600 THEN c_events.valuenum ELSE NULL END) as weight_adm,

MIN(CASE WHEN itemid = 220739 and c_events.valuenum > 0 and c_events.valuenum <= 5
THEN c_events.valuenum END) as gcs_eye_min,
MIN(CASE WHEN itemid = 223900 and c_events.valuenum > 0 and c_events.valuenum <= 6
THEN c_events.valuenum END) as gcs_verbal_min,
MIN(CASE WHEN itemid = 223901 and c_events.valuenum > 0 and c_events.valuenum <= 7
THEN c_events.valuenum END) as gcs_motor_min

from physionet-data.mimiciv_2_2_icu.chartevents as c_events inner join entradas as
icu on c_events.stay_id = icu.stay_id
where c_events.itemid in(
    -- Chart Events
    220045, -- Heart Rate | Routine Vital Signs | bpm | bpm | 6460839
    220210, -- Respiratory Rate | Respiratory | insp/min | insp/min | 6393762
    220050, -- Arterial Blood Pressure systolic | Routine Vital Signs | mmHg |
mmHg | 2264001
    220179, -- Non Invasive Blood Pressure systolic | Routine Vital Signs |
mmHg | mmHg | 4067219
    220051, -- Arterial Blood Pressure diastolic | Routine Vital Signs | mmHg
| mmHg | 2263647
    220180, -- Non Invasive Blood Pressure diastolic | Routine Vital Signs |
mmHg | mmHg | 4066337
    220052, -- Arterial Blood Pressure mean | Routine Vital Signs | mmHg |
mmHg | 2272013
    220181, -- Non Invasive Blood Pressure mean | Routine Vital Signs | mmHg |
mmHg | 4064537
    220277, -- O2 saturation pulseoxymetry | Respiratory | % | % | 6324341
    223761, -- Temperature Fahrenheit | Routine Vital Signs | °F | °F |
1515962
    223762, -- Temperature Celsius | Routine Vital Signs | °C | °C | 264628
    226512, -- Admission Weight (Kg) | General | kg | kg | 67826

```

```

220739, -- GCS - Eye Opening | Neurological | val | val | 1630825
223900, -- GCS - Verbal Response | Neurological | val | val | 1627786
223901 -- GCS - Motor Response | Neurological | val | val | 1623795
    )
    and c_events.charttime >= icu.icu_intime and c_events.charttime <=
DATETIME_ADD(icu.icu_intime, INTERVAL '1' DAY)
    and (icu.deathtime IS NULL OR c_events.charttime < DATETIME_SUB(icu.deathtime,
INTERVAL 6 HOUR))
group by c_events.stay_id
),

```

```

patients AS (
    SELECT subject_id, gender, anchor_age, anchor_year, dod
    FROM `physionet-data.mimiciv_2_2_hosp.patients`
    WHERE anchor_age >= 18
)

```

```

SELECT
    e.*,
    p.gender, p.anchor_age, p.anchor_year, p.dod,
    c.*,
    (c.gcs_eye_min + c.gcs_verbal_min + c.gcs_motor_min) AS GCS_total_min,
    CASE
        WHEN fv.first_vent_time <= DATETIME_ADD(e.icu_intime, INTERVAL 24 HOUR) THEN 1
        ELSE 0
    END AS any_vent_first_24h,
    CASE
        WHEN fv.first_invasive_vent_time <= DATETIME_ADD(e.icu_intime, INTERVAL 24 HOUR)
    THEN 1
        ELSE 0
    END AS invasive_vent_first_24h
FROM entradas e
JOIN patients p ON e.subject_id = p.subject_id
JOIN chart_events c ON e.stay_id = c.stay_id
LEFT JOIN first_vent fv ON e.stay_id = fv.stay_id
ORDER BY e.stay_id;

```

D - Tabela das estatísticas descritivas das variáveis numéricas

Tabela A2 - Variáveis numéricas usadas e suas estatísticas descritivas

Variáveis numéricas					
Variável	Contagem de valores preenchidos	Mínimo	Máximo	Média	Desvio Padrão
HeartRate_min	65748.0	0.0	178.0	70.56545	15.239611
HeartRate_max	65748.0	32.0	10109.0	102.929777	61.786319
HeartRate_std	65748.0	0.0	2501.549613	8.679328	15.087652
HeartRate_mean	65748.0	28.5	816.5	84.545453	16.232172
RespiratoryRate_min	65652.0	0.0	38.0	11.995659	4.296855
RespiratoryRate_max	65652.0	8.0	2355560.0	63.867011	9193.262401
RespiratoryRate_std	65652.0	0.0	461959.463723	10.971315	1802.950817
RespiratoryRate_mean	65652.0	1.689655	90617.384615	20.384962	353.618231
O2SaturationPulseoxymetry_min	65708.0	-272.0	100.0	91.850753	6.582969
O2SaturationPulseoxymetry_max	65708.0	0.0	9765430.0	285.701391	38483.214577
O2SaturationPulseoxymetry_std	65708.0	0.0	1953066.800001	39.252159	7696.635477
O2SaturationPulseoxymetry_mean	65708.0	0.0	390709.36	104.308946	1539.324599
gcs_eye_min	65519.0	1.0	4.0	2.833407	1.259505
gcs_verbal_min	65498.0	1.0	5.0	3.246985	1.874074
gcs_motor_min	65491.0	1.0	6.0	4.66884	2.016544
GCS_total_min	65464.0	3.0	15.0	10.748534	4.809145
Hematocrit_m	63744.0	4.3	65.6	31.034951	6.475566

in					
Hematocrit_max	63744.0	12.5	68.6	33.615829	5.745359
Hematocrit_mean	63744.0	12.15	65.6	32.351053	5.850334
Hemoglobin_min	63541.0	1.3	19.8	10.288972	2.203361
Hemoglobin_max	63541.0	3.5	23.3	11.052774	2.018454
Hemoglobin_mean	63541.0	3.5	20.2	10.677353	2.034264
WhiteBloodCells_min	63549.0	0.0	471.7	10.6678	8.144225
WhiteBloodCells_max	63549.0	0.1	572.5	12.92995	10.769607
WhiteBloodCells_mean	63549.0	0.075	474.7	11.776346	9.170332
AnionGap_min	63883.0	-9.0	49.0	13.273203	3.577943
AnionGap_max	63883.0	-3.0	89.0	15.291862	4.430695
AnionGap_mean	63883.0	-3.0	52.428571	14.25572	3.734734
Bicarbonate_min	63950.0	2.0	50.0	22.47898	4.726282
Bicarbonate_max	63950.0	4.0	50.0	24.225934	4.252156
Bicarbonate_mean	63950.0	4.0	50.0	23.367429	4.319507
CalciumTotal_min	59075.0	0.0	29.5	8.21149	0.831303
CalciumTotal_max	59075.0	2.9	29.5	8.529493	0.811667
CalciumTotal_mean	59075.0	2.9	29.5	8.370927	0.749251
Creatinine_min	63966.0	0.0	31.8	1.320491	1.431646
Creatinine_max	63966.0	0.0	80.0	1.496132	1.716034
Creatinine_m	63966.0	0.0	31.95	1.407894	1.54762

ean					
Chloride_min	63992.0	57.0	150.0	102.888334	6.149171
Chloride_max	63992.0	67.0	155.0	105.233717	6.279644
Chloride_mean	63992.0	67.0	152.25	104.069875	5.99521
Glucose_min	63713.0	3.0	1407.0	122.450175	44.302873
Glucose_max	63713.0	25.0	2440.0	154.424529	86.09272
Glucose_mean	63713.0	25.0	1407.0	137.365853	54.605677
Potassium_min	63967.0	0.8	9.5	3.976002	0.567866
Potassium_max	63967.0	1.6	26.5	4.400448	0.740899
Potassium_mean	63967.0	1.6	9.5	4.180682	0.569786
Sodium_min	63990.0	80.0	175.0	137.412736	4.947486
Sodium_max	63990.0	107.0	185.0	139.395593	4.784568
Sodium_mean	63990.0	104.222222	179.0	138.417602	4.645373
UreaNitrogen_min	63947.0	1.0	263.0	23.688476	19.921653
UreaNitrogen_max	63947.0	1.0	263.0	26.558244	22.205475
UreaNitrogen_mean	63947.0	1.0	263.0	25.125665	20.898531
Magnesium_min	61602.0	0.0	9.2	1.923212	0.381744
Magnesium_max	61602.0	0.7	86.0	2.167828	0.791013
Magnesium_mean	61602.0	0.7	22.85	2.040804	0.407363
Phosphate_min	59220.0	0.0	15.7	3.375461	1.190543
Phosphate_max	59220.0	0.4	58.0	3.889932	1.448582
Phosphate_mean	59220.0	0.4	21.533333	3.626555	1.229314
Lactate_min	35048.0	0.0	23.6	1.727386	1.237562

Lactate_max	35048.0	0.0	1276103.0	39.114098	6816.370926
Lactate_mean	35048.0	0.0	425368.4	14.327924	2272.121154
pO2_min	34708.0	0.0	587.0	88.94543	57.690063
pO2_max	34708.0	14.0	4242.0	194.586464	138.83404
pO2_mean	34708.0	14.0	2155.5	133.845615	81.37517
pCO2_min	34704.0	7.0	201.0	37.837223	9.843678
pCO2_max	34704.0	12.0	246.0	46.819185	12.960452
pCO2_mean	34704.0	10.4	228.5	41.960134	9.981017
PlateletCount_min	63565.0	5.0	2001.0	195.846079	103.014971
PlateletCount_max	63565.0	5.0	2360.0	215.967781	106.799184
PlateletCount_mean	63565.0	5.0	2180.5	205.726669	103.683465
SystolicBlood Pressure_min	65538.0	-94.0	247.0	96.083664	17.915845
SystolicBlood Pressure_max	65539.0	0.0	1025100.0	161.53736	4083.858898
SystolicBlood Pressure_std	65538.0	0.0	151126.029585	16.420817	607.159853
SystolicBlood Pressure_mean	65538.0	0.0	22395.717391	118.714374	94.522332
DiastolicBlood Pressure_min	65535.0	-16.0	138.0	49.251728	12.588428
DiastolicBlood Pressure_max	65535.0	0.0	114109.0	115.802838	1383.861072
DiastolicBlood Pressure_std	65535.0	0.0	27776.572659	16.246536	306.497871
DiastolicBlood Pressure_mean	65535.0	0.0	11410.333333	65.898271	77.746649
MeanBloodPressure_min	65529.0	-22767.0	215.0	61.713102	90.178509
MeanBloodPressure_max	65529.0	0.0	1234570.0	133.905065	4971.109842

MeanBloodPressure_std	65529.0	0.0	319348.988078	18.183899	1273.823659
MeanBloodPressure_mean	65529.0	0.0	80396.533333	79.958324	318.942408
Temperature_min	63947.0	0.0	208.58	97.258615	4.185411
Temperature_max	63947.0	0.0	9834.0	99.629756	42.679195
Temperature_std	63947.0	0.0	4353.959781	2.523834	19.895414
Temperature_mean	63947.0	0.0	2045.4	98.287493	8.479582
age	65756.0	18.0	102.0	64.571218	17.005852

Fonte: Elaboração própria

E - Tabela das estatísticas descritivas das variáveis categóricas

Tabela A3 - Variáveis categóricas usadas e suas estatísticas descritivas

Variáveis categóricas				
Variável	Contagem de valores preenchidos	Valores único	Valor de maior frequência e sua frequência	
gender	66158	2	M	36785
any_vent_first_24h	65756	2	1	64190
invasive_vent_first_24h	65756	2	0	42267
target	65756.	2	0	
admission_type	66158	9	EW EMER.	35019

Fonte: Elaboração própria

F - Tabela das estatísticas descritivas das variáveis numéricas do conjunto D1 antes da imputação de valores

Tabela A4 - Variáveis numéricas e suas estatísticas descritivas do conjunto D1 antes da imputação de valores

Variável	Contagem	Mínimo	Máximo	Média	Desvio Padrão
HeartRate_min	61426.0	0.000000	1.670000e+02	70.388598	15.100094
HeartRate_max	61426.0	32.000000	1.010900e+04	102.793866	63.648621
HeartRate_std	61426.0	0.000000	2.501550e+03	8.674370	15.553107
HeartRate_mean	61426.0	28.500000	8.165000e+02	84.378302	16.143564
RespiratoryRate_min	61346.0	0.000000	3.800000e+01	11.944935	4.275652
RespiratoryRate_max	61346.0	8.000000	2.355560e+06	66.372999	9510.437691
RespiratoryRate_std	61346.0	0.000000	4.619595e+05	11.463815	1865.154114
RespiratoryRate_mean	61346.0	1.689655	9.061738e+04	20.432721	365.816846
O2SaturationPulseoxymetry_min	61398.0	-272.000000	1.000000e+02	91.902054	6.461443
O2SaturationPulseoxymetry_max	61398.0	0.000000	9.765430e+06	298.595492	39810.994318
O2SaturationPulseoxymetry_std	61398.0	0.000000	1.953067e+06	41.813746	7962.191079
O2SaturationPulseoxymetry_mean	61398.0	0.000000	3.907094e+05	104.839957	1592.435365
gcs_eye_min	61262.0	1.000000	4.000000e+00	2.841974	1.257881
gcs_verbal_min	61244.0	1.000000	5.000000e+00	3.261299	1.873632
gcs_motor_min	61239.0	1.000000	6.000000e+00	4.676187	2.018258
GCS_total_min	61213.0	3.000000	1.500000e+01	10.778609	4.811023
Hematocrit_mi	60964.0	4.300000	6.560000e+00	31.111323	6.470006

n			1		
Hematocrit_max	60964.0	12.500000	6.860000e+01	33.688588	5.721579
Hematocrit_mean	60964.0	12.150000	6.560000e+01	32.426576	5.835577
Hemoglobin_min	60848.0	1.300000	1.980000e+01	10.320760	2.200343
Hemoglobin_max	60848.0	3.500000	2.330000e+01	11.081544	2.012316
Hemoglobin_mean	60848.0	3.500000	2.020000e+01	10.707723	2.029918
WhiteBloodCells_min	60856.0	0.000000	4.717000e+02	10.608943	8.035963
WhiteBloodCells_max	60856.0	0.100000	5.725000e+02	12.851822	10.629297
WhiteBloodCells_mean	60856.0	0.075000	4.747000e+02	11.708465	9.028044
AnionGap_min	61343.0	-9.000000	4.900000e+01	13.227931	3.553769
AnionGap_max	61343.0	-3.000000	8.900000e+01	15.214124	4.386365
AnionGap_mean	61343.0	-3.000000	5.242857e+01	14.194499	3.699778
Bicarbonate_min	61409.0	2.000000	5.000000e+01	22.534400	4.687770
Bicarbonate_max	61409.0	4.000000	5.000000e+01	24.256127	4.219905
Bicarbonate_mean	61409.0	4.000000	5.000000e+01	23.410071	4.284020
CalciumTotal_min	56589.0	0.000000	1.640000e+01	8.219216	0.820095
CalciumTotal_max	56589.0	2.900000	2.750000e+01	8.529803	0.792658
CalciumTotal_mean	56589.0	2.900000	1.810000e+01	8.375116	0.737291
Creatinine_min	61400.0	0.000000	3.180000e+01	1.307243	1.426422
Creatinine_max	61400.0	0.000000	8.000000e+01	1.479927	1.713432
Creatinine_me	61400.0	0.000000	3.195000e+01	1.393140	1.542919

an			1		
Chloride_min	61418.0	57.000000	1.500000e+0 2	102.938557	6.076142
Chloride_max	61418.0	67.000000	1.550000e+0 2	105.261666	6.212960
Chloride_mean	61418.0	67.000000	1.522500e+0 2	104.108689	5.924997
Glucose_min	61174.0	8.000000	1.407000e+0 3	122.413836	44.182769
Glucose_max	61174.0	25.000000	2.440000e+0 3	153.749322	85.380607
Glucose_mean	61174.0	25.000000	1.407000e+0 3	137.034874	54.349701
Potassium_min	61374.0	0.800000	9.500000e+0 0	3.977010	0.565022
Potassium_max	61374.0	1.600000	2.650000e+0 1	4.396056	0.738640
Potassium_mean	61374.0	1.600000	9.500000e+0 0	4.179056	0.567483
Sodium_min	61406.0	80.000000	1.750000e+0 2	137.440641	4.886036
Sodium_max	61406.0	107.000000	1.850000e+0 2	139.394831	4.718235
Sodium_mean	61406.0	104.222222	1.790000e+0 2	138.430557	4.581374
UreaNitrogen_min	61390.0	1.000000	2.630000e+0 2	23.250741	19.503164
UreaNitrogen_max	61390.0	1.000000	2.630000e+0 2	26.067747	21.763884
UreaNitrogen_mean	61390.0	1.000000	2.630000e+0 2	24.661343	20.468073
Magnesium_min	59078.0	0.000000	9.200000e+0 0	1.923459	0.381405
Magnesium_max	59078.0	0.700000	8.600000e+0 1	2.164708	0.788131
Magnesium_mean	59078.0	0.700000	2.285000e+0 1	2.039315	0.402768
Phosphate_min	56732.0	0.000000	1.570000e+0 1	3.366372	1.180196
Phosphate_max	56732.0	0.400000	5.800000e+0	3.869067	1.432457

x			1		
Phosphate_mean	56732.0	0.400000	2.153333e+01	3.611545	1.215900
Lactate_min	32989.0	0.000000	2.360000e+01	1.712493	1.220542
Lactate_max	32989.0	0.200000	1.276103e+06	41.355994	7025.872560
Lactate_mean	32989.0	0.200000	4.253684e+05	15.062806	2341.954977
pO2_min	32624.0	0.000000	5.870000e+02	89.523572	57.496399
pO2_max	32624.0	14.000000	4.242000e+03	196.811213	139.946250
pO2_mean	32624.0	14.000000	2.155500e+03	135.214484	81.840868
pCO2_min	32620.0	7.000000	2.010000e+02	37.861588	9.775477
pCO2_max	32620.0	12.000000	2.460000e+02	46.788688	12.765689
pCO2_mean	32620.0	10.400000	2.285000e+02	41.967997	9.871540
PlateletCount_min	60858.0	5.000000	2.001000e+03	196.321864	102.172561
PlateletCount_max	60858.0	5.000000	2.360000e+03	216.273226	105.824739
PlateletCount_mean	60858.0	5.000000	2.180500e+03	206.126589	102.779308
SystolicBloodPressure_min	61244.0	-94.000000	2.470000e+02	96.158222	17.762233
SystolicBloodPressure_max	61244.0	0.000000	1.025100e+06	163.036200	4224.623773
SystolicBloodPressure_std	61244.0	0.000000	1.511260e+05	16.637943	628.081601
SystolicBloodPressure_mean	61244.0	0.000000	2.239572e+04	118.823748	97.656137
DiastolicBloodPressure_min	61241.0	-16.000000	1.380000e+02	49.283176	12.507403
DiastolicBloodPressure_max	61241.0	0.000000	1.141090e+05	117.634412	1431.252995
DiastolicBlood	61241.0	0.000000	2.777657e+0	16.614087	316.969393

Pressure_std			4		
DiastolicBloodPressure_mean	61241.0	0.000000	1.141033e+04	65.984686	80.332866
MeanBloodPressure_min	61234.0	-22767.000000	2.150000e+02	61.724121	93.207792
MeanBloodPressure_max	61234.0	0.000000	1.234570e+06	133.532433	5110.894045
MeanBloodPressure_std	61234.0	0.000000	3.193490e+05	18.072735	1310.516274
MeanBloodPressure_mean	61234.0	0.000000	8.039653e+04	79.956268	328.212413
Temperature_min	59769.0	0.000000	1.045000e+02	97.265292	4.057076
Temperature_max	59769.0	0.000000	9.834000e+03	99.639196	43.825201
Temperature_std	59769.0	0.000000	4.353960e+03	2.577234	20.428069
Temperature_mean	59769.0	0.000000	2.045400e+03	98.293862	8.673932
age	61428.0	18.000000	1.020000e+02	64.472260	16.943624

Fonte: Elaboração Própria

G - Tabela das estatísticas descritivas das variáveis numéricas do conjunto D1 após da imputação de valores

Tabela A5 - Estatísticas descritivas das variáveis numéricas do conjunto D1 após da imputação de valores

Variável	Contagem	Mínimo	Máximo	Média	Desvio Padrão
HeartRate_min	61428.0	0.000000	1.670000e+02	70.388601	15.099848
HeartRate_max	61428.0	32.000000	1.010900e+04	102.793856	63.647585
HeartRate_std	61428.0	0.000000	2.501550e+03	8.674368	15.552854
HeartRate_mean	61428.0	28.500000	8.165000e+02	84.378303	16.143301

Respiratory Rate_min	61428.0	0.000000	3.800000e+01	11.944953	4.272797
Respiratory Rate_max	61428.0	8.000000	2.355560e+06	66.395124	9504.087763
Respiratory Rate_std	61428.0	0.000000	4.619595e+05	11.468157	1863.908788
Respiratory Rate_mean	61428.0	1.689655	9.061738e+04	20.433591	365.572598
O2SaturationPulseoxymetry_min	61428.0	-272.000000	1.000000e+02	91.902044	6.459865
O2SaturationPulseoxymetry_max	61428.0	0.000000	9.765430e+06	298.627693	39801.271620
O2SaturationPulseoxymetry_std	61428.0	0.000000	1.953067e+06	41.820191	7960.246541
O2SaturationPulseoxymetry_mean	61428.0	0.000000	3.907094e+05	104.841241	1592.046458
gcs_eye_min	61428.0	1.000000	4.000000e+00	2.841993	1.256180
gcs_verbal_min	61428.0	1.000000	5.000000e+00	3.261325	1.870824
gcs_motor_min	61428.0	1.000000	6.000000e+00	4.676220	2.015151
GCS_total_min	61428.0	3.000000	1.500000e+01	10.778701	4.802597
Hematocrit_min	61428.0	4.300000	6.560000e+01	31.111331	6.445523
Hematocrit_max	61428.0	12.500000	6.860000e+01	33.688567	5.699929
Hematocrit_mean	61428.0	12.150000	6.560000e+01	32.426583	5.813495
Hemoglobin_min	61428.0	1.300000	1.980000e+01	10.320758	2.189930
Hemoglobin_max	61428.0	3.500000	2.330000e+01	11.081521	2.002793
Hemoglobin_mean	61428.0	3.500000	2.020000e+01	10.707715	2.020312
WhiteBlood Cells_min	61428.0	0.000000	4.717000e+02	10.608560	7.998462

WhiteBlood Cells_max	61428.0	0.100000	5.725000e+0 2	12.851387	10.579693
WhiteBlood Cells_mean	61428.0	0.075000	4.747000e+0 2	11.708070	8.985913
AnionGap_ min	61428.0	-9.000000	4.900000e+0 1	13.227928	3.551310
AnionGap_ max	61428.0	-3.000000	8.900000e+0 1	15.214115	4.383330
AnionGap_ mean	61428.0	-3.000000	5.242857e+0 1	14.194493	3.697217
Bicarbonate _min	61428.0	2.000000	5.000000e+0 1	22.534405	4.687045
Bicarbonate _max	61428.0	4.000000	5.000000e+0 1	24.256130	4.219252
Bicarbonate _mean	61428.0	4.000000	5.000000e+0 1	23.410075	4.283357
CalciumTota l_min	61428.0	0.000000	1.640000e+0 1	8.219087	0.787131
CalciumTota l_max	61428.0	2.900000	2.750000e+0 1	8.529586	0.760797
CalciumTota l_mean	61428.0	2.900000	1.810000e+0 1	8.374980	0.707655
Creatinine_ min	61428.0	0.000000	3.180000e+0 1	1.307239	1.426097
Creatinine_ max	61428.0	0.000000	8.000000e+0 1	1.479923	1.713041
Creatinine_ mean	61428.0	0.000000	3.195000e+0 1	1.393136	1.542567
Chloride_mi n	61428.0	57.000000	1.500000e+0 2	102.938555	6.075648
Chloride_ma x	61428.0	67.000000	1.550000e+0 2	105.261664	6.212454
Chloride_me an	61428.0	67.000000	1.522500e+0 2	104.108686	5.924515
Glucose_mi n	61428.0	8.000000	1.407000e+0 3	122.414330	44.091327
Glucose_ma x	61428.0	25.000000	2.440000e+0 3	153.749519	85.203900
Glucose_me an	61428.0	25.000000	1.407000e+0 3	137.035263	54.237217

Potassium_min	61428.0	0.800000	9.500000e+00	3.977005	0.564774
Potassium_max	61428.0	1.600000	2.650000e+01	4.396051	0.738315
Potassium_mean	61428.0	1.600000	9.500000e+00	4.179051	0.567233
Sodium_min	61428.0	80.000000	1.750000e+02	137.440641	4.885161
Sodium_max	61428.0	107.000000	1.850000e+02	139.394832	4.717390
Sodium_mean	61428.0	104.222222	1.790000e+02	138.430558	4.580553
UreaNitrogen_min	61428.0	1.000000	2.630000e+02	23.250707	19.497131
UreaNitrogen_max	61428.0	1.000000	2.630000e+02	26.067711	21.757151
UreaNitrogen_mean	61428.0	1.000000	2.630000e+02	24.661309	20.461741
Magnesium_min	61428.0	0.000000	9.200000e+00	1.923442	0.374038
Magnesium_max	61428.0	0.700000	8.600000e+01	2.164659	0.772909
Magnesium_mean	61428.0	0.700000	2.285000e+01	2.039296	0.394988
Phosphate_min	61428.0	0.000000	1.570000e+01	3.365968	1.134188
Phosphate_max	61428.0	0.400000	5.800000e+01	3.868600	1.376615
Phosphate_mean	61428.0	0.400000	2.153333e+01	3.611159	1.168500
Lactate_min	61428.0	0.000000	2.360000e+01	1.712955	0.894440
Lactate_max	61428.0	0.200000	1.276103e+06	49.063708	5148.718793
Lactate_mean	61428.0	0.200000	4.253684e+05	17.632372	1716.237735
pO2_min	61428.0	0.000000	5.870000e+02	89.505285	41.900865
pO2_max	61428.0	14.000000	4.242000e+03	196.865628	101.986721

pO2_mean	61428.0	14.000000	2.155500e+03	135.214337	59.642045
pCO2_min	61428.0	7.000000	2.010000e+02	37.876013	7.123520
pCO2_max	61428.0	12.000000	2.460000e+02	46.804028	9.302520
pCO2_mean	61428.0	10.400000	2.285000e+02	41.985731	7.193530
PlateletCount_min	61428.0	5.000000	2.001000e+03	196.321371	101.697411
PlateletCount_max	61428.0	5.000000	2.360000e+03	216.271780	105.332606
PlateletCount_mean	61428.0	5.000000	2.180500e+03	206.125687	102.301336
SystolicBloodPressure_min	61428.0	-94.000000	2.470000e+02	96.158191	17.735610
SystolicBloodPressure_max	61428.0	0.000000	1.025100e+06	163.056820	4218.291772
SystolicBloodPressure_std	61428.0	0.000000	1.511260e+05	16.640493	627.140212
SystolicBloodPressure_mean	61428.0	0.000000	2.239572e+04	118.824020	97.509766
DiastolicBloodPressure_min	61428.0	-16.000000	1.380000e+02	49.283276	12.488351
DiastolicBloodPressure_max	61428.0	0.000000	1.141090e+05	117.639346	1429.072781
DiastolicBloodPressure_std	61428.0	0.000000	2.777657e+04	16.615585	316.486558
DiastolicBloodPressure_mean	61428.0	0.000000	1.141033e+04	65.985128	80.210496
MeanBloodPressure_min	61428.0	-22767.000000	2.150000e+02	61.723668	93.060491
MeanBloodPressure_max	61428.0	0.000000	1.234570e+06	133.569408	5102.817040

MeanBloodPressure_std	61428.0	0.000000	3.193490e+05	18.081738	1308.445197
MeanBloodPressure_mean	61428.0	0.000000	8.039653e+04	79.958346	327.693722
Temperature_min	61428.0	0.000000	1.045000e+02	97.265171	4.001915
Temperature_max	61428.0	0.000000	9.834000e+03	99.639737	43.229343
Temperature_std	61428.0	0.000000	4.353960e+03	2.576513	20.150324
Temperature_mean	61428.0	0.000000	2.045400e+03	98.294049	8.555999
age	61428.0	18.000000	1.020000e+02	64.472260	16.943624

Fonte: Elaboração própria

H – Resultados do GridSearch CV em cada conjunto

Tabela A6 – Resultados do GridSearchCV em cada conjunto

Modelo XGB							
Conjunto a ser usado	eta	max_depth	n_estimators	subsample	AUPRC média	Tempo médio de fit(s)	Desvio tempo médio(s)
D1	0,1	3	500	0,7	0,5604 (IC=[0,519; 0,6018])	2,129673	0,457203
D2	0,1	3	200	0,5	0,5483 (IC=[0,5227; 0,5739])	1,553019	0,91773
D3	0,1	3	200	1,0	0,5124 (IC=[0,4598; 0,565])	0,723619	0,022371
D4	0,1	3	200	0,7	0,5055 (IC=[0,4692; 0,5418])	0,738425	0,014018
Modelo MLP							
Conjunto a ser usado	learning_rate_init	hidden_layers_sizes	early_stopping	max_iter	AUPRC média	Tempo médio de fit(s)	Desvio tempo médio
D1	0,001	(100,100)	True	20	0,4893	13,18583	0,261163

					(IC=[0,4336; 0,545])	6	
D2	0,001	(100,100)	True	20	0,5102 (IC=[0,4516; 0,5688])	12,99440 3	0,198729
D3	0,001	(50,100,50)	True	50	0,4419 (IC=[0,3859; 0,4979])	2,809154	0,306468
D4	0,001	(50,100,50)	True	50	0,4646 (IC=[0,4266; 0,5026])	2,970915	0,483048
Modelo MLP com arquitetura do HyperFast							
Conjunt o a ser usado	learning_r ate_init	hidden_lay ers_sizes	early_s toppin g	max_it er	AUPRC média	Tempo médio de fit(s)	Desvio tempo médio
D1	0,001	(1024,1024)	True	50	0,4936 (IC=[0,4504; 0,5368])	409,4766 19	48,803894
D2	0,001	(1024,1024)	True	50	0,5118 (IC=[0,4571; 0,5665])	513,3761 25	129,77960 2
D3	0,0001	(1024,1024)	True	50	0,4363 (IC=[0,35; 0,5226])	75,75691 2	18,254999
D4	0,001	(1024,1024)	True	50	0,4524 (IC=[0,4113; 0,4935])	45,58766 5	11,178225
Modelo HyperFast em CPU							
Conjunt o a ser usado	batch_size	n_ensemle	optimze_steps		AUPRC média	Tempo médio de fit(s)	Desvio tempo médio
D1	2048	1	16		0,3882 (IC=[0,2997; 0,4768])	174,4274 51	1,009873
D2	2048	1	16		0,3989 (IC=[0,3889; 0,4089])	172,0604 06	0,536139
D3	2048	1	8		0,414 (IC=[0,3816; 0,4463])	131,7482 81	0,612842
D4	2048	1	8		0,4355 (IC=[0,3904; 0,4807])	131,4032 74	0,101333

Modelo HyperFast em CPU sem optimization						
Conjunt o a ser usado	batch_size	n_ensemle	optimzation	AUPRC média	Tempo médio de fit(s)	Desvio tempo médio
D1	1024	1	None	0,315 (IC=[0,2529; 0,377])	13,34902 3	0,128807
D2	2048	1	None	0,3169 (IC=[0,255; 0,3789])	92,80191	1,7795
D3	1024	1	None	0,3168 (IC=[0,2665; 0,3672])	12,93592 8	0,023339
D4	2048	1	None	0,3564 (IC=[0,3017; 0,4111])	95,66774	1,33648
Modelo HyperFast em GPU						
Conjunt o a ser usado	batch_size	n_ensemle	optimze_steps	AUPRC média	Tempo médio de fit(s)	Desvio tempo médio
D1	2048	1	16	0,3891 (IC=[0,3002; 0,478])	3,260064	0,081627
D2	2048	1	16	0,3972 (IC=[0,3885; 0,406])	3,28998	0,091641
D3	2048	1	8	0,414 (IC=[0,3814; 0,4467])	2.257852	0.032834
D4	2048	1	8	0,4352 (IC=[0,3897; 0,4807])	2.248059	0.053534
Modelo HyperFast em GPU sem optimization						
Conjunt o a ser usado	batch_size	n_ensemle	optimzation	AUPRC média	Tempo médio de fit(s)	Desvio tempo médio
D1	1024	1	None	0,3151 (IC=[0,253; 0,3771])	1,158738	0,027001
D2	2048	1	None	0,3182 (IC=[0,2584; 0,378])	1,859441	0,086075

D3	1024	1	None	0,3169 (IC=[0,2667; 0,3672])	0,813726	0,019218
D4	2048	1	None	0,3563 (IC=[0,3014; 0,4113])	1,521907	0,010786

Fonte: Elaboração própria

I – Tabelas com os resultados dos modelos no conjunto de Teste de D1, D2 e D3.

Tabela A7 - Resultados dos modelos no conjunto de teste de D1

Modelo	AUC ROC	AUPRC	Balanced Accuracy	Precision Macro
XGBoost	0.9145 (IC95%: 0.9067 – 0.9221)	0.5542 (IC95%: 0.5249 – 0.5807)	0.6641 (IC95%: 0.6499 – 0.6775)	0.8350 (IC95%: 0.8145 – 0.8537)
MLP	0.8916 (IC95%: 0.8817 – 0.9005)	0.4712 (IC95%: 0.4396 – 0.5018)	0.6445 (IC95%: 0.6299 – 0.6581)	0.7790 (IC95%: 0.7571 – 0.7982)
MLP com arquitetura similar ao HyperFast	0.8923 (IC95%: 0.8827 – 0.9017)	0.4657 (IC95%: 0.4352 – 0.4971)	0.6333 (IC95%: 0.6202 – 0.6459)	0.8081 (IC95%: 0.7867 – 0.8292)
HyperFast em CPU	0.8445 (IC95%: 0.8328 – 0.8569)	0.3949 (IC95%: 0.3675 – 0.4251)	0.6604 (IC95%: 0.6469 – 0.6746)	0.7016 (IC95%: 0.6857 – 0.7185)
HyperFast em CPU sem optimization	0.8091 (IC95%: 0.7962 – 0.8226)	0.2826 (IC95%: 0.2575 – 0.3104)	0.7316 (IC95%: 0.7175 – 0.7464)	0.5704 (IC95%: 0.5647 – 0.5763)
HyperFast em GPU	0.8445 (IC95%: 0.8328 – 0.8570)	0.3946 (IC95%: 0.3673 – 0.4248)	0.6582 (IC95%: 0.6450 – 0.6725)	0.7009 (IC95%: 0.6853 – 0.7172)
HyperFast em GPU sem optimization	0.8091 (IC95%: 0.7963 – 0.8226)	0.2827 (IC95%: 0.2576 – 0.3105)	0.7312 (IC95%: 0.7170 – 0.7459)	0.5703 (IC95%: 0.5647 – 0.5762)

Fonte: Elaboração própria

Tabela A8 - Resultados dos modelos no conjunto de teste de D2

Modelo	AUC ROC	AUPRC	Balanced Accuracy	Precision Macro
XGBoost	0.9147 (IC95%: 0.9061 – 0.9225)	0.5396 (IC95%: 0.5082 – 0.5695)	0.6547 (IC95%: 0.6413 – 0.6688)	0.8450 (IC95%: 0.8241 – 0.8641)
MLP	0.8958 (IC95%: 0.8865 – 0.9044)	0.4927 (IC95%: 0.4608 – 0.5226)	0.6512 (IC95%: 0.6375 – 0.6655)	0.8044 (IC95%: 0.7829 – 0.8245)

MLP arquitetura similar ao HyperFast	com	0.8978 (IC95%: 0.8885 – 0.9068)	0.4974 (IC95%: 0.4673 – 0.5285)	0.6602 (IC95%: 0.6455 – 0.6745)	0.7965 (IC95%: 0.7771 – 0.8172)
HyperFast CPU	em	0.8506 (IC95%: 0.8385 – 0.8622)	0.3817 (IC95%: 0.3509 – 0.4152)	0.6359 (IC95%: 0.6228 – 0.6496)	0.7550 (IC95%: 0.7349 – 0.7768)
HyperFast CPU optimization	em sem	0.8359 (IC95%: 0.8238 – 0.8477)	0.3023 (IC95%: 0.2756 – 0.3290)	0.7497 (IC95%: 0.7359 – 0.7639)	0.5817 (IC95%: 0.5754 – 0.5878)
HyperFast GPU	em	0.8512 (IC95%: 0.8392 – 0.8628)	0.3816 (IC95%: 0.3510 – 0.4147)	0.6345 (IC95%: 0.6207 – 0.6485)	0.7533 (IC95%: 0.7329 – 0.7753)
HyperFast GPU optimization	em sem	0.8359 (IC95%: 0.8239 – 0.8477)	0.3029 (IC95%: 0.2760 – 0.3295)	0.7490 (IC95%: 0.7352 – 0.7632)	0.5814 (IC95%: 0.5753 – 0.5876)

Fonte: Elaboração própria

Tabela A9 - Resultados dos modelos no conjunto de teste de D3

Modelo		AUC ROC	AUPRC	Balanced Accuracy	Precision Macro
XGBoost		0.9063 (IC95%: 0.8973 – 0.9145)	0.5132 (IC95%: 0.4807 – 0.5439)	0.8289 (IC95%: 0.8176 – 0.8399)	0.6134 (IC95%: 0.6065 – 0.6204)
MLP		0.8759 (IC95%: 0.8662 – 0.8859)	0.4112 (IC95%: 0.3807 – 0.4426)	0.7862 (IC95%: 0.7765 – 0.7963)	0.5710 (IC95%: 0.5666 – 0.5755)
MLP arquitetura similar ao HyperFast	com	0.8834 (IC95%: 0.8738 – 0.8931)	0.4330 (IC95%: 0.4037 – 0.4648)	0.7915 (IC95%: 0.7809 – 0.8022)	0.5754 (IC95%: 0.5708 – 0.5801)
HyperFast CPU	em	0.8763 (IC95%: 0.8658 – 0.8863)	0.4148 (IC95%: 0.3843 – 0.4473)	0.7977 (IC95%: 0.7851 – 0.8096)	0.5885 (IC95%: 0.5828 – 0.5943)
HyperFast CPU optimization	em sem	0.8210 (IC95%: 0.8093 – 0.8330)	0.2787 (IC95%: 0.2525 – 0.3061)	0.7402 (IC95%: 0.7273 – 0.7529)	0.5642 (IC95%: 0.5595 – 0.5693)
HyperFast GPU	em	0.8762 (IC95%: 0.8656 – 0.8861)	0.4152 (IC95%: 0.3847 – 0.4477)	0.7952 (IC95%: 0.7827 – 0.8068)	0.5877 (IC95%: 0.5821 – 0.5934)
HyperFast GPU optimization	em sem	0.8210 (IC95%: 0.8093 – 0.8330)	0.2783 (IC95%: 0.2521 – 0.3058)	0.7398 (IC95%: 0.7270 – 0.7528)	0.5641 (IC95%: 0.5594 – 0.5693)

Fonte: Elaboração própria

J – Artigo

Comparação entre algoritmos HyperFast, MLP e XGBoost para previsão de mortalidade em UTI

Anderson Sales Zambeli¹, André Wüst Zibetti¹

¹Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

Caixa Postal 476 – CEP 88040-370 – Florianópolis – SC – Brazil

²Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

Caixa Postal 476 – CEP 88040-370 – Florianópolis – SC – Brazil

Abstract. *This paper presents a comparative analysis of the HyperFast, Multilayer Perceptron (MLP), and XGBoost algorithms for predicting seven-day mortality after ICU admission. Using MIMIC-IV data from the first 24 hours of hospitalization, four datasets were constructed varying in class balance and outlier removal. XGBoost obtained the best predictive performance (AUPRC 0.5134), while HyperFast showed competitive results but required substantial computational resources.*

Resumo. *Este artigo apresenta uma análise comparativa dos algoritmos HyperFast, Multilayer Perceptron (MLP) e XGBoost na previsão de mortalidade em até sete dias após a admissão na UTI. Utilizando dados do MIMIC-IV referentes às primeiras 24 horas da internação, foram construídos quatro conjuntos de dados variando quanto ao balanceamento e remoção de outliers. O XGBoost apresentou o melhor desempenho preditivo (AUPRC 0.5134), enquanto o HyperFast mostrou desempenho intermediário, porém com elevadas exigências computacionais.*

1. Introdução

Entender e estudar as ocupações das Unidades de Terapia Intensiva (UTI) é algo de suma importância. Elas são cruciais no tratamento de pessoas em estado gravíssimo provendo equipamento especializado, tratamento priorizado e suporte a funções vitais de paciente, e por isso afetam diretamente o resultado de um tratamento delicado. Porém seu custo operacional é elevado em relação às outras unidades, envolvendo uma infraestrutura complexa, equipamentos de alta tecnologia e equipes multidisciplinares especializadas. Por esse motivo, recomenda-se que as taxas de ocupação se mantenham entre 70% e 75%, de forma a equilibrar eficiência econômica e capacidade de resposta a

aumentos inesperados de demanda, como em pandemias ou surtos sazonais (Valentin et al., 2020).

No contexto brasileiro, a expansão dos leitos de terapia intensiva tem avançado na última década, com crescimento aproximado de 52% entre 2014 e 2024, atingindo mais de 73 mil leitos em operação. Apesar dessa ampliação, persistem acentuadas desigualdades regionais, nas quais a oferta é significativamente maior em grandes centros urbanos e regiões mais desenvolvidas, enquanto áreas periféricas e municípios de pequeno porte apresentam baixa cobertura (AMIB, 2024). Estudos do Conselho Federal de Medicina indicam que menos de 10% dos municípios brasileiros dispõem de ao menos um leito de UTI, o que demonstra a forte concentração territorial desses recursos (CFM, 2023).

Com isso em mente, ter maneiras de prever quem precisa de uma UTI, o desfecho da internação, tempo de estadia e o perfil das pessoas que eventualmente fazem o uso dos leitos de uma UTI, é uma informação interessante para as tomadas de decisão sobre as políticas de saúde que envolvem a expansão, manutenção e logística dos leitos de UTI.

Tendo em vista o grande volume de dados coletados pelos sistemas de saúde privados e governamentais, o uso de técnicas de mineração de dados se apresenta como uma maneira interessante de extrair informações e criar aplicações, no âmbito de UTIs, que possam auxiliar na otimização dos recursos, tomadas de decisões rápidas e administração de leitos. Exemplos dessas aplicações são prever se um paciente vai morrer em até um certo número de dias após sua entrada no hospital (ZHAI et al., 2020), dizer se a estadia de um paciente vai ser de longa duração (WU et al., 2021) ou prever se o paciente vai desenvolver choque séptico durante sua estadia em uma UTI (NEMATİ et al, 2018).

Novos algoritmos de aprendizado de máquina, uma técnica da mineração de dados, têm surgido nos últimos anos, trazendo novas inovações para a área. Logo, avaliá-los no uso da área médica é importante para entender como e onde seu uso traz ganhos, quais suas limitações e vantagens e que cenários eles são melhores indicados.

Este trabalho visa fazer um estudo de métodos de aprendizado de máquina para criar modelos de predição de mortalidade após sete dias da admissão na UTI, baseado nos dados coletados nas primeiras vinte e quatro horas após entrada na UTI. A escolha da mortalidade em até 7 dias como desfecho neste trabalho pode ser entendida como uma especialização da mortalidade precoce em UTI, em especial dando atenção ao algoritmo HyperFast, publicado em fevereiro de 2024, um algoritmo que usa meta-learning e hiper-redes para dados tabulares. Clinicamente, uma janela de 7 dias é suficiente para captar a maior densidade dos óbitos diretamente relacionados à gravidade inicial e ao manejo

intensivo, evitando a influência excessiva de fatores tardios, como infecções hospitalares adquiridas durante a estadia (MARTIN-LOECHES et al, 2018).

1.1 Objetivos

1.1.1 Objetivo Geral

Criar modelos de predição de mortalidade em até sete dias após a primeira entrada na Unidade Terapia Intensiva e compará-los, em especial ao modelo gerado pelo algoritmo HyperFast.

1.1.1 Objetivos Específicos

- Criar modelos de machine learning (Rede Neurais MLP, XGBoost e HyperFast), para predição de mortalidade em até sete dias após a primeira entrada na Unidade Terapia Intensiva, usando os dados coletados nas primeiras vinte quatro horas da estadia.
- Comparar os modelos gerados pelos algoritmos MLP e XGBoost com os modelos gerados pelo HyperFast em relação a presença de outliers, desbalanceamento entre classes e tempo para treinar e testar o modelo.

2. Trabalhos Relacionados

A seguir, apresenta-se um levantamento de estudos que abordam predição de mortalidade em diferentes contextos clínicos, variando quanto ao tipo de população, momento da coleta das variáveis e algoritmos utilizados. Esses trabalhos ilustram a diversidade de abordagens e métricas empregadas na área, evidenciando como distintos conjuntos de dados e métodos de aprendizado de máquina têm sido aplicados para antecipar desfechos críticos em ambientes hospitalares. A Tabela 1 sintetiza os principais estudos identificados, destacando seu foco, os modelos utilizados e o melhor desempenho obtido em termos de AUC.

Tabela 1 - Trabalhos Relacionados

Autor	Resumo	Melhor modelo em relação a métrica AUC
Saadatmand et al. (2022)	Predição de Mortalidade em pacientes com COVID-19	XGB, 0.928
Taylor et al. (2016)	Predição de morte em pacientes diagnosticados com sepse na admissão	Random Forest, 0.860

Raita et al.(2019)	Predição de admissão em UTI ou morte durante hospitalização	Rede neural Profunda, 0.86
Zhai et al.(2020)	Predição de morte em até 7 dias após admissão em UTIs de emergência	XGB, 0.849
Choi et al.(2022)	Predição de mortalidade durante a hospitalização	XGB e LightGBT, 0.977
Schmidt et al.(2018)	Predição de mortalidade durante a internação em uma UTI usando os primeiros dados coletados de cada variável após a entrada na UTI	Rede neural Convolutacional, 0.83
Veith e Steele (2018)	Predição de morte no momento de admissão usando variáveis administrativas	LazyStar, 0.751

Saadatmand et al. (2022) desenvolveu modelos de predição de mortalidade em pacientes com COVID-19. Saadatmand usou dados de dois hospitais iranianos com pacientes que testaram positivo para COVID-19 para predizer a morte na hospitalização. Foram 963 pacientes incluídos, sendo coletados de cada os sinais vitais, laboratoriais, sintomas, comorbidades e informações demográficas para serem usados no trabalho. Os modelos usados foram Random Forest, XGB, LogitBoost, Bagged CART e k-NearestNeighbors. Entre as métricas usadas, estão a acurácia, recall, especificidade e AUC. O modelo que apresentou melhor resultado foi o XGB com acurácia de 0.78, recall de 0.90, especificidade de 0.58 e AUC de 0.928. O modelo Random Forest teve 0.78, 1.00, 0.41, 0.868.

Taylor et al. (2016) fez modelos de previsão de morte em pacientes diagnosticados com sepse na admissão. Os dados coletados foram de pacientes de 4 hospitais por um período de 12 meses, tendo um total de 4676 pacientes. A variável a ser predita era mortalidade em 28 dias após admissão. Os modelos usados foram a árvore de decisão CART, regressão logística e RF. Além disso, os modelos foram comparados também com ferramentas de predição de mortalidade já conhecidas. O modelo Random Forest teve a melhor performance geral, com o valor AUC de 0.860.

Raita et al. (2019) fez modelos de predição de admissão em UTI ou morte durante hospitalização no momento da triagem no departamento de emergência com as variáveis comumente disponíveis nesse contexto e predição de hospitalização, com dados coletados de vários hospitais nos Estados Unidos coletados no período de 2007 a 2005 pelo National Hospital and Ambulatory Medical Care Survey. Os modelos usados foram regressão logística com regularização Lasso, Random Forest, Gradient Boosted Decision Tree (GBDT) e rede neural profunda. Para primeira predição o modelo de rede

neural profunda teve AUC de 0.86, recall de 0.80 e especificidade de 0.76. O modelo de Random Forest teve AUC de 0.85, recall de 0.86 e especificidade de 0.68.

Zhai et al. (2020) gerou modelos de predição de morte em até 7 dias após admissão em UTIs de emergência. Os modelos usados foram Support Vector Machines, GBDT, XGB e regressão logística. As AUC para os respectivos modelos foram 0.794, 0.840, 0.849 e 0.822.

Choi et al. (2022) fez modelos de predição de mortalidade durante a hospitalização em UTIs com dados de dois hospitais universitários coletados entre 2006 e 2020. Usando variáveis demográficas e os piores valores de variáveis de sinais vitais e laboratoriais, vários modelos foram treinados com algumas configurações diferentes dos dados. Os melhores resultados foram obtidos com os modelos XGB, LightGBM e RF com AUCs de 0.977, 0.977 e 0.972.

Schmidt et al. (2018) fez modelos de predição de mortalidade durante a internação em uma UTI usando os primeiros dados coletados de cada variável após a entrada na UTI. Os modelos foram Redes Neurais Convolucionais usando dados do MIMIC III. A AUC do melhor modelo foi 0.83, com recall de 0.697, especificidade 0.959 e acurácia de 0.916.

Veith e Steele (2018) fizeram modelos de predição de morte no momento de admissão usando variáveis administrativas também com os dados do MIMIC III. Usando algumas configurações diferentes dos dados e os modelos baseados nos algoritmos LazyKStar, Logistic, Simple Logistic, NaïveBayes e BayesNet do software WEKA, conseguiram valores de AUC entre 0.751 e 0.689, sendo 0.751 o do algoritmo LazyKstar.

Os trabalhos apresentados compartilham o objetivo de aprimorar a predição de mortalidade em diferentes cenários clínicos, mas diferem quanto ao contexto de aplicação, às variáveis disponíveis e à complexidade dos modelos utilizados. Em comum, destacam-se a adoção de algoritmos de aprendizado de máquina que superam métodos tradicionais, como os escores de gravidade, em termos de desempenho preditivo, sobretudo quando incorporam grande volume de informações clínicas, como sinais vitais, exames laboratoriais e comorbidades. Entre os pontos fortes, ressaltam-se a capacidade desses modelos de capturar interações não lineares entre variáveis, o uso de bases extensas como MIMIC-III e EHRs hospitalares, e a altos valores de AUC ROC alcançados pelos melhores modelos de cada trabalho. Por outro lado, algumas limitações são recorrentes: falta de detalhamento de fatores importantes como balanceamento entre as classes e foco na métrica AUC ROC na avaliação dos modelos.

Este trabalho além de usar dois tradicionais de aprendizado de máquina (XGboost e Redes Neurais MLP), usa um algoritmo recente (HyperFast). Também é

feito uma análise de tempos e tamanhos dos algoritmos no treinamento e predição, algo que não foi feito pelos trabalhos citados.

3. Metodologia

Nesta parte mostraremos primeiro o dataset usado e depois o processo usado para preparar os dados para serem usados nos modelos escolhidos.

3.1 MIMIC IV

O Medical Information Mart for Intensive Care IV (MIMIC IV) é um conjunto de dados de pacientes críticos anonimizado com informações coletadas no hospital Beth Israel Deaconess Medical Center na cidade de Boston em Massachusetts nos Estados Unidos durante o período de 2008 até 2019 e disponível na Physionet, uma plataforma que mantém bancos de dados da área médica e mantido por membros do Laboratório para Fisiologia Computacional do Massachusetts Institute of Technology.

Os dados vieram de pacientes de mais de 18 anos que ou deram entrada no departamento de emergência do hospital, ou que deram entrada na UTI dele. Após essa coleta, os dados passaram por um processo de anonimização para estar em conformidade com a Health Insurance Portability and Accountability Act (HIPAA), que define as regras e normas para garantir a privacidade e segurança em dados eletrônicos de saúde de pacientes.

O banco é dividido em dois módulos: *hosp* e *icu*. O módulo *hosp* contém os dados coletados durante a estadia de um paciente no hospital após ser admitido do departamento de emergência. O módulo *icu* contém os dados do paciente durante sua estadia nas UTIs do hospital.

Ao todo, na sua versão 2.2, que será usada neste trabalho, o banco contém 431231 hospitalizações únicas de 299712 indivíduos no módulo *hosp* e 73181 estadias em UTI no módulo *icu*.

A tabela de eventos laboratoriais (*labevents*) do módulo *hosp* traz todos os resultados laboratoriais de um paciente durante sua estadia no hospital. O tipo do exame feito é identificado pelo campo *itemid*, e sua definição fica na tabela *d_labitems*, onde é descrito o exame (*label*), sua categoria (*category*) e fluido de coleta (*fluid*). A unidade de medição fica direto na tabela *labevents* (*valueuom*).

A tabela de eventos gerais (*chartevents*) do módulo *icu* traz todos os eventos que ocorreram com um paciente durante sua estadia em uma UTI. Isso inclui medições de sinais vitais, exames laboratoriais, mudanças de medicamento etc. Basicamente

qualquer evento é registrado. O tipo do evento é identificado pelo campo *itemid*, e sua definição fica na tabela *d_items*, onde é descrito o evento (*label*), sua categoria (*category*), unidade de medição quando é aplicável (*valueuom*) e algumas outras informações.

Neste trabalho são usadas variáveis dos pacientes coletadas nas primeiras 24 horas da primeira estadia na UTI de uma admissão no hospital. Essas variáveis são os sinais vitais e exames laboratoriais, além da idade, gênero, os valores mínimos de cada componente do Glasgow Coma Score e o escore usando estes valores, uso de ventilação mecânica nas primeiras 24 horas, uso de ventilação mecânica invasiva nas primeiras 24 horas e o tipo de admissão na UTI. Elas foram escolhidas a partir dos escores de gravidade e de outros trabalhos relevantes, além de explorações no banco de dados. As variáveis de sinais vitais serão agregadas e usadas as médias, os máximos, os mínimos e os desvios padrões das medições. As variáveis laboratoriais somente médias, os máximos e os mínimos.

As variáveis de sinais vitais foram: batimentos cardíacos, frequência respiratória, pressão arterial sistólica, pressão arterial diastólica, pressão arterial média, saturação de oxigênio medida por oximetria de pulso, temperatura corporal, peso na admissão.

As variáveis de exames laboratoriais as variáveis foram: hematócritos, hemoglobina, contagem de células brancas, albumina, proteína, proteína c-reativa, hiato aniônico, bicarbonato, cálcio total, cálcio livre, creatinina, cloreto, glicose, potássio, sódio, nitrogênio ureico no sangue, magnésio, fosfato, lactato, pO₂(pressão parcial de oxigênio no sangue), pCO₂(pressão parcial de dióxido de carbono no sangue) e bilirrubina total.

3.2 Pré-processamento dos Dados

Primeiro foi criado uma coluna única para as variáveis de pressão, onde quando há medição de pressão não invasiva, esse valor é usado, na falta deste, a medição invasiva é usada. Para temperatura uma única coluna foi feita e usando a medição em Fahrenheit que é a mais preenchida, e na falta, a temperatura em Celsius convertida para Fahrenheit.

Por fim, para os registros onde o campo de desvio padrão é nulo, mas temos média, máximo e mínimo preenchido, o que significa que houve somente uma medição daquela variável, o desvio foi colocado como zero.

A idade do paciente no momento da admissão foi extraída usando os campos *anchor_age*, *anchor_year* e *icu_intime*.

Após isso, a variável de interesse, morte em até 7 dias dentro do hospital após a primeira entrada na UTI, foi extraída usando os campos *deathtime*, *hospital_expire_flag* e *icu_intime*.

As variáveis que estavam vazias para mais de cinquenta por cento dos pacientes foram descartadas. Outros campos como *subjetc_id* que não são pertinentes para os modelos foram retirados.

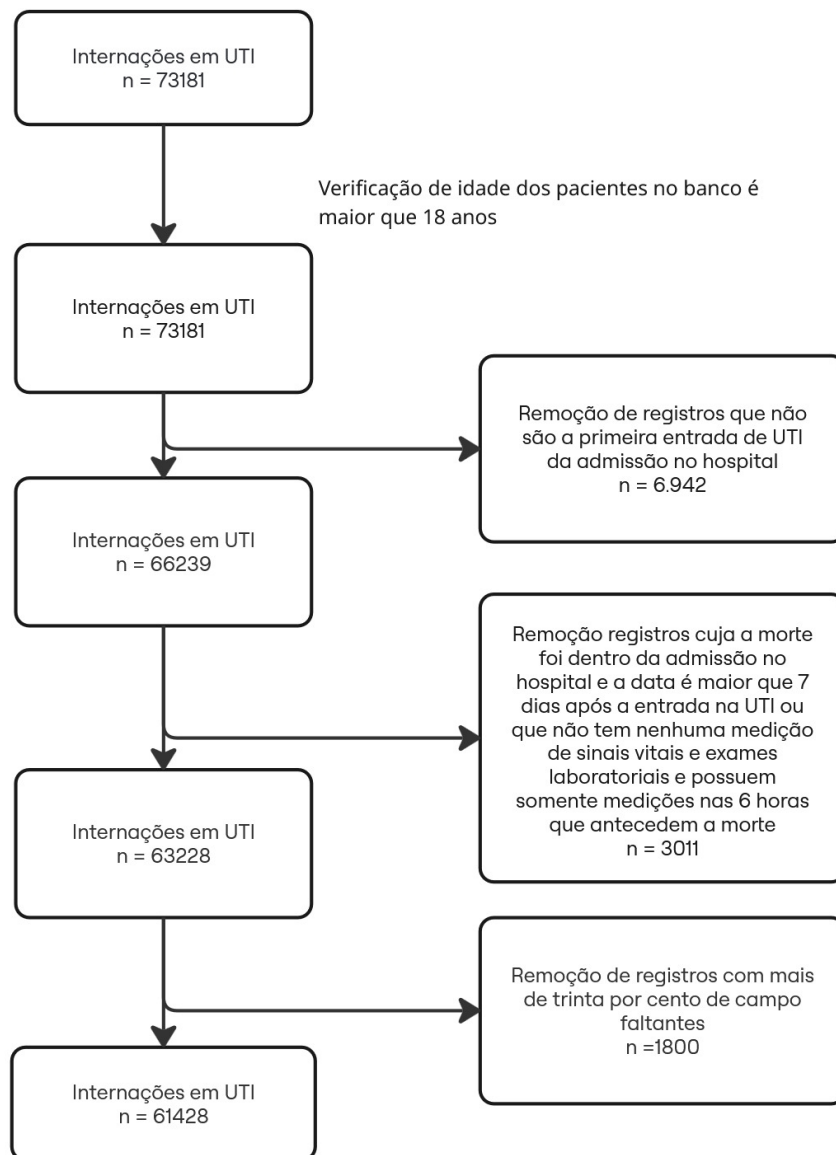


Figura 1 - Fluxo de exclusão de dados

3.3 Exclusão de Dados

O processo de exclusão de dados seguiu o seguinte fluxo (Figura 1): retiradas de registros com idade menor que 18 anos, retirada de registros que não são da primeira entrada de UTI da admissão, retirada de registros cuja morte foi dentro da admissão no hospital e a data é maior que 7 dias após a entrada na UTI ou que não possui nenhuma medida laboratorial ou de sinais vitais, e retirada de registros com mais de trinta por cento de campos faltantes. Além disso, estamos desconsiderando medidas que foram coletadas nas 6 horas que antecedem a morte de um paciente, os chamados dados agonais. Embora não exista um consenso sobre o período agonal, a remoção de dados

entre 4 a 12 horas anteriores ao evento é usada na literatura de ML clínico (BOUVAREL, CARRAT, LAPIDUS; 2023).

3.4 Criação dos conjuntos

Para as análises, quatro conjuntos de dados foram criados. O primeiro conjunto, chamado D1, são os dados da maneira como estão, sem eliminação de outliers e sem fazer balanceamento de classes nos dados de treino. Valores nulos foram imputados usando o *SimpleImputer* do scikit-learn, um imputador que preenche os dados faltantes com o valor definido pelo usuário, que neste trabalho foi a mediana da coluna. Os dados foram separados em treino e teste aqui, seguindo a proporção 70% para treino e 30% para teste e também garantindo que a proporção entre as classes foi a mesma do dataset inteiro nos dois conjuntos. A proporção após a eliminação dos registros ficou em 6,02% para a classe positiva e 93,98% para a classe negativa.

O segundo conjunto, D2, passou pela remoção de medições com valores outliers para calcular máximos, mínimos, médias e desvios. Os valores outliers foram definidos como valores que ficam fora dos mínimos e máximos possíveis para a variável em um ser humano, decididos a partir de pesquisas sobre as variáveis. Também foi garantido que todas as estadias em D1 sejam as mesmas em D2, porém com os valores das variáveis diferentes em virtude da retirada de medições outliers. Depois foi feita a imputação de valores faltantes usando o *SimpleImputer* usando a mediana da coluna como valor imputado para as variáveis numéricas, visto que as variáveis categóricas estavam completamente preenchidas. Nesse estudo, os valores limites para dizer se um valor é ou não outlier, foram escolhidos manualmente a partir do que se julga como um valor possível ou não daquela variável, de acordo com a literatura. Após a exclusão e das medidas outliers, foram retirados 252 registros por ficarem com mais 30% de campos nulos.

O conjunto D3, que vem de D1, não terá a eliminação de outliers e o balanceamento de classes no conjunto de treino será feito. O balanceamento é feito usando a técnica de *undersampling*, onde retiramos valores das classes majoritárias até atingir uma proporção desejada entre as classes. Aqui essa proporção escolhida foi de 1 para 1.

O conjunto D4, feito a partir de D2, passa pela eliminação de outliers e balanceamento de classes no conjunto de treino.

As variáveis categóricas foram transformadas em numéricas usando o *get_dummies* do Pandas que faz *one hot encoding* delas e os dados de cada conjunto foi dividido em treino e teste para seguindo a proporção 70 por cento de treino e trinta por cento para teste, mantendo a proporção de classes dos dados originais em cada

conjunto de treino e teste. Foi garantido que um dado que aparece no conjunto de treino de um conjunto não apareça no conjunto de teste de outro. As variáveis numéricas foram normalizadas usando o *MinMaxScaler* do scikit-learn usando como base os dados de treino para ajustar o normalizador após as imputações.

Visto que a falta de uma variável, em especial as variáveis laboratoriais que são coletadas no em torno de 2 vezes no máximo nas primeiras 24 horas, pode não ser um valor faltante aleatório (*Missing Completely at Random* ou *Missing at Random*) e ter chance de possuir um significado clínico, para cada variável foi criada uma coluna que indica se para aquele registro, determinada variável estava faltante e foi imputada. Por exemplo, um exame de Lactato, que avalia doença aguda, não ser feito pode indicar quadro estável do paciente.

A imputação de valores gerou diferença mínima nas médias e desvios padrões das variáveis.

3.5 Modelagem

Para a modelagem, foi usado o método do scikit-learn *GridSearchCV*, que recebe grupos de vários parâmetros ou grids de um determinado modelo, cada um com opções de valores para serem usados, e faz validação cruzada para cada combinação dos diferentes parâmetros do modelo que será treinado. Ao final ele traz os tempos de treino em cada fold, as médias dos tempos das validações, valor das métricas passadas ao método de cada treino e as médias da validação para cada combinação dos parâmetros. O número de folds escolhidos para a validação cruzada do *GridSearchCV* foi 5. A fim de evitar *data leakage*, os conjuntos não foram passados para o *GridSearchCV*. Foram criados 2 *pipelines*, um que faz imputação e normalização, o outro imputação + normalização + undersampling. Os conjuntos D1 e D2 sem imputação e normalização, foram então passados para cada *pipeline*, com o intuito de achar os melhores hiperparâmetros para cada algoritmo em cada conjunto correspondente. Por exemplo, usar D2 no segundo *pipeline* para gerar achar os hiperparâmetros de do conjunto D4. Foram utilizados dois pipelines porque cada método de modelagem requer um fluxo de pré-processamento distinto: um pipeline realizando apenas imputação e normalização, e outro adicionando a etapa de *undersampling*. Incorporar essas etapas dentro dos pipelines garante que todas as transformações sejam executadas somente sobre os folds de treino dentro da validação cruzada.

Com o intuito de avaliar melhor o algoritmo MLP em relação ao HyperFast, fizemos um grid de parâmetros com redes menores para o MLP e um grid com uma arquitetura similar em relação ao número de parâmetros da rede gerada pelo HyperFast(MLP-Hyper). Para o HyperFast usamos dois grids com os mesmos parâmetros, um para rodar em CPU igual ao algoritmo MLP do scikit-learn e o outro em

GPU. No HyperFast, o parâmetro ‘*optimization*’ tem valor True como padrão. Isso faz com que após a geração da rede principal, ela passe por *fine tuning* com passos definidos pelo parâmetro ‘*optimization steps*’. Por isso, um segundo grid para o HyperFast em cada tipo de processador foi usado com ‘*optimization*’ como None, o que possibilita avaliar a rede principal gerada sem nenhuma modificação posterior.

4. Resultados

Os melhores hiperparâmetros achados pelo *GridSearchCV*, os resultados de AUPRC, junto com o intervalo de confiança de 95% calculado usando dois desvios padrões estão apresentados na tabela A6 no apêndice H. Os melhores parâmetros foram usados, então, para treinar os modelos e depois usados nos conjuntos de teste.

O conjunto de dados de treinamento com melhor resultado, tendo como ênfase a AUPRC e acurácia balanceada (*Balanced Accuracy*), que é equivalente ao Recall Macro, foi o conjunto D4. Os resultados podem ser vistos na Tabela 2. Levando em consideração os resultados dos HyperFast em todos os conjuntos, é possível notar que o HyperFast foi sensível ao processo de remoção dos valores outliers e ao balanceamento das classes no momento de treinar o modelo.

Foi feito também, para cada modelo em cada conjunto, o gráfico da curva Precision Recall, cujo somente do HyperFast com otimização em D4 será mostrado como exemplo (Figura 2).

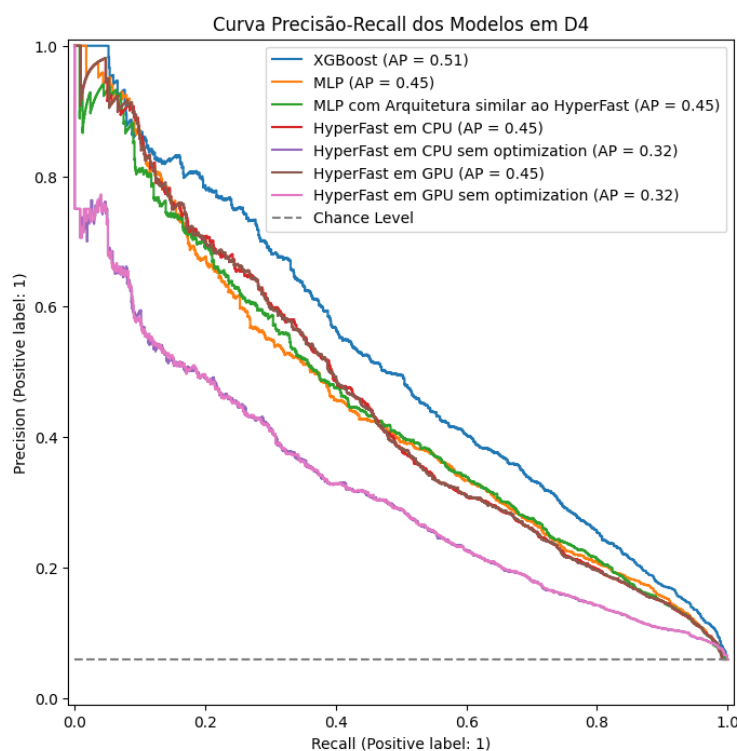


Figura 2 - Gráfico da curva Precisão-Recall para os modelos no conjunto D4

Tabela 2 - Resultados dos modelos no conjunto de teste de D4

Modelo	AUC ROC	AUPRC	Balanced Accuracy	Precision Macro
XGBoost	0,9094 (IC95%: 0,9001 – 0,9176)	0,5134 (IC95%: 0,4811 – 0,5438)	0,8292 (IC95%: 0,8172 – 0,8410)	0,6097 (IC95%: 0,6028 – 0,6168)
MLP	0,8832 (IC95%: 0,8725 – 0,8920)	0,4421 (IC95%: 0,4117 – 0,4718)	0,7972 (IC95%: 0,7847 – 0,8084)	0,5927 (IC95%: 0,5866 – 0,5984)
MLP com arquitetura similar ao HyperFast	0,8860 (IC95%: 0,8752 – 0,8953)	0,4392 (IC95%: 0,4080 – 0,4730)	0,8022 (IC95%: 0,7889 – 0,8151)	0,6063 (IC95%: 0,5992 – 0,6134)
HyperFast em CPU	0,8815 (IC95%: 0,8717 – 0,8910)	0,4496 (IC95%: 0,4196 – 0,4796)	0,7967 (IC95%: 0,7845 – 0,8088)	0,5931 (IC95%: 0,5868 – 0,5992)
HyperFast em CPU sem optimization	0,8368 (IC95%: 0,8254 – 0,8482)	0,3175 (IC95%: 0,2898 – 0,3468)	0,7487 (IC95%: 0,7351 – 0,7618)	0,5694 (IC95%: 0,5642 – 0,5745)
HyperFast em GPU	0,8814 (IC95%: 0,8716 – 0,8910)	0,4492 (IC95%: 0,4193 – 0,4791)	0,7956 (IC95%: 0,7838 – 0,8077)	0,5927 (IC95%: 0,5864 – 0,5988)
HyperFast em GPU sem optimization	0,8368 (IC95%: 0,8254 – 0,8484)	0,3176 (IC95%: 0,2899 – 0,3468)	0,7503 (IC95%: 0,7370 – 0,7631)	0,5696 (IC95%: 0,5645 – 0,5748)

Com base na Tabela 7, vemos que o XGBoost obteve o melhor desempenho em todas as métricas avaliadas, destacando-se com uma AUC ROC de 0,9094 (IC95%: 0,9001 – 0,9176), acurácia balanceada de 0,8292 (IC95%: 0,8172 – 0,8410) e, crucialmente, uma AUPRC de 0,5134 (IC95%: 0,4811 – 0,5438). Os demais modelos baseados em redes neurais — MLP padrão, MLP com arquitetura similar ao HyperFast os HyperFast com optimization (CPU e GPU) — apresentaram desempenho intermediário e relativamente próximo entre si. As AUCs encontraram-se na faixa de 0,881 a 0,886, os valores de AUPRC situaram-se entre 0,439 e 0,449 e as acurácias

balanceada em torno de 0,8. Esses resultados, embora expressivos, foram consistentemente inferiores aos alcançados pelo XGBoost. Os HyperFast sem otimização apresentaram desempenho consideravelmente inferior, com destaque para as reduções marcantes em AUPRC ($\approx 0,318$) e acurácia balanceada ($\approx 0,75$). Esses resultados indicam que o algoritmo depende fortemente de ajustes finos para convergir para uma representação significativa dos dados, e que a ausência dessas otimizações compromete severamente sua capacidade preditiva.

Olhando a Figura 2, observamos que a curva Precisão-Recall do XGBoost se mantém consistentemente acima das demais ao longo de praticamente todo o espectro de *recall*, evidenciando melhor capacidade de identificar verdadeiros positivos sem sacrificar excessivamente a precisão. As curvas dos MLPs e dos HyperFast com optimization ficaram bem parecidas, indicando que tem desempenho preditivos similares.

Outro ponto a ser observado é o valor das precisões que ficaram 0,6. Isso sugere que os modelos, podem ser calibrados para melhorar aumentar a precisão sacrificando um pouco do recall, algo que pode ser observado no gráfico das curvas.

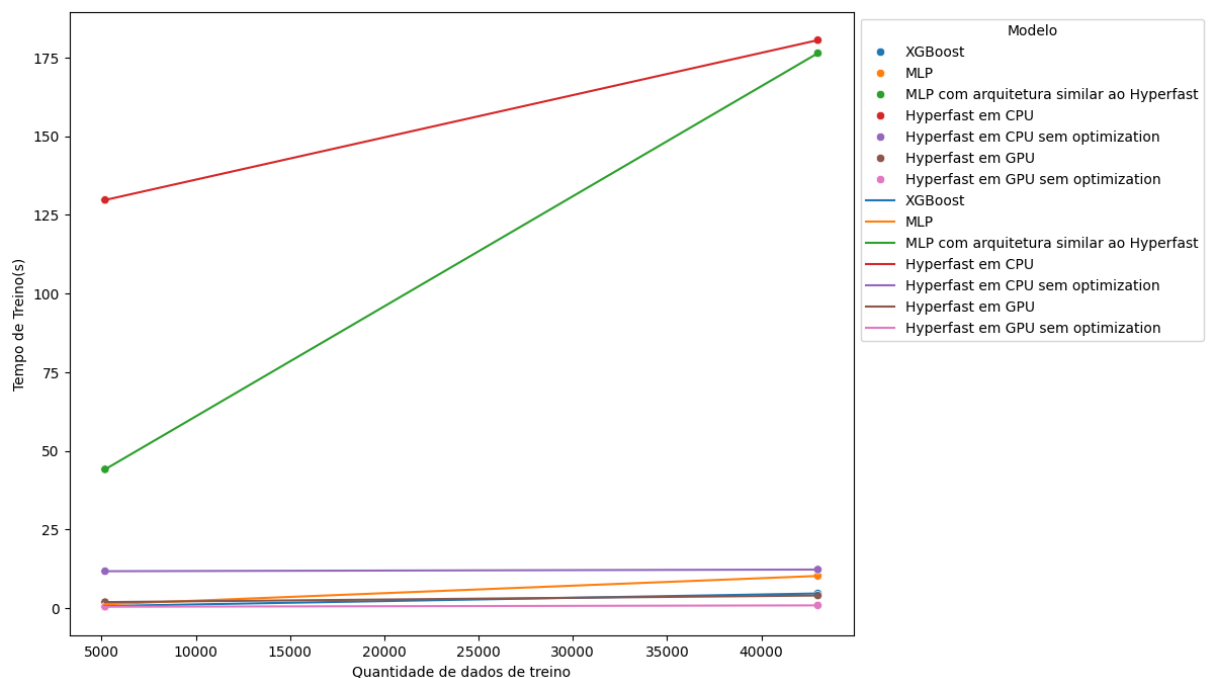


Figura 3 - Gráfico do número de dados vs tempo de Treinamento usando os conjuntos D1 e D3

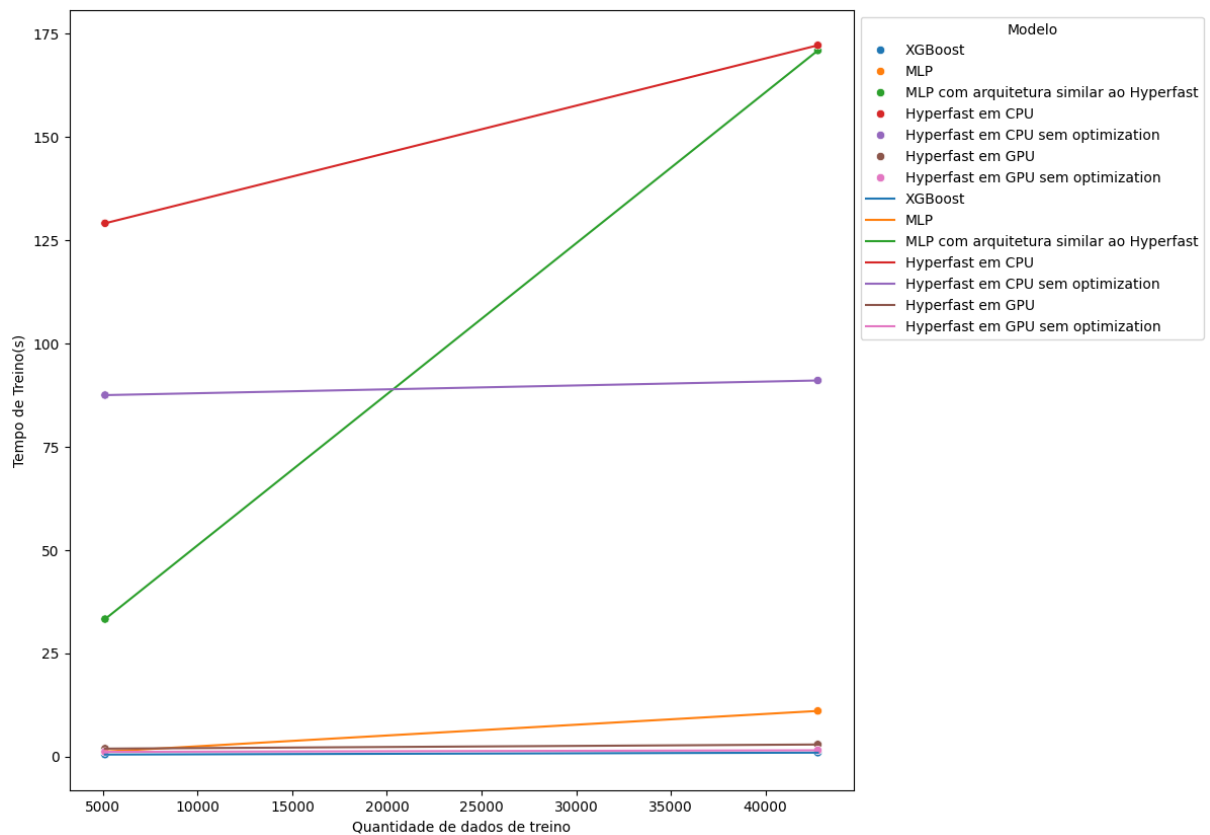


Figura 4 - Gráfico do número de dados vs tempo de Treinamento usando os conjuntos D2 e D4

Em relação aos tempos para treino, como podemos ver pelos gráficos das figuras 3 e 4, o Modelo MPL com arquitetura similar ao HyperFast, cresceu mais em relação aos dados que o HyperFast sendo rodado em CPU. O HyperFast sem otimização não apresentou aumento de tempo. Quando olhamos para o hiperparâmetros do HyperFast com otimização, vemos que no conjunto D1 o parâmetro *optimize_steps* é 16 e 8 em D3. O HyperFast sem otimização fica com tempo de treinamento igual, independentemente da quantidade de dados. O mesmo se repete para D2 e D4, com a diferença que o HyperFast sem otimização usou *batch_size* de 2048 em D2 e D4 mas 1024 em D2 e D1.

De fato, ao pegar as tabelas geradas pela validação cruzada, e fazer o gráfico do tempo de treino nos modelos sem *optimize_steps* (Figura 5) e depois fixando o *batch_size* nos modelos de D1 e D3 com *fine tuning* (Figura 6), podemos observar a quantidade de dados não influência no tempo de treinamento, somente os hiperparâmetros usados.

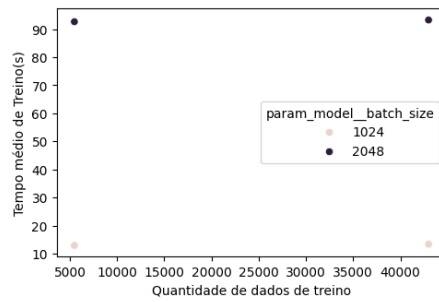
Outro ponto é que o tempo de predição foi maior que os outros modelos no HyperFast rodando em CPU. Observando a Tabela 3, vemos que o HyperFast tem o maior tempo de predição, em torno de 12 a 13 segundos. Isso se deve ao fato de que o modelo sempre precisa fazer o processo de PCA e RF para deixar as entradas com a mesma dimensão da rede principal gerada.

Tabela 3 - Tempo de predição do HyperFast em CPU e Quantidade de dados

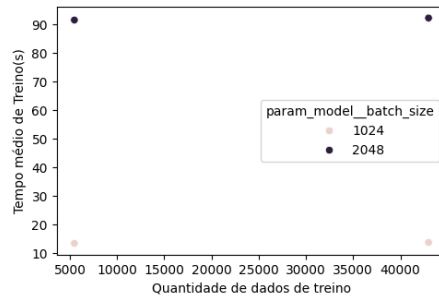
Modelo	Tempo de Predição(s)	Quantidade de dados de test
D1		
XGBoost	0,0449	18429
MLP	0,0321	18429
MLP com arquitetura similar ao HyperFast	0,3990	18429
HyperFast em CPU	12,8867	18429
HyperFast em CPU sem optimization	12,9607	18429
HyperFast em GPU	1,4701	18429
HyperFast em GPU sem optimization	2,3546	18429
D2		
XGBoost	0,0217	18327
MLP	0,0253	18327
MLP com arquitetura similar ao HyperFast	0,3201	18327
HyperFast em CPU	12,6081	18327
HyperFast em CPU sem optimization	12,6515	18327
HyperFast em GPU	1,4630	18327
HyperFast em GPU sem optimization	1,3649	18327
D3		
XGBoost	0,0223	18429
MLP	0,0244	18429
MLP com arquitetura similar ao HyperFast	0,3060	18429

HyperFast em CPU	12,5967	18429
HyperFast em CPU sem optimization	12,9028	18429
HyperFast em GPU	1,4146	18429
HyperFast em GPU sem optimization	2,3227	18429
D4		
XGBoost	0,0233	18327
MLP	0,0246	18327
MLP com arquitetura similar ao HyperFast	0,3321	18327
HyperFast em CPU	12,6556	18327
HyperFast em CPU sem optimization	12,5860	18327
HyperFast em GPU	1,4196	18327
HyperFast em GPU sem optimization	1,3441	18327

Por fim, com base na Tabela 4, observamos que os modelos gerados pelo HyperFast possuem tamanho em memória muito maior que os modelos gerados pelos outros algoritmos. Especificamente em relação a MLP com a arquitetura similar do HyperFast que tem segundo maior tamanho (~28Mb), os modelos HyperFast tem tamanho em torno de 181 vezes maior(~5000Mb) e aproximadamente 10000 vezes maior que o XGBoost, que foi o menor em todos os conjuntos.

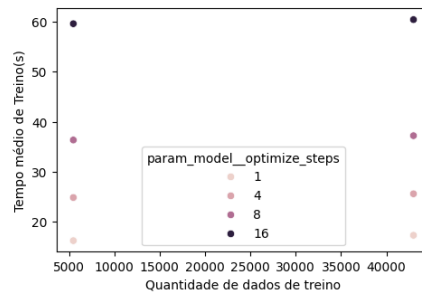


(a) - Tempo médio de treino por quantidade de dados por batch_size usando as quantidades dos conjunto D1 e D3

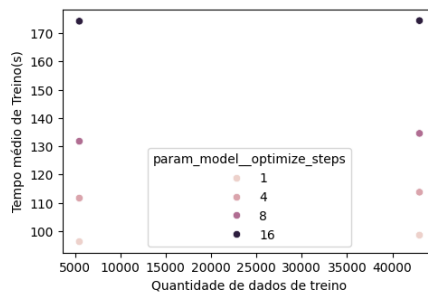


(b) - Tempo médio de treino por quantidade de dados por batch_size usando as quantidades dos conjunto D2 e D4

Figura 5 - Gráficos dos tempos de treino em relação a quantidade de dados dos modelos HyperFast em CPU sem fine tuning



(a) - Tempo médio de treino por quantidade de dados por optimize_step fixando batch_size em 1024



(b) - Tempo médio de treino por quantidade de dados por optimize_step fixando batch_size em 2048

Figura 6 - Gráficos dos tempos de treino em relação a quantidade de dados dos modelos HyperFast em CPU com fine tuning usando as quantidades de dados dos conjuntos D1 e D3

Tabela 4 - Tamanho dos modelos em MegaBytes

Conjunto	Modelo	Tamanho do modelo(megabytes)
D1	XGBoost	0,565
D1	MLP	0,671
D1	MLP com arquitetura similar ao HyperFast	28,440
D1	HyperFast em CPU	5148,404
D1	HyperFast em CPU sem optimization	5019,670
D1	HyperFast em GPU	4985,703
D1	HyperFast em GPU sem optimization	4984,969
D2	XGBoost	0,234
D2	MLP	0,671
D2	MLP com arquitetura similar ao HyperFast	28,440
D2	HyperFast em CPU	5148,404
D2	HyperFast em CPU sem optimization	5148,404
D2	HyperFast em GPU	4985,703
D2	HyperFast em GPU sem optimization	4985,703
D3	XGBoost	0,229
D3	MLP	0,458
D3	MLP com arquitetura similar ao HyperFast	28,441
D3	HyperFast em CPU	5148,405
D3	HyperFast em CPU sem optimization	5019,670
D3	HyperFast em GPU	4985,703

D3	HyperFast em GPU sem optimization	4984,969
D4	XGBoost	0,232
D4	MLP	0,458
D4	MLP com arquitetura similar ao HyperFast	28,440
D4	HyperFast em CPU	5148,404
D4	HyperFast em CPU sem optimization	5148,404
D4	HyperFast em GPU	4985,703
D4	HyperFast em GPU sem optimization	4985,703

5. Conclusões

Este trabalho fez modelos de predição de mortalidade em até sete dias após a primeira entrada na UTI usando dados coletados nas primeiras 24 horas após a entradas na UTI usando os modelos XGBoost, MLP e HyperFast e comparando-os quanto suas performances preditivas, tamanhos e tempos.

Levando em consideração os resultados vistos na seção anterior, conclui-se que o algoritmo que teve melhor foi desempenho foi XGBoost, sendo melhor mão só no conjunto D4, mas também nos outros conjuntos. HyperFast se mostrou mais competitivo com balanceamento e eliminação de outliers sendo comparável aos modelos sendo comparável MPL. O HyperFast sem otimização só foi melhor em relação a acurácia balanceada nos conjuntos onde não foi feito balanceamento, o que pode ser explicado pelo processo de otimização (*fine tuning*) sendo enviesado pela maior proporção da classe positiva.

Pensando no cenário complexo de mortalidade em UTIs e hospitais em geral, que exigem decisões rápidas e que buscam ter um Recall alto, o modelo XGBoost mostrou-se o mais adequado pelo desempenho, tamanho e velocidade de treino e predição. Os resultados de tamanho e velocidade do HyperFast, que teve resultados de performance preditiva razoáveis e comparáveis as redes neurais no melhor cenário, são um fator limitante, por exigir alta memória e a necessidade de ter GPU para fazer predições rápidas, impossibilitando por exemplo a implementação dentro de um dispositivo móvel que possa ser usada pela equipe médica.

5.1 Limitações

Neste trabalho, o número de modelos e hiperparâmetros usados nos grids foi limitado pelo tempo disponível de uso da máquina virtual do Google Colab.

5.2 Trabalhos Futuros

Como trabalhos futuros, podemos avaliar o uso do HyperFast compará-lo com outros algoritmos que não foram usados aqui. Outro trabalho é fazer uma comparação com uma maior quantidade de conjuntos de dados e com quantidade de dados diferentes para ter uma avaliação melhor da influência da quantidade de dados em relação ao *batch size*, considerando a influência que terá na performance geral do modelo. Nesse mesmo contexto avaliar como os outros hiperparâmetros, por exemplo *n_ensemble*, influenciam na performance e no tempo de treinamento e predição.

Outro possível ponto de pesquisa, é avaliar os modelos ao longo do tempo, ou seja, como os modelos que são treinados com dados de um período específico de tempo vão performar com dados de outros períodos.

Por fim, gerar os parâmetros das hiper redes usando conjuntos de dados deste contexto, e avaliar as redes geradas por esse HyperFast, também é uma ideia interessante para um próximo trabalho, porém a atual implementação não permite fazer esse treinamento da hiper rede.

6 Referências

VRANAS, K. C. et al. Identifying Distinct Subgroups of ICU Patients. **Critical Care Medicine**, v. 45, n. 10, p. 1607–1615, out. 2017. DOI: <https://doi.org/10.1097/CCM.0000000000002548>.

CASTELA FORTE, J. et al. Identifying and characterizing high-risk clusters in a heterogeneous ICU population with deep embedded clustering. **Scientific Reports**, v. 11, n. 1, 8 jun. 2021. DOI: <https://doi.org/10.1038/s41598-021-91297-x>.

SAADATMAND, S. et al. Using machine learning in prediction of ICU admission, mortality, and length of stay in the early stage of admission of COVID-19 patients. **Annals of Operations Research**, 29 set. 2022. DOI : <https://doi.org/10.1007/s10479-022-04984-x>.

FENN, A. et al. Development and Validation of Machine Learning Models to Predict Admission From Emergency Department to Inpatient and Intensive Care Units. **Annals of Emergency Medicine**, v. 78, n. 2, p. 290–302, ago. 2021. DOI: <https://doi.org/10.1016/j.annemergmed.2021.02.029>.

HONG, W. S.; HAIMOVICH, A. D.; TAYLOR, R. A. Predicting hospital admission at emergency department triage using machine learning. **PLOS ONE**, v. 13, n. 7, p. e0201016, 20 jul. 2018. DOI: <https://doi.org/10.1371/journal.pone.0201016>.

SUBUDHI, S. et al. Comparing machine learning algorithms for predicting ICU admission and mortality in COVID-19. **npj Digital Medicine**, v. 4, n. 1, 21 maio 2021. DOI: <https://doi.org/10.1038/s41746-021-00456-x>.

WU, J. et al. Predicting Prolonged Length of ICU Stay through Machine Learning. **Diagnostics**, v. 11, n. 12, p. 2242, 1 dez. 2021. DOI: <https://doi.org/10.3390/diagnostics11122242>.

MCWILLIAMS, C. J. et al. Towards a decision support tool for intensive care discharge: machine learning algorithm development using electronic healthcare data from MIMIC-III and Bristol, UK. **BMJ Open**, v. 9, n. 3, p. e025925, 1 mar. 2019. DOI: <https://doi.org/10.1136/bmjopen-2018-025925>.

ZHAI, Q. et al. Using machine learning tools to predict outcomes for emergency department intensive care unit patients. **Scientific Reports**, v. 10, n. 1, dez. 2020. DOI: <https://doi.org/10.1038/s41598-020-77548-3>.

RAITA, Y. et al. Emergency department triage prediction of clinical outcomes using machine learning models. **Critical Care**, v. 23, n. 1, 22 fev. 2019. DOI: <https://doi.org/10.1186/s13054-019-2351-7>

HAYKIN, Simon S. *Neural Networks: A Comprehensive Foundation*. 2. ed. Upper Saddle River: Prentice Hall, 1999.

BURKOV, Andriy. *The Hundred-Page Machine Learning Book*. 1. ed. Québec City: Andriy Burkov, 2019.

CHEN, T.; GUESTRIN, C. XGBoost: a Scalable Tree Boosting System. **KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 785–794, 2016. DOI: <https://doi.org/10.1145/2939672.2939785>.

XGBOOST DEVELOPERS. **XGBoost Documentation — xgboost 1.5.1 documentation**. Disponível em: <<https://xgboost.readthedocs.io/en/stable/>>.

BONET, D. et al. HyperFast: Instant Classification for Tabular Data. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 38, n. 10, p. 11114–11123, 24 mar. 2024. DOI: <https://doi.org/10.48550/arXiv.2402.14335>.

KNAUER, Ricardo; RODNER, Erik. Squeezing Lemons with Hammers: An Evaluation of AutoML and Tabular Deep Learning for Data-Scarce Classification Applications. **arXiv preprint arXiv:2405.07662**, 2024. DOI: <https://doi.org/10.48550/arXiv.2405.07662>

LEE, Chongmin; KIM, Jihie. Biomarker based Cancer Classification using an Ensemble with Pre-trained Models. **arXiv preprint arXiv:2406.10087**, 2024. DOI: <https://doi.org/10.48550/arXiv.2406.10087>

HA, David; DAI, Andrew; LE, Quoc V. Hypernetworks. **arXiv preprint arXiv:1609.09106**, 2016. DOI: <https://doi.org/10.48550/arXiv.1609.09106>

CABALLAR, R.; STRYKER, C. **Meta learning**. Disponível em: <https://www.ibm.com/think/topics/meta-learning>.

AI-SANDBOX. **GitHub - AI-sandbox/HyperFast**. Disponível em: <https://github.com/AI-sandbox/HyperFast>. Acesso em: 4 dez. 2024.

BREIMAN, L. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001. DOI: <https://doi.org/10.1023/A:1010933404324>

SCIKIT-LEARN. **scikit-learn: machine learning in Python**. Disponível em: <https://scikit-learn.org/stable/>.

TAYLOR, R. A. et al. Prediction of In-hospital Mortality in Emergency Department Patients With Sepsis: A Local Big Data-Driven, Machine Learning Approach. **Academic Emergency Medicine**, v. 23, n. 3, p. 269–278, 13 fev. 2016.

SCHMIDT, Diogo; DA SILVA, Denise Bandeira; DA COSTA, Cristiano André; RIGHI, Rodrigo da Rosa. Um Modelo de Predição de Mortalidade em Unidades de Terapia Intensiva Baseado em Deep Learning. *In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO APLICADA À SAÚDE (SBCAS)*, 18., 2018, Natal. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2018. ISSN 2763-8952. DOI: <https://doi.org/10.5753/sbcas.2018.3677>.

CHOI, M. H. et al. Mortality prediction of patients in intensive care units using machine learning algorithms based on electronic health records. **Scientific Reports**, v. 12, n. 1, 3 maio 2022. DOI: <https://doi.org/10.1038/s41598-022-11226-4>.

KEEGAN, Mark T.; SOARES, Marcio. What every intensivist should know about prognostic scoring systems and risk-adjusted mortality. *Revista Brasileira de Terapia Intensiva*, v. 28, n. 3, p. 264-269, Jan. 2016.

VEITH, N.; STEELE, R. Machine Learning-based Prediction of ICU Patient Mortality at Time of Admission. **Proceedings of the 2nd International Conference on Information System and Data Mining - ICISDM '18**, 2018.

NEMATİ, S. et al. An Interpretable Machine Learning Model for Accurate Prediction of Sepsis in the ICU. **Critical Care Medicine**, v. 46, n. 4, p. 547–553, 1 abr. 2018.

CHIA, A. H. T. et al. Explainable machine learning prediction of ICU mortality. **Informatics in Medicine Unlocked**, v. 25, p. 100674, 2021. DOI: <https://doi.org/10.1016/j.imu.2021.100674>.

MARTIN-LOECHES, Ignacio *et al.* Determinants of time to death in hospital in critically ill patients around the world. **Intensive Care Medicine**, v. 42, n. 9, p. 1454-1460, set. 2016. DOI: 10.1007/s00134-016-4479-0. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/27518322/>.

CHAPMAN, Pete et al. CRISP-DM 1.0. **CRISP-DM Consortium**, v. 76, n. 3, 2000.

Classificação: precisão, recall, precisão e métricas relacionadas. Disponível em: <<https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=pt-br>>. Acesso em: 22 nov. 2024.

JOHNSON, A. E. W. et al. MIMIC-IV, a freely accessible electronic health record dataset. **Scientific Data**, v. 10, n. 1, 3 jan. 2023.

JOHNSON, A. et al. **MIMIC-IV**. 2024. Disponível em: <<https://physionet.org/content/mimiciv/3.1/>>. Acesso em: 25 nov. 2024.

SAITO, T.; REHMSMEIER, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. **PLOS ONE**, v. 10, n. 3, p. e0118432, 4 mar. 2015. DOI: <https://doi.org/10.1371>

BERTRAND BOUVAREL; FABRICE CARRAT; LAPIDUS, N. Updating mortality risk estimation in intensive care units from high-dimensional electronic health records with incomplete data. **BMC Medical Informatics and Decision Making**, v. 23, n. 1, 30 ago. 2023.