UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

MARCO CESAR DA SILVA

Modelo preditivo dos anos de vida perdidos por morte prematura para os municípios brasileiros de médio porte utilizando aprendizagem de máquina

FLORIANÓPOLIS
2024

MARCO CESAR DA SILVA

Modelo preditivo dos anos de vida perdidos por morte prematura para os municípios brasileiros de médio porte utilizando aprendizagem de máquina

Trabalho de Conclusão de Curso submetido ao curso de Sistemas de Informação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Vinicius Faria Culmant Ramos, Dr.
Coorientador: Prof. José Eduardo De Lucca, Me.

FLORIANÓPOLIS
2024

Silva, Marco Cesar da
Modelo preditivo dos anos de vida perdidos por morte prematura para os municípios brasileiros de médio porte utilizando aprendizagem de máquina / Marco Cesar da Silva ; orientador, Vinícius Faria Culmant Ramos, coorientador, José Eduardo De Lucca, 2024.
121 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Sistemas de Informação, Florianópolis, 2024.

Inclui referências.

1. Sistemas de Informação. 2. Séries Temporais. 3. Machine Learning. 4. Ciência de Dados. 5. Modelo Preditivo. I. Ramos, Vinícius Faria Culmant. II. De Lucca, José Eduardo. III. Universidade Federal de Santa Catarina. Graduação em Sistemas de Informação. IV. Título.

MARCO CESAR DA SILVA

Modelo preditivo dos anos de vida perdidos por morte prematura para os municípios brasileiros de médio porte utilizando aprendizagem de máquina

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel em Sistemas de Informação e aprovado em sua forma final pelo Curso de Sistemas de Informação.

Florianópolis, 18 de dezembro de 2024.

Prof. Dr. Renato Cislaghi
Coordenador

Banca Examinadora:

Prof. Vinicius Faria Culmant Ramos, Dr.
Orientador

Prof. José Eduardo De Lucca, Me.
Coorientador

Emanuel Marques Queiroga, Dr.

Luís Antonio Lourenço, Dr.

RESUMO

O indicador de Anos de Vida Perdidos por Morte Prematura (YLL, na sigla em inglês) é uma métrica essencial em saúde pública, utilizada para avaliar o impacto de doenças e fatores de risco na mortalidade precoce. No entanto, o cálculo do YLL depende de dados atualizados de mortalidade por localidade, os quais apresentam defasagens em sua disponibilização. Além disso, esses dados podem sofrer distorções significativas, como as observadas durante a pandemia de COVID-19. Diante desse contexto, este trabalho tem como objetivo avaliar modelos preditivos para estimar os anos de vida perdidos por morte prematura em municípios brasileiros de médio porte, empregando técnicas de aprendizagem de máquina. Para alcançar esse objetivo, utilizam-se dados de mortalidade fornecidos pelo Departamento de Informática do Sistema Único de Saúde (DATASUS), com foco na predição do YLL tanto para o período atual (*nowcasting*) quanto para períodos futuros (*forecasting*). A metodologia empregada seguiu o framework CRISP-DM (*Cross Industry Standard Process for Data Mining*) e iniciou com o entendimento da questão de negócio, embasado em uma revisão bibliográfica. Os dados foram extraídos, analisados e explorados para obter uma compreensão detalhada de suas características. Em seguida, foi desenvolvido um pipeline automatizado, visando garantir eficiência no processamento e qualidade dos dados. Na etapa de modelagem, foram aplicados e avaliados os modelos ARIMA, SARIMA, XGBoost, Prophet e LSTM. Os resultados obtidos demonstraram a capacidade do modelo LSTM em fornecer as estimativas mais confiáveis do YLL tanto para o presente quanto para períodos subsequentes. Esses resultados têm o potencial de apoiar gestores públicos na formulação de políticas de saúde mais eficazes, fundamentadas em dados e previsões robustas.

Palavras-chaves: Dados; YLL; Machine Learning; Modelagem Preditiva; Séries Temporais.

ABSTRACT

The Years of Life Lost due to Premature Mortality (YLL) indicator is an essential metric in public health, used to assess the impact of diseases and risk factors on early mortality. However, the calculation of YLL relies on updated mortality data by location, which often faces delays in availability. Furthermore, this data can suffer significant distortions, such as those observed during the COVID-19 pandemic. Given this context, this study aims to evaluate predictive models to estimate years of life lost due to premature death in medium-sized Brazilian municipalities, employing machine learning techniques. To achieve this objective, mortality data provided by the Department of Informatics of the Unified Health System (DATASUS) are used, focusing on YLL prediction for both the current period (nowcasting) and future periods (forecasting). The methodology employed followed the CRISP-DM (Cross Industry Standard Process for Data Mining) framework, beginning with an understanding of the business problem, supported by a literature review. The data were extracted, analyzed, and explored to obtain a detailed understanding of their characteristics. Subsequently, an automated pipeline was developed to ensure efficient processing and data quality. In the modeling stage, ARIMA, SARIMA, XGBoost, Prophet, and LSTM models were applied and evaluated. The results demonstrated the LSTM model's ability to provide the most reliable YLL estimates for both the present and subsequent periods. These findings have the potential to support public managers in formulating more effective health policies based on robust data and predictions.

Keywords: Data; YLL; Machine Learning; Predictive Modeling; Time Series.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O propósito deste capítulo reside na apresentação do contexto no qual o conteúdo deste trabalho se insere. Nesse sentido, procede-se à contextualização do tema, à definição do problema de pesquisa, os objetivos e, subsequentemente, à justificativa que, por sua vez, destaca sobretudo a importância, a relevância e a viabilidade da presente investigação.

1.1 CONTEXTUALIZAÇÃO DO TEMA

Os anos de vida perdidos por morte prematura ou YLL (sigla em inglês para *Years of Life Lost*) têm sido um indicador essencial para avaliar a carga da doença e a saúde da população. Essa métrica estima a perda de tempo de vida saudável devido a óbitos ocorridos antes da expectativa de vida estabelecida. O YLL foi utilizado no Estudo de Carga Global de Doença, desenvolvido conjuntamente pela Organização Mundial de Saúde (OMS) e pelo Banco Mundial (BM). Ele é calculado, utilizando-se a expectativa de vida de um indivíduo calculada pelo *World Population Prospects 2019* da ONU (ORGANIZAÇÃO DAS NAÇÕES UNIDAS).

No cálculo do YLL, utiliza-se uma tabela de expectativa de vida padronizada para ponderar cada óbito de acordo com idade e patologia específica. Essa padronização visa garantir que todos os óbitos contribuam igualmente na estimativa da carga total (COSTA, 2007). O indicador YLL diferencia-se dos indicadores tradicionais, pois não considera apenas o número de mortes, mas também os anos de vida perdidos.

No contexto brasileiro, compreender e prever o YLL em nível municipal é de suma importância para o planejamento e implementação de políticas de saúde eficazes, visando à redução dessas perdas e à promoção de uma vida mais saudável para a população. Para o cálculo do YLL, são utilizados os dados de mortalidade, que no Brasil são disponibilizados pelo Ministério da Saúde através do DATASUS, que se trata do Departamento de Informática do Sistema Único de Saúde, que por sua vez gerencia e disponibiliza os dados através do SIM (Sistema de Informação sobre Mortalidade). No entanto, estes dados não são disponibilizados em tempo real, já que há um fluxo longo e burocrático, estabelecido pela Portaria SVS nº 116, de 11 de fevereiro de 2009 (MINISTÉRIO DA SAÚDE, 2009), que estima um prazo de até

dois anos para a disponibilização dos dados de forma definitiva e validada, promovendo uma defasagem nas informações para os tomadores de decisões e agentes de saúde.

Paralelo a isso, os algoritmos de *machine learning* vêm se consolidando como ferramentas essenciais em pesquisas, devido à sua capacidade de lidar com conjuntos de dados complexos e identificar padrões não triviais. Silva et al. (2020) discutem o *trade-off* entre viés e variância no desempenho preditivo do *machine learning* (ML), destacando que a redução das incertezas das predições pode ser alcançada à custa de um maior controle do viés dos estimadores. Além disso, o ML, como uma técnica avançada de análise de dados, supera métodos estatísticos tradicionais ao oferecer maior precisão em bases de dados amplas e heterogêneas, além de proporcionar métodos mais sofisticados de avaliação e seleção de modelos.

No contexto da análise preditiva, amplamente utilizada na saúde pública, algoritmos como Random Forest e Gradient Boosting Machines têm demonstrado alta eficácia ao processar informações complexas e variadas, possibilitando decisões mais rápidas e assertivas (RAJKOMAR; DEAN; KOHANE, 2018). Contudo, a precisão das estimativas depende da capacidade do modelo de mitigar fontes de erro, como viés e variância. Silva et al. (2020) apontam que, apesar dos avanços oferecidos pelo ML, erros inevitáveis, como aqueles causados pela ausência de variáveis relevantes ou fatores externos não controláveis, permanecem uma limitação inerente às projeções realizadas.

Desta forma, o desenvolvimento de um modelo para a predição do YLL nos municípios brasileiros utilizando *machine learning* pode gerar uma importante informação para a tomada de decisões estratégicas e o planejamento de ações de saúde mais eficazes. A previsão do YLL permitirá a identificação de grupos de risco, áreas geográficas com maior necessidade de intervenção e fatores de risco específicos que podem ser alvo de programas de prevenção e controle.

Além disso, a utilização de técnicas avançadas de análise de dados e aprendizado de máquina permitirá explorar padrões complexos e identificar variáveis-chave que influenciam o YLL. Dessa forma, será possível fornecer subsídios para a formulação de políticas públicas mais direcionadas, promovendo a redução das perdas de anos de vida por morte prematura e melhorando a saúde da população brasileira como um todo.

1.2 PROBLEMA DE PESQUISA

O problema de pesquisa pode ser resumido na seguinte pergunta: **Como prever da forma mais acurada os índices atuais e futuros de anos de vida perdidos por morte prematura para os municípios brasileiros de médio porte?**

1.3 OBJETIVOS

Com o intuito de estabelecer um escopo definido para o presente estudo, a seguir, são delineados o objetivo geral proposto para a pesquisa e os objetivos específicos que permeiam sua consecução.

1.3.1 Objetivo Geral

Avaliar modelos preditivos dos anos de vida perdidos por morte prematura para os municípios brasileiros de médio porte, utilizando aprendizagem de máquina.

1.3.2 Objetivos Específicos

De forma a atingir o objetivo geral, o presente projeto tem os seguintes objetivos específicos:

a) Identificar as características relevantes para o cálculo do YLL;

b) Identificar as bases de dados disponíveis e relevantes para o objetivo proposto e obter uma compreensão dos dados;

c) Desenvolver um pipeline automatizado de extração, transformação e carga de dados;

d) Identificar os algoritmos de aprendizagem de máquina para realização de predição do YLL;

e) Avaliar o modelo preditivo para determinar sua eficácia na predição de YLL dos municípios brasileiros de médio porte.

1.4 JUSTIFICATIVA

Diante desses objetivos, o desenvolvimento deste modelo é fundamentado na importância de compreender e quantificar o impacto das mortes prematuras na sociedade, além de fornecer subsídios para a tomada de decisão dos gestores de saúde por meio do processo de *nowcasting*, isso é, prever os indicadores presentes, já que existe uma defasagem na disponibilização dos dados atuais do YLL.

A análise dos Anos de Vida Perdidos por Morte Prematura é de extrema relevância para a sociedade como um todo. Essa métrica representa a medida do impacto causado pela morte de indivíduos em idades consideradas prematuras, ou seja, antes do esperado para a idade média da população. Ao quantificar a perda de anos de vida, é possível compreender a magnitude das perdas e direcionar recursos e políticas públicas de forma mais eficiente para prevenir doenças e promover a saúde.

No contexto acadêmico, a presente pesquisa contribui para o avanço do conhecimento científico em diferentes áreas, como a epidemiologia, a saúde pública e a inteligência artificial. A aplicação de técnicas de aprendizagem de máquina na previsão dos Anos de Vida Perdidos por Morte Prematura permite explorar os benefícios da utilização de modelos preditivos, que podem capturar padrões complexos e gerar insights valiosos para o planejamento e a gestão da saúde.

Destaca-se ainda a viabilidade dessa pesquisa em razão do interesse da comunidade médica em compreender e reduzir as mortes prematuras. Os gestores de saúde enfrentam desafios constantes na alocação de recursos e no desenvolvimento de estratégias para a prevenção e controle de doenças. Ao fornecer um modelo preditivo que estima os Anos de Vida Perdidos por Morte Prematura, é possível auxiliar esses profissionais na identificação de áreas

geográficas ou grupos populacionais com maior vulnerabilidade e, assim, direcionar esforços para ações preventivas mais efetivas.

A acessibilidade aos dados é outro fator que reforça a relevância desse estudo. Com o avanço das tecnologias e a disponibilidade de bases de dados amplas e atualizadas, é possível reunir informações detalhadas sobre óbitos e características socioeconômicas dos municípios brasileiros. Essa facilidade de acesso aos dados permite a construção de modelos preditivos robustos, com potencial de fornecer informações valiosas para a tomada de decisão, tanto no âmbito local como no nível nacional.

Em suma, o desenvolvimento de um modelo preditivo dos Anos de Vida Perdidos por Morte Prematura para os municípios brasileiros, utilizando aprendizagem de máquina e o processo de *nowcasting*, traz inúmeras contribuições para a sociedade, o meio acadêmico, a medicina e a tomada de decisão em saúde. Esse modelo pode fornecer informações atualizadas e precisas para os gestores de saúde, permitindo a adoção de medidas preventivas e corretivas direcionadas, com o objetivo de reduzir a mortalidade prematura e melhorar a qualidade de vida da população.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica deste trabalho tem como objetivo fornecer o embasamento necessário para o desenvolvimento de um modelo preditivo dos Anos de Vida Perdidos por morte prematura (YLL) em municípios brasileiros de médio porte, utilizando técnicas de aprendizado de máquina. Para isso, serão abordados os conceitos fundamentais sobre YLL, discutindo sua relevância na avaliação de impactos de políticas públicas na saúde. Em seguida, serão exploradas as aplicações de m*achine learning* em previsões de saúde, com foco nas potencialidades dessas tecnologias no contexto de predição de dados de saúde. Também será realizada uma análise de séries temporais, uma vez que a predição de YLL envolve dados distribuídos ao longo do tempo. Por fim, serão discutidos os métodos de validação e avaliação de modelos preditivos em saúde, essenciais para garantir a precisão e a eficácia das predições geradas.

2.1 ANOS DE VIDA PERDIDOS POR MORTE PREMATURA (YLL)

Os Anos de Vida Perdidos por Morte Prematura (YLL, do inglês Years of Life Lost) são uma medida essencial no campo da saúde pública, criada para quantificar o impacto da mortalidade precoce na população. Esse indicador foi desenvolvido pelo Institute for Health Metrics and Evaluation (IHME), em colaboração com a Organização Mundial da Saúde (OMS), como parte do Estudo da Carga Global de Doenças (*Global Burden of Disease - GBD*) (IHME, 2024). A intenção era fornecer uma base para comparar as consequências da mortalidade e morbidade em diferentes regiões e ao longo do tempo, permitindo uma análise consistente e baseada em evidências de desafios globais de saúde.

O YLL é definido como o número de anos potenciais de vida perdidos devido a mortes prematuras, ou seja, aquelas que ocorrem antes da expectativa de vida ideal para uma determinada população. Esse método padronizado permite comparações entre diferentes populações, independentemente de variações demográficas ou socioeconômicas (IHME, 2024).

Historicamente, o conceito de YLL foi consolidado no âmbito do estudo Global Burden of Disease 2010 (GBD 2010), coordenado pelo IHME, com contribuições de diversas instituições como a Universidade de Queensland e Harvard School of Public Health (MURRAY

et al., 2021). Essa iniciativa representou uma evolução significativa ao estabelecer uma metodologia padronizada para avaliar perdas de saúde em escala global, permitindo comparações robustas entre doenças, regiões e populações ao longo do tempo (MURRAY et al., 2021). No GBD 2010, métodos avançados de inferência estatística foram incorporados para estimar prevalências de doenças e sequelas, além de incluir uma lista hierárquica de 291 doenças e 1.160 sequelas, garantindo maior precisão e replicabilidade dos resultados (MURRAY et al., 2021).

A principal finalidade do YLL é fornecer uma medida objetiva do impacto da mortalidade precoce em nível populacional. Isso é essencial para priorizar ações em saúde pública, direcionar recursos e avaliar a eficácia de políticas de prevenção e tratamento. Além disso, o YLL é uma das componentes principais do indicador mais amplo chamado “Anos de Vida Ajustados por Incapacidade” (DALYs, do inglês Disability-Adjusted Life Years), que combina mortalidade e morbidade para fornecer uma visão holística da carga de doenças (IHME, 2024).

A importância do YLL está em sua capacidade de destacar os desafios de saúde mais significativos em diferentes regiões. Por exemplo, as taxas de YLL devido a doenças cardiovasculares podem ser comparadas às taxas relacionadas a causas externas, como acidentes de trânsito, permitindo identificar prioridades de intervenção. Além disso, é uma ferramenta valiosa para monitorar o impacto de mudanças epidemiológicas e socioeconômicas ao longo do tempo, como o envelhecimento populacional e a transição de doenças infecciosas para não transmissíveis (IHME, 2024).

O YLL tem sido fundamental para revelar disparidades em saúde global. Estudos baseados nesse indicador mostraram, por exemplo, o impacto desproporcional da pandemia de COVID-19 em regiões como a América Latina e o sul da África, onde os sistemas de saúde foram particularmente sobrecarregados. Da mesma forma, o YLL tem ajudado a evidenciar os efeitos de fatores de risco como hipertensão, diabetes e obesidade na mortalidade precoce (IHME, 2024). Também é uma ferramenta crítica para evidenciar lacunas em regiões com menor acesso a cuidados médicos ou maior prevalência de fatores de risco evitáveis, permitindo uma melhor alocação de recursos e políticas mais equitativas (MURRAY et al., 2021).

Outro aspecto relevante é sua contribuição para as projeções de saúde. Através da utilização de modelos preditivos, é possível estimar como mudanças em políticas de saúde, avanços tecnológicos ou intervenções específicas podem afetar o YLL no futuro, permitindo tomadas de decisão mais informadas e eficientes (IHME, 2024). As revisões metodológicas realizadas no GBD 2010 também visaram simplificar os cálculos de indicadores relacionados,

como os DALYs, promovendo maior acessibilidade para gestores de saúde e pesquisadores (MURRAY et al., 2021).

Em resumo, o YLL é um indicador poderoso e amplamente utilizado que fornece insights profundos sobre as condições de saúde em nível populacional. Sua flexibilidade e relevância têm sido fundamentais para avanços na saúde pública global e no planejamento de intervenções que visam reduzir a mortalidade precoce e melhorar a qualidade de vida (IHME, 2024; MURRAY et al., 2021).

2.2 MACHINE LEARNING EM PREVISÕES DE SAÚDE

Nos últimos anos, o uso de técnicas de *Machine Learning* (ML) tem crescido substancialmente no campo da saúde, principalmente devido à sua capacidade de lidar com grandes volumes de dados e identificar padrões complexos que não são facilmente capturados por métodos estatísticos tradicionais (SHICKEL et al., 2017). Essas técnicas permitem a construção de modelos preditivos que podem auxiliar na previsão de eventos críticos, como a mortalidade, e apoiar decisões clínicas e políticas de saúde pública com maior precisão e velocidade.

Uma das principais razões para o sucesso de *Machine Learning* na saúde é a diversidade de algoritmos disponíveis, que podem ser adaptados a diferentes tipos de problemas e conjuntos de dados. Técnicas supervisionadas, como as baseadas em árvores de decisão, redes neurais artificiais e regressão logística, são amplamente utilizadas para prever resultados clínicos a partir de variáveis demográficas e fatores de risco. Por exemplo, algoritmos como Random Forest e Gradient Boosting Machines têm se mostrado eficazes na predição de mortalidade hospitalar, ao combinarem várias árvores de decisão para gerar previsões robustas e interpretáveis (RAJKOMAR; DEAN; KOHANE, 2018).

A aplicação de *Machine Learning* na saúde pode ser dividida em várias categorias, desde a análise preditiva de dados clínicos até a identificação de padrões em grandes bases de dados epidemiológicos. Para previsões de saúde, especificamente no contexto de mortalidade e expectativa de vida, modelos preditivos têm sido utilizados para prever a probabilidade de morte prematura com base em variáveis socioeconômicas e clínicas, muitas vezes coletadas em bases de dados populacionais. Um exemplo é o uso de redes neurais para prever a mortalidade em pacientes internados em unidades de terapia intensiva (FUTOMA; MORRIS; LUCAS,

2017), que mostrou ser mais eficaz do que modelos tradicionais, como os baseados em regressão linear.

Diferentes tipos de algoritmos de *Machine Learning* são usados em previsões de saúde, cada um com suas características e aplicabilidades. Algoritmos supervisionados, como *Support Vector Machines* (SVM) e *Random Forests*, são úteis quando há um conjunto de dados rotulado disponível, onde o objetivo é prever uma variável de saída com base em um conjunto de características de entrada (CORTES; VAPNIK, 1995). Esses métodos têm sido amplamente utilizados na predição de mortalidade, onde a variável alvo pode ser a ocorrência ou não de morte em determinado período, com base em fatores como idade, sexo, condição clínica, entre outros. Modelos de regressão logística também são frequentemente aplicados para prever mortalidade em função de variáveis contínuas e categóricas, fornecendo uma interpretação direta das probabilidades de risco associadas a cada fator.

Além dos algoritmos supervisionados, técnicas de aprendizado não supervisionado, como *clustering* ou análise de agrupamentos, também têm relevância no campo da saúde. Esses métodos podem ser utilizados para identificar subgrupos de pacientes com características semelhantes ou padrões ocultos em dados de mortalidade, facilitando a criação de intervenções mais direcionadas e personalizadas (ESTEVA et al., 2019).

Outro avanço importante é a aplicação de modelos de aprendizado profundo (*Deep Learning*), que, por meio de redes neurais profundas, conseguem processar grandes volumes de dados heterogêneos, como imagens médicas, sinais vitais e textos clínicos, fornecendo predições mais precisas e muitas vezes superando modelos convencionais. Por exemplo, redes neurais convolucionais têm sido utilizadas para analisar imagens radiológicas, auxiliando na detecção precoce de condições fatais e, assim, impactando as taxas de mortalidade (LITJENS et al., 2017).

A utilização de técnicas de *Machine Learning* no contexto de predição de mortalidade tem mostrado um impacto significativo na capacidade de prever não apenas a mortalidade individual, mas também o impacto populacional, como a predição de anos de vida perdidos (YLL). Isso pode ser crucial para a gestão de recursos e planejamento de políticas públicas de saúde, permitindo identificar áreas de maior vulnerabilidade e criar estratégias de intervenção mais eficazes (OBERMEYER; EMANUEL, 2016).

Em suma, as técnicas de *Machine Learning* oferecem uma abordagem eficaz e flexível para a predição de mortalidade e outros desfechos clínicos. No entanto, a implementação bem-sucedida desses modelos requer não apenas a escolha adequada do algoritmo, mas também uma compreensão profunda dos dados e dos fatores que influenciam

os resultados de saúde. A combinação dessas técnicas com dados populacionais precisos pode melhorar significativamente a capacidade de prever e prevenir mortes prematuras, particularmente em municípios brasileiros, onde a diversidade socioeconômica e de infraestrutura de saúde apresenta desafios únicos.

2.3 ANÁLISE DE SÉRIES TEMPORAIS

Séries temporais são conjuntos de dados ordenados cronologicamente, em que as observações são registradas em intervalos de tempo específicos, como dias, meses ou anos. A principal característica desse tipo de dado é a dependência temporal, ou seja, o fato de que observações passadas influenciam as futuras (MONTGOMERY; PECK; VINING, 2015). Esse tipo de análise permite capturar padrões, tendências, sazonalidades e variações cíclicas que ocorrem ao longo do tempo (SHUMWAY; STOFFER, 2017). No contexto da saúde pública, as séries temporais são frequentemente utilizadas para monitorar a incidência de doenças, prever surtos e analisar indicadores como os anos de vida perdidos (YLL) devido à morte prematura.

A análise de séries temporais tem sido uma ferramenta essencial para predições na área da saúde, especialmente quando se trata de fenômenos que evoluem ao longo do tempo. As séries temporais permitem aos pesquisadores identificar comportamentos recorrentes e mudanças estruturais nos dados, essenciais para previsões acuradas (BOX; JENKINS; REINSEL, 2015). No campo da saúde pública, essa metodologia auxilia no entendimento das tendências de mortalidade, incidência de doenças e na avaliação do impacto de políticas de saúde ao longo do tempo (BHASKARAN et al., 2013). Ao analisar as flutuações anuais ou mensais nas taxas de mortalidade, por exemplo, é possível antecipar mudanças e adotar medidas preventivas ou corretivas com base em padrões observados no passado.

No contexto específico da predição dos anos de vida perdidos (YLL), o uso de séries temporais oferece uma perspectiva crítica para prever as mudanças nas taxas de mortalidade e o impacto de fatores como epidemias, políticas de saúde e mudanças demográficas. As séries temporais permitem identificar não apenas as tendências gerais, mas também a sazonalidade e a ciclicidade, como o aumento de mortes por doenças respiratórias durante os meses mais frios ou em períodos de epidemias.

A modelagem de séries temporais envolve a identificação e a estimação de modelos que melhor se ajustam aos dados. O processo geralmente começa com a análise exploratória, que inclui a verificação de tendências, sazonalidades e a identificação de possíveis pontos de ruptura (BOX; JENKINS; REINSEL, 2015). Entre os métodos mais conhecidos para a análise de séries temporais está o modelo ARIMA (*autoregressive integrated moving average*), amplamente utilizado para ajustar dados com padrões de tendência, sazonalidade e ruído. No entanto, em séries que apresentam ciclos regulares mais explícitos, como sazonalidade anual ou mensal, o modelo SARIMA (*seasonal autoregressive integrated moving average*) é uma alternativa eficaz. Diferentemente do ARIMA, o SARIMA incorpora parâmetros sazonais, o que melhora sua capacidade de capturar variações periódicas nos dados de saúde pública (CHECHI; BAYER, 2012).

Além dos métodos tradicionais, técnicas baseadas em aprendizado de máquina, como o XGBoost (Extreme Gradient Boosting), também têm ganhado espaço na análise de séries temporais. Desenvolvido por Tianqi Chen em 2016, o XGBoost é um modelo baseado em árvores de decisão, conhecido por sua alta eficiência e capacidade de lidar com grandes volumes de dados. Embora não tenha sido projetado especificamente para séries temporais, ele pode ser adaptado para esta finalidade ao transformar os dados em um formato supervisionado. Sua robustez e precisão o tornam adequado para predições em cenários complexos, mas sua implementação requer cuidado para evitar o superajuste (PALIARI; KARANIKOLA; KOTSIANTIS, 2021).

Outro modelo de destaque é o Prophet, desenvolvido pelo time de pesquisa do Facebook em 2017. Este modelo é especialmente útil para séries temporais com lacunas de dados ou mudanças abruptas nos padrões, como sazonalidade e feriados. Sua principal vantagem é a simplicidade de configuração e interpretação, o que facilita seu uso em diferentes aplicações. No entanto, o Prophet pode ter limitações em séries temporais muito complexas, onde métodos mais avançados, como redes neurais, podem ser mais eficazes (KASIEMKHAN, 2023).

Entre as abordagens baseadas em redes neurais, as LSTM (*Long Short-Term Memory Networks*) têm se mostrado uma das mais promissoras para a análise de séries temporais. Propostas por Hochreiter e Schmidhuber em 1997, as LSTMs são capazes de capturar dependências de longo prazo nos dados, superando limitações das redes recorrentes tradicionais. No campo da saúde pública, têm sido amplamente utilizadas para prever taxas de mortalidade e padrões de doenças, especialmente quando os dados apresentam não linearidades complexas. Contudo, sua alta demanda computacional e necessidade de grandes volumes de

dados podem limitar seu uso em cenários de recursos restritos (PALIARI; KARANIKOLA; KOTSIANTIS, 2021).

Quando aplicadas à predição de YLL, essas metodologias são fundamentais para identificar padrões temporais e prever o impacto futuro de variáveis associadas a mortes prematuras. Por exemplo, ao utilizar séries temporais de YLL para diferentes municípios brasileiros, pode-se ajustar modelos que considerem tanto as variações regionais quanto as flutuações sazonais e as tendências de longo prazo. Essa análise permite prever o impacto de mudanças demográficas, como o envelhecimento populacional, e de eventos específicos, como epidemias ou desastres naturais, sobre os anos de vida perdidos (BHASKARAN et al., 2013).

2.4 AVALIAÇÃO DE MODELOS PREDITIVOS DE SÉRIES TEMPORAIS

A avaliação de modelos preditivos de séries temporais desempenha um papel crucial no desenvolvimento de previsões robustas, especialmente em áreas como a saúde pública. A previsão de indicadores como os Anos de Vida Perdidos (YLL) pode impactar diretamente políticas de saúde e estratégias de prevenção. Para garantir a eficácia desses modelos, é essencial adotar métricas de avaliação apropriadas, métodos de validação consistentes e considerar características específicas dos dados temporais (BOX; JENKINS; REINSEL, 2015).

O desempenho dos modelos preditivos é frequentemente avaliado por métricas como o erro quadrático médio (MSE), o erro absoluto médio (MAE), o erro percentual médio absoluto (MAPE) e o erro quadrático médio da raiz (RMSE), além de métricas como o Theil's U2 e o Durbin-Watson, que são descritos abaixo:

- Erro Quadrático Médio (MSE): Mede a média dos quadrados das diferenças entre os valores previstos pelo modelo e os valores observados, oferecendo uma indicação da magnitude dos erros em um conjunto de dados (CHAI; DRAXLER, 2014). O MSE é sempre um valor não negativo e, quanto menor o seu valor, melhor é o desempenho do modelo avaliado. Como o erro é elevado ao quadrado, o MSE é particularmente sensível a grandes discrepâncias, penalizando erros maiores de forma mais acentuada. O intervalo para este indicador varia de zero (representando um modelo perfeito) até

valores infinitos, dependendo da magnitude dos erros no conjunto de dados analisado (CHAI; DRAXLER, 2014).

- Erro Absoluto Médio (MAE): Métrica estatística utilizada para medir a precisão de modelos preditivos, indicando a média das diferenças absolutas entre os valores observados e os previstos. O MAE é definido como a soma dos valores absolutos dos erros dividida pelo número total de observações, proporcionando uma interpretação direta do erro médio sem amplificar discrepâncias maiores, como ocorre com o MSE (BOX; JENKINS; REINSEL, 2015). Esse indicador varia de zero, representando um modelo perfeito sem erros, a valores positivos maiores, dependendo da magnitude dos erros do modelo avaliado. Como métrica, ele é particularmente útil em aplicações práticas devido à sua simplicidade e interpretação intuitiva (BOX; JENKINS; REINSEL, 2015).
- Erro Percentual Médio Absoluto (MAPE): Métrica utilizada para avaliar a precisão de modelos preditivos, calculando a média dos erros percentuais absolutos entre os valores previstos e os valores observados. Ele mede a precisão relativa de um modelo, sendo especialmente útil em contextos onde os valores previstos e observados permanecem substancialmente acima de zero (MYTTENAEREA et al., 2017). Sua interpretação é intuitiva, pois fornece o erro médio em termos percentuais. O intervalo deste indicador é de 0% (modelo perfeito) a valores indefinidamente altos, caso os erros percentuais sejam elevados, tornando-o particularmente sensível a discrepâncias quando os valores reais se aproximam de zero (MYTTENAEREA et al., 2017).
- Erro Quadrático Médio da Raiz (RMSE): Métrica estatística amplamente utilizada para medir o desempenho de modelos de previsão, avaliando a magnitude dos erros cometidos pelo modelo. Ele é calculado como a raiz quadrada da média dos erros quadráticos, fornecendo uma interpretação em termos das mesmas unidades das variáveis previstas e observadas (CHAI; DRAXLER, 2014). O RMSE é sensível a grandes discrepâncias, já que os erros são elevados ao quadrado antes de serem somados, o que o torna particularmente útil em contextos onde erros maiores são mais críticos. O intervalo do RMSE varia de zero (indicando previsões perfeitas) até valores infinitos, dependendo da magnitude dos erros no conjunto de dados avaliado. Essa métrica é especialmente apropriada quando os erros seguem uma distribuição aproximadamente normal (CHAI; DRAXLER, 2014).

- Erro de Theil (Theil's U2): Métrica utilizada para comparar a precisão de um modelo de previsão em relação a um modelo ingênuo. Ele mede a eficácia relativa de um método de previsão em termos de erro, sendo calculado como a razão entre o erro médio quadrático do modelo avaliado e o erro médio quadrático de um modelo ingênuo (GOHAIN, 2021). O valor do Theil's U2 varia de 0 a infinito, onde um valor igual a 1 indica que o modelo avaliado é tão bom quanto o modelo ingênuo. Valores menores que 1 sugerem que o modelo avaliado é melhor do que o ingênuo, enquanto valores maiores que 1 indicam que o modelo ingênuo apresenta melhor desempenho (GOHAIN, 2021). Essa métrica é particularmente útil em contextos onde modelos mais complexos precisam ser avaliados em relação a abordagens simples para justificar sua aplicabilidade prática.
- Durbin-Watson: Ferramenta utilizada para detectar a presença de autocorrelação nos resíduos de uma regressão linear, especialmente em dados de séries temporais. A autocorrelação ocorre quando os erros de uma observação estão correlacionados com os de observações anteriores, o que pode comprometer a validade das inferências estatísticas (DURBIN; WATSON, 1971). O valor do estatístico varia entre 0 e 4, onde um valor próximo de 2 indica ausência de autocorrelação; valores menores que 2 sugerem autocorrelação positiva; e valores maiores que 2 indicam autocorrelação negativa. Em geral, valores entre 1,5 e 2,5 são considerados aceitáveis, enquanto valores fora desse intervalo podem sinalizar problemas significativos de autocorrelação nos resíduos (DURBIN; WATSON, 1971).

Além dessas, o coeficiente de determinação (R^2) também pode ser usado para medir o ajuste do modelo aos dados observados. Porém, sua aplicação em séries temporais deve ser cautelosa, pois a dependência temporal pode inflar os resultados (SHUMWAY; STOFFER, 2017).

A validação de modelos de séries temporais exige abordagens específicas devido à dependência cronológica dos dados, tornando inadequados métodos tradicionais como a validação cruzada aleatória, que violam a ordem temporal. Para superar essa limitação, são amplamente utilizados métodos como a validação walk-forward, na qual o modelo é treinado em um subconjunto cronologicamente anterior e testado em dados subsequentes, replicando um cenário real de previsão (BOX; JENKINS; REINSEL, 2015). Outra abordagem comum é a rolling window, que mantém uma janela de dados de tamanho fixo e a move ao longo do tempo, permitindo que diferentes períodos sejam utilizados para treinamento e teste, garantindo uma

avaliação mais robusta e variada do desempenho do modelo (SHUMWAY; STOFFER, 2017). Esses métodos ajudam a evitar vazamentos de informações do futuro para o modelo, tornando a avaliação mais realista.

A estacionariedade é uma propriedade fundamental em séries temporais, especialmente para modelos como ARIMA. Séries estacionárias possuem propriedades estatísticas constantes, como média e variância. Caso os dados não sejam estacionários, técnicas como diferenciação podem ser aplicadas para ajustá-los (BOX; JENKINS; REINSEL, 2015).

Outro desafio é o ruído presente nos dados. Séries de saúde pública frequentemente contêm outliers ou flutuações aleatórias causadas por eventos inesperados. Métodos como suavização exponencial podem ser aplicados para reduzir o impacto dessas variações, melhorando a sensibilidade do modelo aos padrões reais (BOX; JENKINS; REINSEL, 2015).

A robustez de um modelo é essencial para garantir previsões confiáveis em diferentes cenários. Testar o modelo em contextos variados, como períodos temporais distintos ou regiões geográficas com características demográficas e socioeconômicas diferentes, avalia sua capacidade de generalização (SHUMWAY; STOFFER, 2017).

Estratégias como adição de ruído controlado aos dados de entrada ou exclusão aleatória de amostras podem verificar como o modelo lida com informações incompletas ou distorcidas. Isso é especialmente relevante em aplicações de saúde pública, onde a heterogeneidade entre os municípios brasileiros pode afetar a eficácia do modelo (BOX; JENKINS; REINSEL, 2015).

3 TRABALHOS RELACIONADOS

O objetivo desta seção é apresentar trabalhos relacionados existentes, para realizar uma comparação e buscar apresentar quais os diferenciais, contribuições e motivações que justificam a realização deste trabalho. Com essa finalidade, decidiu-se buscar apenas artigos acadêmicos publicados em revistas com fator de impacto acima de 2, com data de publicação entre 2019 e 2024, pesquisando exclusivamente nos periódicos Compendex, da Elsevier, e IEEE Xplore, por sua relevância na área de tecnologia.

Para buscar os trabalhos mais relevantes, por se tratarem de revistas internacionais, foi utilizada inicialmente a seguinte string de busca em título, palavras chave e resumo: *"time series" and "years of life lost" and "machine learning"*. Essa string não obteve resultado em nenhuma das duas bases de dados consultadas. Então a string foi reduzida para "*time series*" *and* "*years of life lost*". Desta vez, a busca trouxe 22 resultados na Compendex e nenhum resultado na IEEE Xplore. Os 22 artigos retornados tiveram seus títulos e resumos lidos para identificar os mais relevantes para este trabalho, e foram então selecionados 4 trabalhos.

Apesar de relevantes para o trabalho por tratarem de séries temporais e dos anos de vida perdidos, os trabalhos citados acima utilizaram em seus estudos apenas estatística tradicional, não utilizando aprendizagem de máquina para o atingimento dos objetivos propostos. Desta forma, decidiu-se completar essa pesquisa buscando novos estudos com a string *"time series" and "health care" and "machine learning"*, que tira o foco do indicador do YLL, mas busca trabalhos que utilizaram *machine learning* e séries temporais em estudos de assistência médica. Desta vez foram obtidos 18 resultados na IEEE Xplore e 44 na Compendex. Destes, foram selecionados mais 4 artigos que foram estudados e descritos a seguir.

3.1 TRABALHOS ANALISADOS

A seguir são apresentados os oito trabalhos analisados, incluindo seus objetivos, metodologias utilizadas, as ferramentas computacionais empregadas para chegar na solução proposta e os resultados obtidos.

a) *Life loss of cardiovascular diseases per death attributable to ambient temperature: A national time series analysis based on 364 locations in China*

Hu *et al.* (2021) analisam em seu estudo os efeitos da temperatura ambiente nas doenças cardiovasculares na China, especificamente avaliando a perda de anos de vida devido às temperaturas não ideais. O principal objetivo do estudo foi investigar a associação entre a temperatura ambiente e a perda de anos de vida por doenças cardiovasculares, além de quantificar a perda de vida relacionada à temperatura por cada morte. Em resumo, o estudo buscou compreender como as variações de temperatura afetam a expectativa de vida das pessoas que sofrem de doenças cardiovasculares na China.

Para alcançar os objetivos propostos, foram utilizados modelos estatísticos de regressão para analisar a associação entre a temperatura ambiente e a perda de anos de vida por doenças cardiovasculares. Em particular, foi empregado o modelo DLNM (*Distributed Lag Non-linear Model*) para construir curvas de exposição-resposta entre a temperatura diária média e as taxas de perda de anos de vida. Além disso, foram realizadas análises de sensibilidade para verificar a robustez dos resultados, incluindo a comparação entre períodos de estudo diferentes e a avaliação do impacto dos poluentes do ar ajustados e não ajustados nos modelos estatísticos.

A realização das análises dos dados se deu com a utilização do software R (versão 3.6.0), onde foi empregado o pacote "dlnm" para construir o modelo DLNM e o pacote "mvmeta" para realizar a meta-análise multivariada e a melhor previsão linear não tendenciosa. Além disso, as figuras foram geradas utilizando o pacote "ggplot2". O software R foi fundamental para a análise estatística dos dados e a visualização dos resultados obtidos no estudo.

b) *Short-term effects of ambient nitrogen dioxide on years of life lost in 48 major Chinese cities, 2013–2017*

Li *et al*. (2021) analisaram as associações entre a exposição ao dióxido de nitrogênio (NO_2) e os anos de vida perdidos (YLL) em 48 grandes cidades chinesas no período de 2013 a 2017. O principal objetivo foi investigar como a exposição ao NO_2 afeta a perda de anos de vida devido a causas não acidentais, doenças cardiovasculares (CVD) e doenças respiratórias (RD) nessas cidades. Além disso, o estudo buscou avaliar se fatores como idade, sexo, nível educacional, temperatura e características específicas das cidades poderiam modular essas associações. Os resultados fornecem *insights* importantes sobre o impacto da poluição do ar por

NO_2 na saúde da população e podem orientar políticas públicas e intervenções para reduzir os efeitos adversos da exposição ao NO_2 nos anos de vida perdidos.

O estudo utilizou modelos aditivos generalizados para estimar as mudanças percentuais relativas de YLL associadas ao NO_2 em cada uma das 48 cidades chinesas. Posteriormente, esses resultados foram combinados por meio de modelos de efeitos aleatórios para gerar efeitos médios. Além disso, foram realizadas análises estratificadas por fatores demográficos individuais e temperatura, bem como análises de meta-regressão incorporando concentrações de poluentes do ar específicas de cada cidade, condições meteorológicas e indicadores socioeconômicos para explorar possíveis modificações nos efeitos. Esses métodos estatísticos permitiram aos pesquisadores avaliar as associações entre a exposição ao NO_2 e os anos de vida perdidos, considerando diferentes variáveis e contextos específicos de cada cidade.

c) *Ambient sulfur dioxide and years of life lost from stroke in China: a time-series analysis in 48 cities*

Wu *et al*. (2021) avaliaram a associação entre a poluição do ar por dióxido de enxofre (SO_2) e a perda de anos de vida devido a acidentes vasculares cerebrais (AVCs) na China. O objetivo principal era estimar o impacto de curto prazo da exposição ao SO_2 na perda de anos de vida por AVC, bem como calcular as perdas de anos de vida e econômicas associadas a essa relação. A pesquisa foi conduzida em 48 cidades chinesas e buscou identificar subgrupos populacionais sensíveis aos efeitos adversos do SO_2, destacando a necessidade de medidas específicas de mitigação da poluição do ar para proteger a saúde pública.

O estudo utilizou uma análise de séries temporais nacional abrangendo 48 grandes cidades na China para examinar a associação de curto prazo entre a poluição por SO_2 e a perda de anos de vida por AVC. Foram empregados modelos de Regressão Aditiva Generalizada (GAM) e Análise de Sobrevivência de Vida Livre de Anos (VSLY) para explorar o excesso de perda de anos de vida e as perdas econômicas relacionadas. Além disso, foram realizadas análises estratificadas e de modificação de efeito para identificar subgrupos sensíveis à exposição ao SO_2. Subsequentemente, foram calculados os anos de vida perdidos e as perdas econômicas associadas para avaliar a significância em termos de saúde pública dos resultados obtidos.

As análises foram conduzidas principalmente utilizando modelos estatísticos, como Regressão Aditiva Generalizada (GAM) e Análise de Sobrevivência de Vida Livre de Anos (VSLY), para investigar a associação entre a poluição por SO_2 e a perda de anos de vida por

AVC. As análises foram realizadas utilizando o software R (versão 3.4.1). Foram também utilizados os pacotes “mgcv”, “metafor” e “mvmeta” para realizar as análises estatísticas.

d) *Lower-than-standard particulate matter air pollution reduced life expectancy in Hong Kong: A time-series analysis of 8.5 million years of life lost*

Cheng *et al*. (2021) realizaram o estudo com o objetivo de investigar as associações entre a exposição a níveis de material particulado abaixo dos padrões de qualidade do ar e a perda de anos de vida. Os pesquisadores utilizaram dados de mortalidade e poluição do ar ao longo de 17 anos para analisar como a exposição a partículas PM2.5 e PM10 afeta a expectativa de vida, tanto para a população total quanto por gênero, idade e principais causas de doenças. Eles buscaram quantificar o impacto na perda de anos de vida e na expectativa de vida devido à poluição do ar, destacando os potenciais benefícios para a saúde ao reduzir ainda mais os níveis de poluição atmosférica de fundo.

O estudo utilizou um desenho de estudo de séries temporais para avaliar o impacto da exposição de curto prazo a material particulado (PM2.5 e PM10) na mortalidade e na expectativa de vida. Esse desenho de estudo é comumente utilizado em epidemiologia ambiental e permite analisar se as variações diárias na concentração de poluentes atmosféricos estão relacionadas com a mortalidade. Para analisar as associações entre as partículas e a perda de anos de vida, os pesquisadores empregaram um modelo não linear de defasagem distribuída que pode capturar associações lineares e não lineares. Através dessas análises, eles puderam calcular a perda total de anos de vida e a expectativa de vida reduzida devido à poluição do ar, tanto para exposições abaixo quanto acima de uma série de padrões de qualidade do ar ambiental.

e) *Energy-efficient dynamic sensor time series classification for edge health devices*

O trabalho de Wang e Sun (2024) aborda a classificação de séries temporais de dados em dispositivos de saúde de borda, com foco na eficiência energética e na detecção em tempo real de doenças no contexto da Internet das Coisas Médicas (IoMT). Os principais objetivos são desenvolver um algoritmo de classificação de séries temporais online que mitigue os problemas de *"concept drift"* (mudança de conceito) e *"catastrophic forgetting*" (esquecimento catastrófico), que são desafios comuns em ambientes de aprendizado contínuo.

A metodologia utilizada envolve o desenvolvimento do algoritmo denominado OTCD (*Online Time series classification algorithm*). O OTCD detecta a ocorrência de *"concept drift"* e atualiza protótipos para minimizar seu impacto. Além disso, padroniza a distribuição do espaço potencial e preserva seletivamente parâmetros de treinamento essenciais para lidar com o "*catastrophic forgetting*". A avaliação do desempenho do modelo foi realizada utilizando dados de eletrocardiograma (ECG) e fotopletismografia (PPG).

Os modelos de *machine learning* comparados no estudo incluem Redes Neurais Recorrentes (RNN), Perceptron de Múltiplas Camadas (MLP) e Redes Neurais Convolucionais Temporais (TCN). O desempenho do OTCD foi comparado com outros métodos de classificação de séries temporais dinâmicas, como CoPE, GEM e EWC.

Os resultados mostraram que o OTCD superou os outros métodos em termos de precisão média, apresentando uma melhoria de 14,74% em relação ao EWC em um dos conjuntos de dados. O OTCD demonstrou ser mais robusto em uma variedade de conjuntos de dados, especialmente em cenários onde o "*concept drift*" afetou a precisão dos modelos. As métricas de avaliação incluíram precisão, F1-score, erro absoluto médio (MAE) e erro quadrático médio (MSE), com o OTCD apresentando resultados superiores em todas essas métricas.

f) *Data Analysis and Forecasting of the COVID-19 Spread: A Comparison of Recurrent Neural Networks and Time Series Models*

O trabalho de Gomez-Cravioto et al. (2024) aborda a análise e previsão da disseminação do vírus SARS-CoV-2 no México, utilizando técnicas de *machine learning* e modelos estatísticos. Os principais objetivos do estudo são: identificar a função que melhor se ajusta ao crescimento da população infectada, determinar a importância de variáveis climáticas e de mobilidade, e comparar os resultados de modelos tradicionais de séries temporais com abordagens modernas de *machine learning*.

A metodologia utilizada envolve a coleta de dados de casos confirmados e fatalidades da COVID-19 a partir do repositório da Universidade Johns Hopkins, complementados por informações sobre clima e mobilidade social. Os dados foram explorados e preparados para modelagem, utilizando Python e a plataforma Jupyter Notebook. As ferramentas de informática empregadas no trabalho incluem Python, Jupyter Notebook e bibliotecas de *machine learning*, que possibilitaram a análise e visualização dos dados. Os modelos de *machine learning* utilizados especificamente foram: redes neurais do tipo *Long*

Short-Term Memory (LSTM), que se mostraram eficazes na previsão de casos diários. Os modelos estatísticos utilizados incluem regressões linear, polinomial e logística generalizada, que foram aplicados para descrever o crescimento dos casos de COVID-19.

Os resultados obtidos indicaram que o modelo logístico se ajustou melhor ao comportamento da pandemia, e que as variáveis climáticas e de mobilidade apresentaram correlação significativa com os números da doença. Além disso, o modelo LSTM demonstrou potencial para prever casos diários, destacando a eficácia das técnicas de *machine learning* em comparação com os métodos tradicionais.

g) *Time Series Prediction Using Deep Learning Methods in Healthcare*

O trabalho trata da predição de séries temporais no setor de saúde, utilizando métodos de *deep learning* (DL) para lidar com os desafios da alta dimensionalidade e irregularidade dos dados médicos. O objetivo principal é revisar e organizar as pesquisas relacionadas ao uso de redes neurais profundas para predições baseadas em dados estruturados de séries temporais de pacientes.

O estudo adota uma revisão sistemática da literatura, com buscas em bancos de dados como MEDLINE, IEEE, Scopus e ACM Digital Library, focando em trabalhos que empregam redes neurais profundas para predição de eventos clínicos com dados estruturados. A pesquisa se concentrou nas características técnicas dos modelos de *deep learning*, como arquitetura, processamento de dados e estratégias de aprendizagem. Foram utilizados modelos de *machine learning*, especificamente os modelos de *deep learning*. Entre os modelos empregados estão as redes LSTM (*Long Short-Term Memory*) e GRU (*Gated Recurrent Unit*), além de CNN e variações bidirecionais desses modelos, como Bi-LSTM e Bi-GRU.

O estudo destaca que os modelos de *deep learning*, em especial os modelos GRU e LSTM, apresentaram resultados superiores em termos de acurácia preditiva para várias tarefas no setor de saúde, como predição de falha cardíaca, mortalidade e readmissão hospitalar. Modelos GRU, por exemplo, mostraram uma melhoria de até 1% em relação aos LSTM em algumas aplicações específicas.

h) *Improved healthcare monitoring of coronary heart disease patients in time-series fashion using deep learning model*

O trabalho trata da melhoria do monitoramento de pacientes com doenças cardíacas coronárias utilizando modelos de aprendizado profundo (*deep learning*). O objetivo principal é desenvolver um framework de monitoramento em série temporal para prever a evolução da condição dos pacientes, melhorando a precisão das previsões a partir de sinais médicos.

O estudo adota uma metodologia baseada em aprendizado de máquina, especificamente usando uma Rede Neural Convolucional (CNN) integrada a funções de base radial e redes neurais artificiais para classificar os dados temporais obtidos de eletrodos. O modelo foi testado com um conjunto de dados de registros médicos de uma universidade, contendo 335 registros e 36 atributos clínicos.

As ferramentas principais incluem a linguagem de programação Python para simulação e implementação dos modelos, além da utilização de bibliotecas como TensorFlow, focadas em aprendizado profundo. Foram empregados modelos de aprendizado de máquina, especificamente o CNN para classificação de séries temporais, além de técnicas como Recurrent Neural Networks (RNN) para análise de previsões. Comparações foram feitas com outros modelos como ARIMA, DNN-LSTM e SARIMA.

Os resultados mostraram que o modelo proposto apresentou alta precisão, com uma acurácia de 97%, superando outros modelos tradicionais em termos de recall e F1-Score. O framework de monitoramento possibilita a previsão em tempo real da evolução dos pacientes, o que pode levar a melhorias significativas na gestão da saúde pública para pacientes com doenças cardíacas.

3.2 COMPARAÇÃO DOS TRABALHOS E INOVAÇÃO

Após a análise dos trabalhos existentes, é possível identificar algumas tendências e lacunas na literatura atual, bem como destacar as contribuições inovadoras da presente proposta. A maioria dos estudos analisados (HU et al., 2021; LI et al., 2021; WU et al., 2021; CHENG et al., 2021) concentra-se em dados da China ou de Hong Kong, com um foco em cidades de grande porte. Em contraste, o presente trabalho propõe uma análise abrangente dos municípios

brasileiros de médio porte, preenchendo uma lacuna importante na literatura sobre Anos de Vida Perdidos (YLL) no contexto brasileiro.

Os estudos existentes sobre YLL (HU et al., 2021; LI et al., 2021; WU et al., 2021; CHENG et al., 2021) utilizam predominantemente métodos estatísticos tradicionais, como modelos de regressão e análises de séries temporais. Em contrapartida, este trabalho propõe a aplicação de técnicas avançadas de aprendizagem de máquina, alinhando-se com as tendências mais recentes em análise de dados de saúde, como observado nos trabalhos de Wang e Sun (2024) e Gomez-Cravioto et al. (2024).

Enquanto a maioria dos estudos analisados focam em análises retrospectivas ou de curto prazo, este trabalho visa desenvolver um modelo preditivo capaz de fornecer previsões futuras de YLL. Isso se alinha com a abordagem de Gomez-Cravioto et al. (2024) na previsão de casos de COVID-19, mas aplicada ao contexto específico de YLL.

O trabalho proposto busca avaliar e comparar diferentes algoritmos de aprendizagem de máquina para a previsão de YLL, indo além das abordagens tradicionais. Isso se assemelha à metodologia de Aravind e Neethu (2023), que compararam vários modelos de *deep learning* para previsões em saúde, mas com um foco específico em YLL.

Uma inovação significativa deste trabalho é a proposta de utilizar técnicas de nowcasting para prever os indicadores presentes de YLL, considerando a defasagem na disponibilização dos dados atuais, prevista pela Portaria SVS nº 116, de 11 de fevereiro de 2009 (MINISTÉRIO DA SAÚDE, 2009). Isso não foi observado em nenhum dos estudos analisados e representa uma contribuição original para o campo.

A proposta de desenvolver um pipeline automatizado de extração, transformação e persistência de dados é uma contribuição técnica importante, que pode facilitar a replicação e extensão do estudo para outros contextos.

Em resumo, este trabalho se distingue por sua abordagem inovadora que combina:

- Foco específico nos municípios brasileiros de médio porte;
- Utilização de técnicas de *machine learning* para previsão de YLL;
- Desenvolvimento de um modelo preditivo com capacidade de nowcasting;
- Criação de um pipeline automatizado para processamento de dados de mortalidade.

Estas características posicionam o trabalho proposto como uma contribuição significativa e original para o campo de estudo dos Anos de Vida Perdidos, com potencial para impactar positivamente as políticas de saúde pública no Brasil.

4 METODOLOGIA

Para alcançar os objetivos definidos neste trabalho, a execução foi organizada em duas etapas principais: a primeira voltada para atividades de engenharia de dados e a segunda para ciência de dados.

Na primeira etapa, foi desenvolvido um pipeline de dados com o objetivo de automatizar o processo de extração, transformação e carga (ETL). Esse pipeline foi projetado para tornar o processo de manipulação de dados mais ágil, replicável e menos suscetível a erros manuais, garantindo a consistência e a integridade dos dados necessários para as análises.

A segunda etapa concentrou-se no desenvolvimento de modelos preditivos, seguindo a metodologia CRISP-DM (Cross-Industry Standard Process for Data Mining). Essa metodologia foi aplicada para estruturar o processo de modelagem, assegurando uma abordagem sistemática e orientada à obtenção de modelos eficazes para a predição dos indicadores de YLL.

Embora sejam etapas distintas, ambas ocorreram de forma interdependente em determinados momentos. A execução do pipeline dependia da definição dos dados a serem utilizados e de como eles deveriam ser entregues para análise. Por outro lado, a modelagem exigia que os dados estivessem previamente disponíveis e devidamente transformados. Dessa forma, as etapas iniciais da modelagem, como compreensão do problema e definição dos dados necessários, foram realizadas em paralelo ao desenvolvimento do pipeline. Após a disponibilização dos dados, as etapas subsequentes da modelagem puderam ser executadas de forma completa.

4.1 PIPELINE DE DADOS

Na etapa inicial da metodologia de modelagem, identificou-se a necessidade de dados abrangentes de mortalidade no Brasil, contendo informações como a data do óbito, a idade do falecido e o município de residência. Para calcular a taxa de Anos de Vida Perdidos (YLL) e classificar os portes dos municípios, foram utilizados dados complementares sobre o tamanho da população por município e período, além de metadados para relacionar os códigos

IBGE às respectivas localidades, já que a todas as bases de dados utilizam apenas esses códigos para identificação.

Os dados de população foram extraídos do Ministério da Saúde (DATASUS), acessados via Tabnet (http://tabnet.datasus.gov.br), que fornece estimativas anuais por município. Esses dados foram fundamentais para a classificação dos municípios por portes, que no Brasil, conforme estabelecido pela Política Nacional de Assistência Social (PNAS/2004), estão relacionados diretamente à população residente e servem como critério para a organização, planejamento e execução das políticas públicas, especialmente no âmbito da assistência social. Essa classificação é essencial para determinar a estrutura e os recursos necessários para atender as demandas específicas de cada município, considerando suas peculiaridades socioeconômicas e demográficas (Secretaria Nacional de Assistência Social, 2004).

A definição de porte, de acordo com a PNAS/2004, utiliza como base o tamanho populacional, distribuindo os municípios em cinco categorias principais: Pequenos I (até 20.000 habitantes), Pequenos II (20.001 a 50.000 habitantes), Médios (50.001 a 100.000 habitantes), Grandes (100.001 a 900.000 habitantes) e Metrópoles (mais de 900.000 habitantes). Essa estratificação reflete a diversidade territorial do país, abrangendo desde cidades de pequena escala, com predomínio de áreas rurais, até grandes centros urbanos com alta densidade populacional e complexidade socioeconômica (Secretaria Nacional de Assistência Social, 2004).

Os municípios de pequeno porte, que representam 73% do total de cidades brasileiras, possuem características específicas, como uma significativa parcela da população vivendo em áreas rurais, muitas vezes com acesso limitado a serviços essenciais. Por outro lado, as metrópoles concentram 20% da população nacional e enfrentam desafios distintos, como a urbanização acelerada, desigualdades internas e uma elevada demanda por políticas públicas integradas (Secretaria Nacional de Assistência Social, 2004).

Essa classificação não apenas facilita o entendimento das dinâmicas locais, mas também orienta o planejamento de ações de assistência social, permitindo uma distribuição mais equitativa de recursos e esforços para reduzir desigualdades. Por exemplo, municípios menores frequentemente necessitam de apoio financeiro e técnico adicional para implementar políticas públicas, enquanto as metrópoles demandam estratégias para lidar com a alta densidade populacional e os desafios da urbanização.

Além disso, a categorização dos municípios por porte influencia diretamente a implementação do Sistema Único de Assistência Social (SUAS), uma vez que as

especificidades de cada município impactam a operacionalização das ações socioassistenciais, como a alocação de Centros de Referência de Assistência Social (CRAS) e a execução de programas voltados para populações em situação de vulnerabilidade (Secretaria Nacional de Assistência Social, 2004).

Os dados de mortalidade foram obtidos do Departamento de Informática do Sistema Único de Saúde (DATASUS), disponíveis no repositório de dados abertos do Governo Federal (https://dados.gov.br). Para análise, consideraram-se apenas óbitos relacionados a causas evitáveis, ou seja, aqueles que poderiam ser prevenidos por atenção primária ou políticas públicas. Assim, utilizaram-se os registros da Lista Brasileira de Internações por Condições Sensíveis à Atenção Primária (ICSAP), conforme a classificação da Décima Revisão da Classificação Internacional de Doenças (CID-10), definida na Portaria nº 221/2008 do Ministério da Saúde.

Devido ao impacto da pandemia de COVID-19 entre 2020 e 2022 e ao possível desvio nos registros de mortes evitáveis durante este período, restringiu-se a análise ao período de 2010 a 2019, anterior à pandemia.

Além disso, as datas de nascimento e óbito de cada indivíduo foram utilizadas para calcular a idade no momento da morte. Durante a análise, observou-se que 1,21% dos registros apresentavam idades negativas, indicando erros nos registros. Esses dados foram excluídos da base.

Para calcular o índice de YLL por óbito, foi necessário utilizar a tabela de expectativa de vida por faixa etária, conforme a Organização das Nações Unidas (2020). Esses dados foram fundamentais como variável-alvo nos modelos de aprendizado de máquina. A Tabela 1 apresenta as faixas etárias e respectivas expectativas de vida utilizadas.

Tabela 1 – Expectativa de vida por faixa etária

Faixa Etária	Expectativa de Vida (anos)
0 a 27 dias	89,99
28 a 364 dias	89,55
1 a 4 anos	89,07
5 a 9 anos	82,58
10 a 14 anos	77,58
15 a 19 anos	72,60
20 a 24 anos	67,62
25 a 29 anos	62,66
30 a 34 anos	57,71
35 a 39 anos	52,76
40 a 44 anos	47,83
45 a 49 anos	42,94
50 a 54 anos	38,10
55 a 59 anos	33,33
60 a 64 anos	28,66
65 a 69 anos	24,12
70 a 74 anos	19,76
75 a 79 anos	15,65
80 a 84 anos	11,69
85 anos ou mais	7,05

Fonte: adaptado de Organizações das Nações Unidas (2020)

Os metadados para associar municípios às Unidades Federativas (UF) foram obtidos da Receita Federal (https://www.gov.br/receitafederal).

Para garantir a extração, transformação e carga dos dados de forma eficiente e segura, foi desenvolvido um pipeline de dados em Python, automatizando a manipulação e armazenamento das informações. O Google BigQuery, plataforma de banco de dados do Google Cloud, foi escolhido para a persistência dos dados, devido à sua acessibilidade remota e simplicidade de uso.

O pipeline executa as seguintes etapas principais:

- Conexão com o BigQuery: Verifica a existência do dataset e das tabelas; se necessário, cria ambos.
- Extração de dados: Baixa os arquivos das fontes especificadas para um repositório local.
- Transformação de dados: Utiliza a biblioteca Pandas para manipulação, limpeza e cálculo de métricas, como o índice YLL.

- Carga de dados: Armazena os dados processados no BigQuery e exporta-os em formato CSV para uma pasta local.

O código do pipeline está disponível em um repositório aberto no GitLab da Universidade Federal de Santa Catarina (codigos.ufsc.br) e pode ser acessado em: https://codigos.ufsc.br/marco.cesar/yll_time_series_machine_learning.

4.2 MODELO PREDITIVO

Para o desenvolvimento do modelo preditivo, a metodologia adotada neste trabalho seguiu as diretrizes do CRISP-DM (Cross Industry Standard Process for Data Mining), reconhecido como uma referência paradigmática na gestão de projetos envolvendo atividades de *machine learning* (MARTINEZ-PLUMED et al., 2021).

O CRISP-DM é uma metodologia abrangente e consolidada para mineração de dados, oferecendo uma estrutura completa que orienta usuários de diferentes níveis de experiência, de iniciantes a especialistas, na execução de projetos de *Data Mining* (RAMOS et al., 2020). O processo é dividido em seis fases principais: Entendimento do negócio; Compreensão dos dados; Preparação dos dados; Modelagem; Avaliação; Implantação. Cada fase desse processo tem seus objetivos específicos e atividades associadas, visando guiar de forma eficiente a execução de projetos de mineração de dados, como é detalhado a seguir.

1) Compreensão da questão de negócio

A primeira etapa do CRISP-DM consiste em compreender os objetivos do projeto de mineração de dados sob a perspectiva do negócio. Essa fase é essencial para alinhar os objetivos técnicos com as necessidades organizacionais, permitindo uma definição clara do problema a ser resolvido e a elaboração de um plano de ação preliminar para o projeto (RAMOS et al., 2020).

Nesse contexto, realizou-se uma revisão bibliográfica com o objetivo de identificar as características relevantes para o cálculo de YLL. A revisão incluiu a análise do arcabouço legal e da literatura científica sobre o tema, permitindo a formulação do problema de pesquisa deste trabalho: prever os indicadores de YLL para municípios de médio porte no Brasil, no

período de 2020 a 2022. Essa etapa foi fundamental para garantir que o desenvolvimento do modelo preditivo estivesse alinhado com as necessidades do domínio e com as melhores práticas descritas na literatura.

2) Compreensão dos dados

A segunda etapa do CRISP-DM consiste em compreender os dados, começando com a coleta inicial. O objetivo é desenvolver familiaridade com o conjunto de dados, identificar possíveis problemas de qualidade e explorar ideias preliminares ou subconjuntos relevantes que possam revelar padrões e informações úteis. Essa fase é crucial para antecipar e mitigar problemas que possam surgir na preparação dos dados, geralmente a etapa mais trabalhosa do projeto (RAMOS et al., 2020).

Neste trabalho, os dados de mortalidade foram extraídos diretamente do DATASUS por meio de um repositório FTP público, enquanto os dados geográficos dos municípios brasileiros foram obtidos junto ao Instituto Brasileiro de Geografia e Estatística (IBGE). Adicionalmente, foram utilizados metadados que traduzem os códigos CID-10 (Organização Mundial da Saúde, 2007) e os códigos IBGE dos municípios. Uma análise exploratória inicial foi conduzida para identificar valores nulos, verificar a completude dos dados e definir critérios de agrupamento para as séries históricas.

Dado o impacto da pandemia de COVID-19 em 2020 e possíveis alterações na classificação das doenças ao longo das atualizações do CID, optou-se por trabalhar com a série histórica de 2010 a 2019. Para garantir a comparabilidade entre os municípios, foi calculada a taxa de YLL por 1.000 habitantes, dividindo o YLL pela população projetada e multiplicando o resultado por 1.000. Essa abordagem permitiu observar padrões entre diferentes portes de municípios e avaliar questões como completude, outliers e distribuição dos dados.

Todos os dados foram armazenados no BigQuery da Google Cloud Platform para facilitar o processamento e a análise. Durante o ETL, os registros foram consolidados, totalizando mais de 12 milhões de óbitos no período analisado, dos quais aproximadamente 1,76 milhão incluíam um código CID-10 referente à Lista Brasileira de Internações por Condições Sensíveis à Atenção Primária (ICSAP). Registros com informações inconsistentes, como idade negativa ou ausência de data de óbito, foram descartados, resultando em 1.755.025 óbitos válidos.

Uma análise exploratória revelou índices médios de YLL superiores a 80 em diversos municípios, indicando alta mortalidade infantil ou possíveis inconsistências nos

registros, especialmente em municípios de pequeno porte. Um gráfico do desvio padrão do YLL por município reforçou essa concentração em localidades menores, possivelmente devido à variabilidade nos registros.

Com base nessa análise, decidiu-se segmentar a modelagem para considerar apenas municípios de médio porte, garantindo maior uniformidade e confiabilidade nos resultados. Essa decisão também buscou reduzir o impacto de inconsistências que poderiam distorcer a análise e prejudicar a eficácia dos modelos preditivos.

3) Preparação dos dados

A terceira etapa do CRISP-DM, conforme descrito por Ramos et al. (2020), é dedicada à preparação dos dados, sendo uma das fases mais cruciais do processo de mineração de dados. Nessa etapa, são realizadas todas as atividades necessárias para transformar os dados brutos em um conjunto adequado para a modelagem. Essas atividades incluem a seleção de tabelas, registros e atributos, bem como a transformação, limpeza e formatação dos dados. É comum que essas tarefas sejam iterativas, ocorrendo em diferentes ordens e sendo repetidas várias vezes ao longo do processo.

No presente trabalho, essa etapa foi integrada ao processo de ETL, com o objetivo de facilitar futuras análises e permitir que o foco do pesquisador estivesse na modelagem e parametrização dos modelos preditivos. Os dados foram lidos diretamente da Google Cloud Platform (GCP) utilizando Jupyter Notebook e a linguagem Python, que possibilitaram o agrupamento e a transformação necessária para a criação de um conjunto de dados robusto.

Entre as atividades realizadas, destacam-se:

- Limpeza de dados: Remoção de registros inconsistentes, como valores nulos, idades negativas e datas de óbito ausentes.
- Transformação: Cálculo da taxa de YLL por 1.000 habitantes, normalizando os dados para facilitar comparações entre municípios.
- Agrupamento: Estruturação dos dados em séries temporais quadrimestrais, considerando apenas municípios de médio porte, conforme definido na etapa anterior.

Essas tarefas garantiram a qualidade e a consistência dos dados, aspectos fundamentais para a eficácia dos modelos preditivos. O resultado foi um conjunto de dados

pronto para ser utilizado na etapa de modelagem, reduzindo o esforço em etapas posteriores e promovendo maior eficiência no desenvolvimento do modelo.

Com os dados devidamente preparados, o foco do trabalho pôde se voltar à criação de modelos que fossem capazes de atender aos objetivos de previsão definidos para o projeto.

4) Modelagem

A quarta etapa do CRISP-DM é a modelagem, onde técnicas específicas são aplicadas para resolver o problema de mineração de dados. Nessa fase, os analistas utilizam diferentes algoritmos e abordagens, ajustando parâmetros e avaliando métricas de desempenho para otimizar os resultados. O processo é iterativo, permitindo revisões e ajustes contínuos, incluindo retornos à etapa de preparação dos dados para adequá-los às necessidades dos modelos selecionados (RAMOS et al., 2020).

Para este trabalho, foram selecionados e implementados cinco modelos preditivos: ARIMA, SARIMA, XGBoost, Prophet e LSTM. A escolha desses modelos foi baseada em suas características e capacidades específicas, já descritas na sessão de fundamentação teórica e relembradas abaixo:

- ARIMA e SARIMA: Modelos tradicionais de séries temporais, amplamente utilizados para capturar padrões sazonais e tendências em dados históricos.
- XGBoost: Um algoritmo de aprendizado de máquina baseado em árvores de decisão, capaz de lidar com variáveis não lineares e produzir resultados robustos em cenários complexos.
- Prophet: Desenvolvido pelo Facebook, é ideal para séries temporais com sazonalidades complexas e períodos de feriados. Sua facilidade de uso e capacidade de incorporar sazonalidades explícitas justificaram sua inclusão.
- LSTM (Long Short-Term Memory): Uma variante de redes neurais recorrentes, projetada para capturar dependências de longo prazo em séries temporais, sendo particularmente eficaz para dados com padrões não lineares.

Os modelos foram desenvolvidos utilizando Python, com bibliotecas específicas para cada técnica. Durante o processo, parâmetros iniciais foram definidos com base em configurações padrão, seguidos por ajustes iterativos para otimização.

Essa etapa de modelagem representa o núcleo do processo de mineração de dados, proporcionando as bases para gerar previsões precisas e contribuir para a tomada de decisões fundamentadas. A seguir, as previsões geradas pelos modelos serão avaliadas em relação aos dados reais para determinar sua eficácia e aplicabilidade.

5) Avaliação

A avaliação é uma etapa fundamental no CRISP-DM, pois permite verificar se o modelo construído atende aos objetivos do projeto e às expectativas do negócio. Além de garantir que os resultados sejam confiáveis, essa fase busca identificar possíveis lacunas ou problemas que possam ter sido negligenciados durante o desenvolvimento. Caso os resultados não sejam satisfatórios, o processo pode ser revisitado desde as etapas iniciais (RAMOS et al., 2020).

Para avaliar os modelos desenvolvidos, foram utilizadas as seguintes métricas de erro e validação, amplamente recomendadas na literatura para a análise de séries temporais, já descritas na sessão de fundamentação teórica e relembradas abaixo:

- Erro Absoluto Médio (MAE): Mede a magnitude média dos erros, fornecendo uma interpretação direta do desvio médio das previsões em relação aos valores reais.
- Erro Quadrático Médio (MSE): Penaliza erros maiores de forma mais severa, destacando a influência de outliers e erros significativos.
- Raiz do Erro Quadrático Médio (RMSE): Fornece uma métrica na mesma escala dos dados originais, facilitando a interpretação dos resultados.
- Erro Percentual Médio Absoluto (MAPE): Expressa os erros como uma porcentagem dos valores reais, permitindo comparações em diferentes escalas.
- Erro de Theil (Theil's U2): Compara o desempenho do modelo em relação a previsões ingênuas (como usar o valor anterior), avaliando sua eficácia relativa.
- Durbin-Watson: Avalia a autocorrelação dos resíduos, indicando se os erros estão correlacionados ao longo do tempo, o que pode sugerir inadequações no modelo.

Essas métricas foram calculadas para cada modelo, fornecendo uma visão abrangente do desempenho em termos de precisão, robustez e adequação aos dados. Os resultados obtidos orientaram a seleção do modelo final e indicaram possíveis ajustes para melhorar sua eficácia. Além disso, a análise residual, com base na estatística de Durbin-Watson,

ajudou a garantir que os modelos fossem adequados para séries temporais, sem autocorrelações significativas nos erros.

Essa etapa de avaliação não apenas valida os resultados obtidos, mas também permite um refinamento contínuo dos modelos, garantindo que estejam alinhados com os objetivos do projeto e com as melhores práticas para análise de séries temporais.

6) Implantação

A etapa final do CRISP-DM é a implantação, onde os resultados do projeto são aplicados de forma prática, gerando valor para a organização. Essa fase pode incluir a entrega de relatórios, dashboards interativos ou a implementação de processos automatizados para utilização contínua do modelo preditivo (RAMOS et al., 2020).

Neste trabalho, o modelo LSTM treinado foi salvo em formato .h5, que é amplamente utilizado para modelos baseados em Keras e TensorFlow. Esse formato permite armazenar tanto a arquitetura quanto os pesos do modelo, garantindo que ele possa ser reutilizado para previsões futuras. O arquivo gerado, chamado lstm_model.h5, está disponível para ser carregado em ambientes locais ou em plataformas de nuvem, como o Google Cloud Platform, caso seja necessário expandir o projeto.

Embora a implantação em ambiente produtivo na nuvem não tenha sido realizada neste momento, o modelo foi devidamente estruturado para que possa ser integrado futuramente em um sistema automatizado. Esse sistema poderá incluir um pipeline de previsões contínuas, armazenamento em banco de dados e visualização por meio de dashboards. O arquivo do modelo e as previsões futuras foram armazenados localmente, facilitando a análise e validação dos resultados.

Essa etapa final marca o encerramento do processo de desenvolvimento, com o modelo pronto para ser utilizado em aplicações práticas, como a previsão de indicadores de saúde para municípios de médio porte no Brasil, contribuindo para a formulação de políticas públicas mais eficientes.

5 RESULTADOS E DISCUSSÃO

Neste capítulo, são apresentados os resultados das análises realizadas e discutidos os principais achados relacionados à taxa média de Anos de Vida Perdidos (YLL) em municípios brasileiros de médio porte. Inicialmente, é realizada uma análise exploratória da série temporal, que permite identificar características como tendências, sazonalidades e variações nos dados. Em seguida, são descritos os processos de modelagem preditiva, onde diferentes técnicas foram aplicadas e avaliadas em busca do modelo mais eficiente para as previsões.

Os modelos analisados incluem abordagens tradicionais, como ARIMA e SARIMA, além de métodos baseados em aprendizado de máquina, como XGBoost, Prophet e LSTM. Cada um desses modelos é detalhado quanto à construção, parametrização e resultados obtidos, com discussões sobre seu desempenho. Por fim, a seção inclui uma análise comparativa entre os modelos e apresenta previsões futuras realizadas com o modelo mais eficaz, fornecendo insights importantes para a gestão e planejamento em saúde pública.

5.1 PIPELINE DE DADOS

O pipeline de dados desenvolvido foi uma etapa fundamental para o sucesso deste trabalho, possibilitando a construção de um banco de dados consistente e otimizado, armazenado no Google BigQuery. Este processo automatizado garantiu a extração, transformação e carga (ETL) de dados oriundos de diversas fontes, como os registros de mortalidade do DATASUS, tabelas de população estimada e metadados de municípios brasileiros. O resultado é um banco consolidado que viabiliza as análises e modelagens preditivas com eficiência e confiabilidade.

O pipeline foi projetado seguindo práticas modernas de engenharia de dados, utilizando a linguagem Python e bibliotecas como Pandas e BigQuery SDK. As etapas principais incluíram:

- Extração: Dados brutos de mortalidade, população e metadados foram obtidos de fontes governamentais.

- Transformação: As informações foram limpas, agrupadas e enriquecidas, com destaque para o cálculo das taxas de YLL (Years of Life Lost) por quadrimestre, normalizadas em relação à população.
- Carga: Os dados processados foram armazenados no BigQuery, permitindo acessos eficientes e consultas escaláveis.

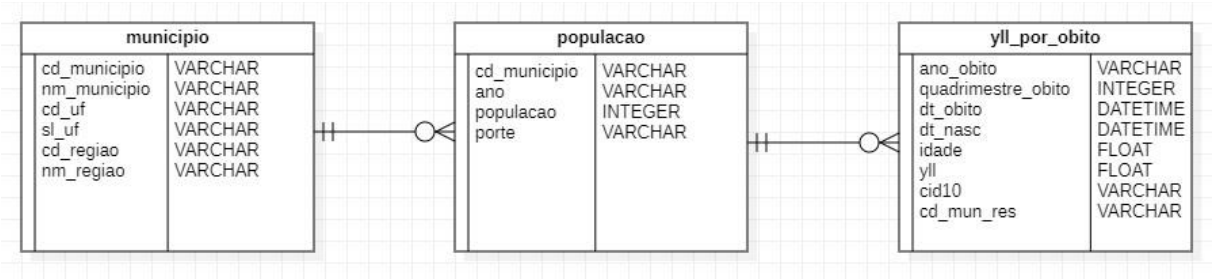
O resultado desse pipeline foi a criação de um banco de dados estruturado e pronto para o consumo em análises e modelagens de aprendizado de máquina. A tabela consolidada permite a obtenção de informações detalhadas, como taxas de YLL, estratificadas por município e quadrimestre, eliminando duplicidades e inconsistências que poderiam comprometer a qualidade dos resultados.

O impacto positivo desse pipeline reflete-se em vários aspectos:

- Eficiência: A automação reduziu o tempo de processamento e mitigou erros manuais.
- Reprodutibilidade: O pipeline pode ser facilmente replicado ou ajustado para novos períodos ou outras regiões.
- Escalabilidade: A integração com o BigQuery permite que grandes volumes de dados sejam processados rapidamente.
- Prontidão Analítica: Os dados finais oferecem uma base sólida para a aplicação de modelos preditivos, resultando em previsões confiáveis das taxas de YLL.

Além disso, o diagrama ER apresentado na Figura 1 ilustra a estrutura do banco de dados, destacando as relações entre as tabelas de mortalidade, população e YLL. Esse design foi fundamental para garantir a integridade referencial e otimizar consultas analíticas complexas.

Figura 1 - Diagrama ER das tabelas no BigQuery



Fonte: elaborado pelo autor

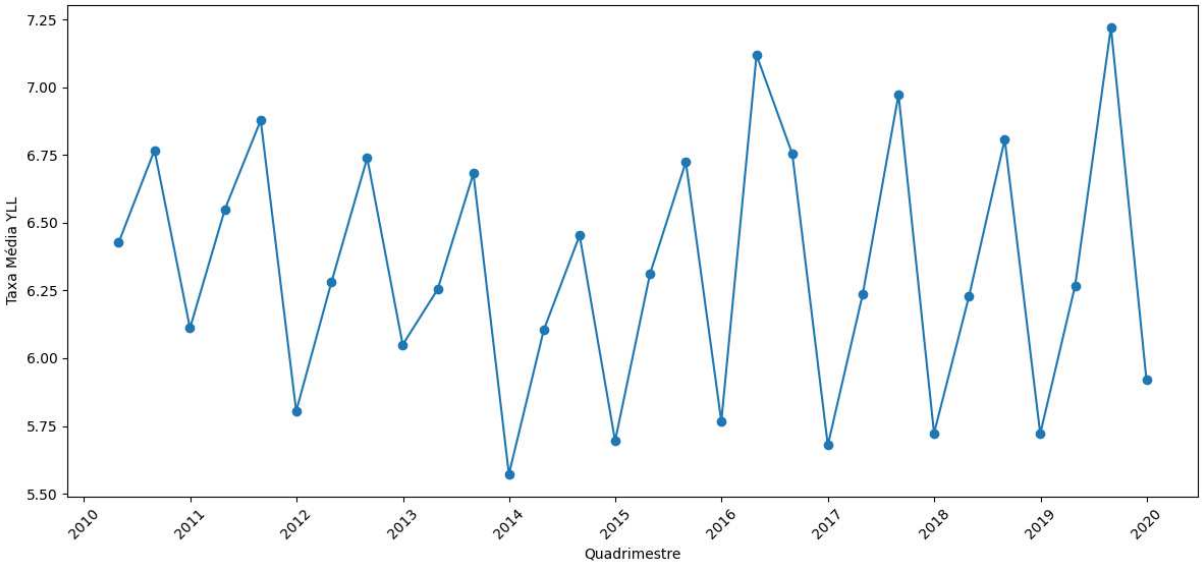
Por fim, os resultados obtidos com o pipeline de dados demonstram a excelência do processo implementado, que serviu como um alicerce para a realização das modelagens preditivas. Com a base de dados pronta, foi possível focar nos modelos de machine learning, proporcionando estimativas alinhadas com os objetivos do estudo.

5.2 ANÁLISE EXPLORATÓRIA

Para compreender o comportamento da série temporal da taxa média de Anos de Vida Perdidos (YLL) e preparar a base para modelagem preditiva, foi realizada uma análise exploratória detalhada, com a apresentação de estatísticas descritivas e visualizações gráficas dos dados. Os resultados são descritos a seguir, com a indicação de tabelas e gráficos para apoiar a discussão.

A série temporal foi inicialmente visualizada no gráfico apresentado na Figura 2, permitindo uma compreensão geral de sua estrutura. Observou-se que a série apresenta uma tendência ascendente moderada ao longo dos anos, indicando um aumento gradual no valor de YLL. Além disso, possíveis sazonalidades foram percebidas, sugerindo variações recorrentes em determinados períodos. A série temporal analisada abrange o período de abril de 2010 a dezembro de 2019, totalizando 30 pontos de observação, que representam a média quadrimestral da taxa de YLL. Esse período é longo o suficiente para capturar tendências e padrões sazonais que impactam as taxas de YLL ao longo do tempo.

Figura 2 – Série temporal da taxa média do YLL

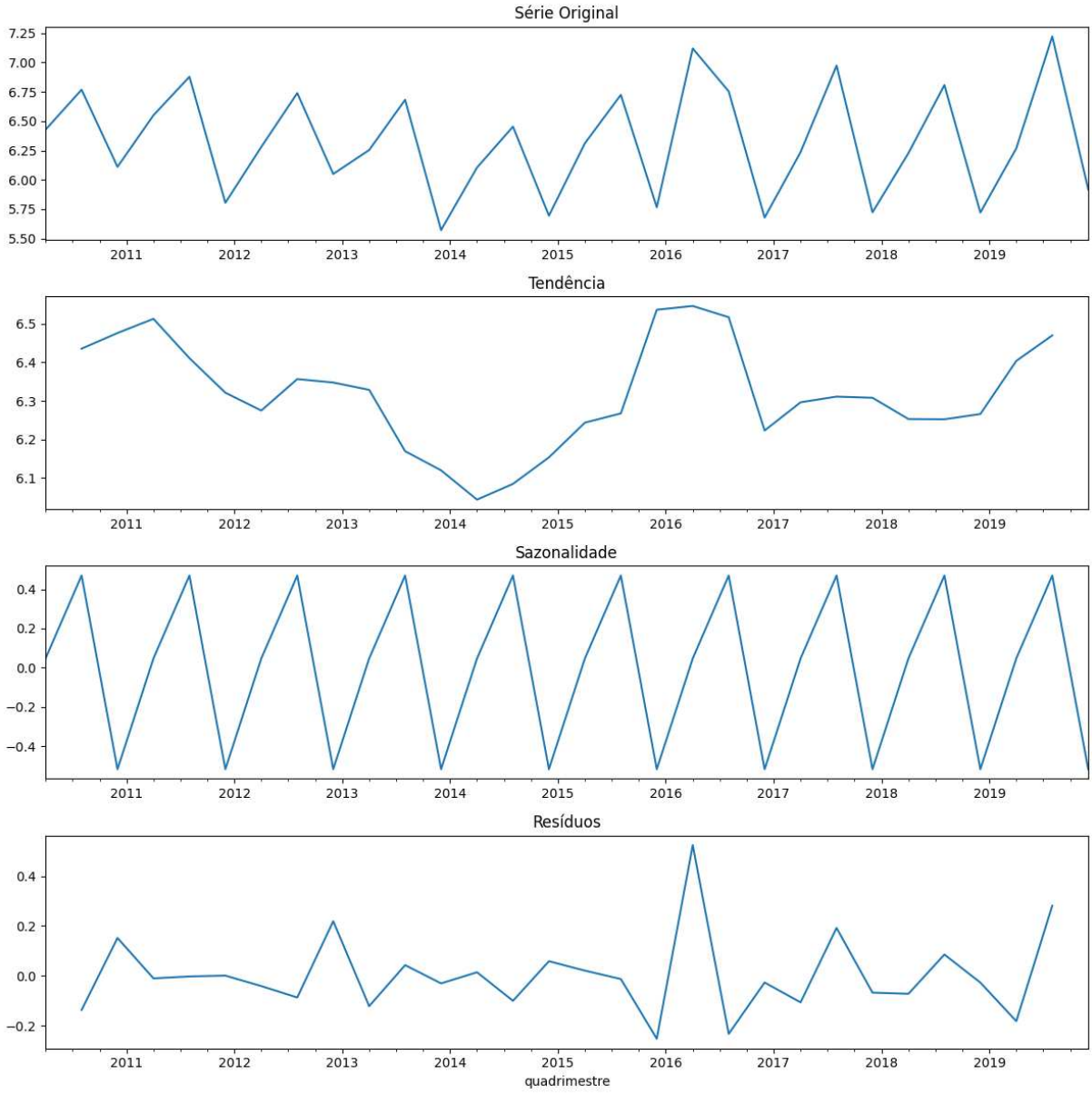


Fonte: elaborado pelo autor

Para caracterizar a distribuição dos valores da taxa média de YLL, foram calculadas diversas estatísticas descritivas. A média observada foi de 6,33, com desvio padrão de 0,47, sugerindo uma dispersão moderada ao longo dos 10 anos de dados observados. O valor mínimo registrado foi de 5,57, e o máximo foi de 7,22. Essas informações ajudam a entender a variabilidade dos dados, que pode estar associada a fatores sazonais ou eventos específicos que influenciam a saúde pública.

Para explorar as componentes de tendência e sazonalidade, a série foi decomposta em três partes: tendência, sazonalidade e resíduos. A decomposição, ilustrada na Figura 3, revelou uma tendência ascendente clara ao longo do tempo, o que reflete o aumento progressivo dos valores de YLL. A componente de sazonalidade apresentou flutuações cíclicas, indicando variações recorrentes que parecem estar associadas a fatores sazonais específicos. Por fim, a análise dos resíduos mostrou variabilidade que pode ser atribuída a fatores aleatórios, sem padrões claros.

Figura 3 – Decomposição da série temporal



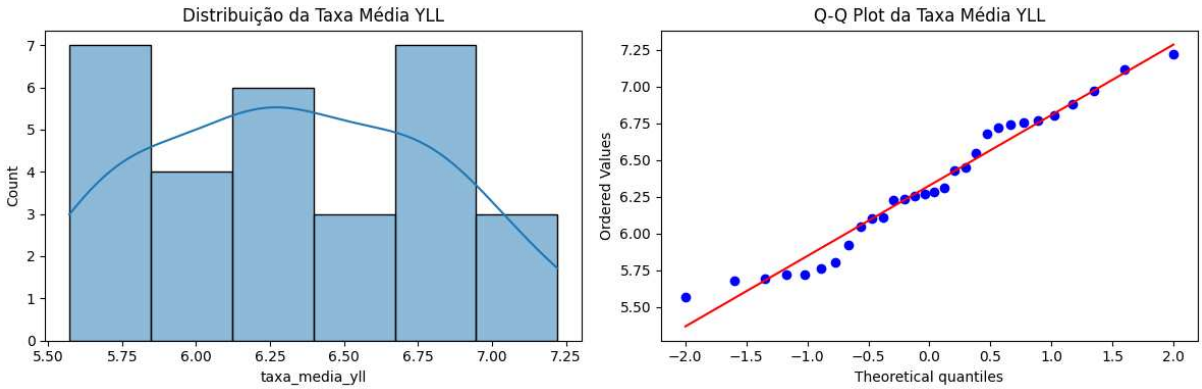
Fonte: elaborado pelo autor

Para validar a necessidade de transformações na série, foi realizado o teste de Dickey-Fuller aumentado (ADF), cujo valor de p foi superior ao nível de significância de 0,05. Esse resultado indica que a série não é estacionária, sugerindo a presença de uma tendência de longo prazo. Transformações, como diferenciação, podem ser necessárias antes da aplicação de modelos preditivos tradicionais, como ARIMA e SARIMA, para garantir a estacionariedade.

Para determinar a extensão da dependência temporal nos dados, foram analisados os gráficos de Autocorrelação (ACF) e Autocorrelação Parcial (PACF). Na Figura 4, o ACF apresenta uma queda gradual, indicando correlação significativa em múltiplos *lags*, o que é consistente com a presença de uma tendência na série. Já o PACF, na Figura 5, mostra

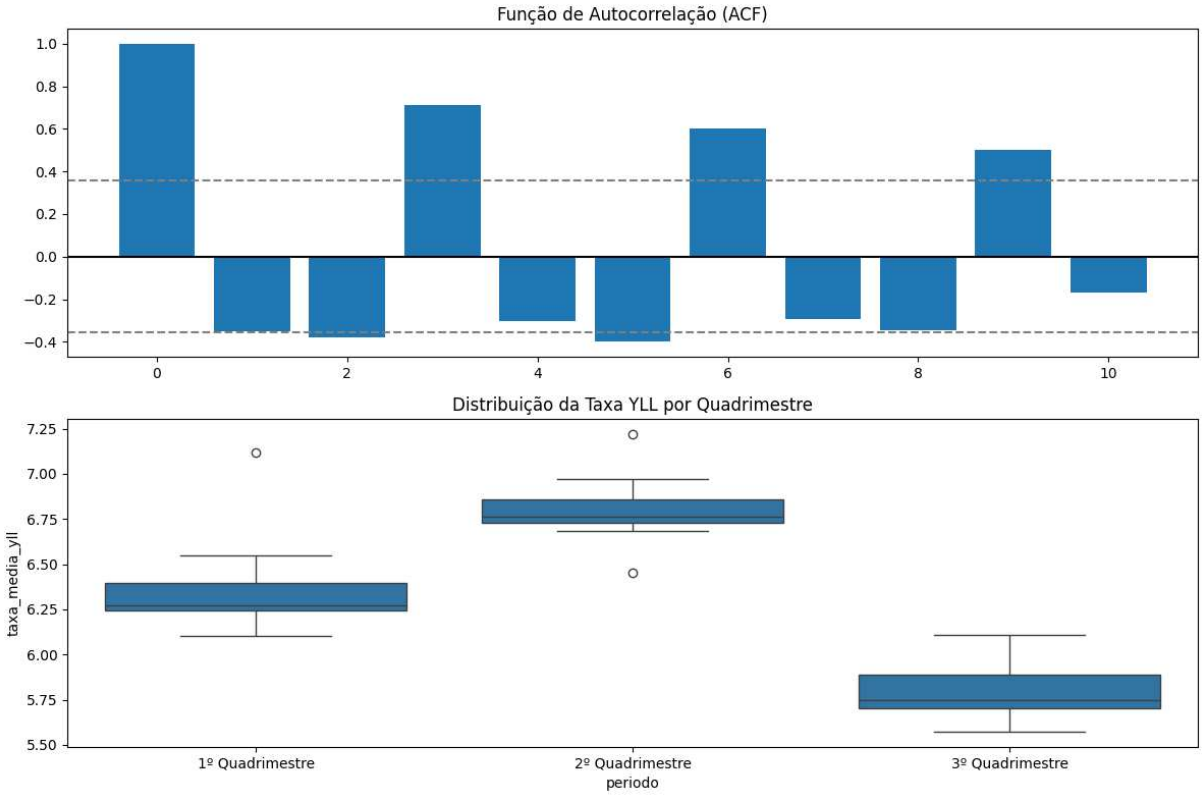
significância nos primeiros *lags*, sugerindo uma estrutura de dependência que poderia ser capturada por um modelo ARIMA ou uma rede neural LSTM, após transformações para garantir a estacionariedade.

Figura 4 – Gráfico de autocorrelação (ACF)



Fonte: elaborado pelo autor

Figura 5 – Gráfico de autocorrelação parcial (PACF)



Fonte: elaborado pelo autor

Para investigar sazonalidades mais detalhadas, a série foi analisada por quadrimestres. Essa análise revelou variações nas médias e na variabilidade das taxas de YLL

ao longo dos períodos do ano. O segundo quadrimestre apresentou a maior média (6,80) e a menor variabilidade (desvio padrão de 0,20), enquanto o terceiro quadrimestre teve a menor média (5,80). Tais flutuações sugerem a influência de fatores sazonais, que devem ser considerados na construção de modelos preditivos.

Além disso, a variação percentual entre períodos consecutivos foi analisada, apresentando uma média de 0,44% com um desvio padrão de 12,06%. A variação mínima observada foi de -18,02%, enquanto a máxima foi de 23,45%, refletindo a natureza volátil dos valores de YLL. Esse nível de volatilidade sugere que modelos de previsão que levem em consideração essas flutuações, como LSTM e Prophet, podem capturar melhor a dinâmica dos dados.

A análise exploratória da série temporal de YLL para municípios médios no Brasil forneceu informações fundamentais para o desenvolvimento de um modelo preditivo robusto. A série apresenta uma tendência ascendente com sazonalidades significativas, além de ser não-estacionária, características que demandam transformações para garantir precisão nos modelos. A presença de volatilidade e dependências temporais reforça a escolha por modelos como LSTM e Prophet, que podem capturar tendências e sazonalidades complexas.

5.3 MODELO ARIMA

Para a construção do modelo ARIMA, os dados da taxa média de Anos de Vida Perdidos (YLL) por quadrimestre foram extraídos da plataforma BigQuery e organizados em uma série temporal com média quadrimestral, abrangendo o período de abril de 2010 a dezembro de 2019. Para garantir que a série atendesse aos pressupostos do modelo ARIMA, realizou-se o teste de Dickey-Fuller Aumentado (ADF). Os resultados do teste mostraram uma estatística ADF de -2,343977 e um valor de p de 0,158176, superior ao nível de significância de 0,05. Com isso, concluiu-se que a série não era estacionária, indicando a necessidade de diferenciação para remover a tendência.

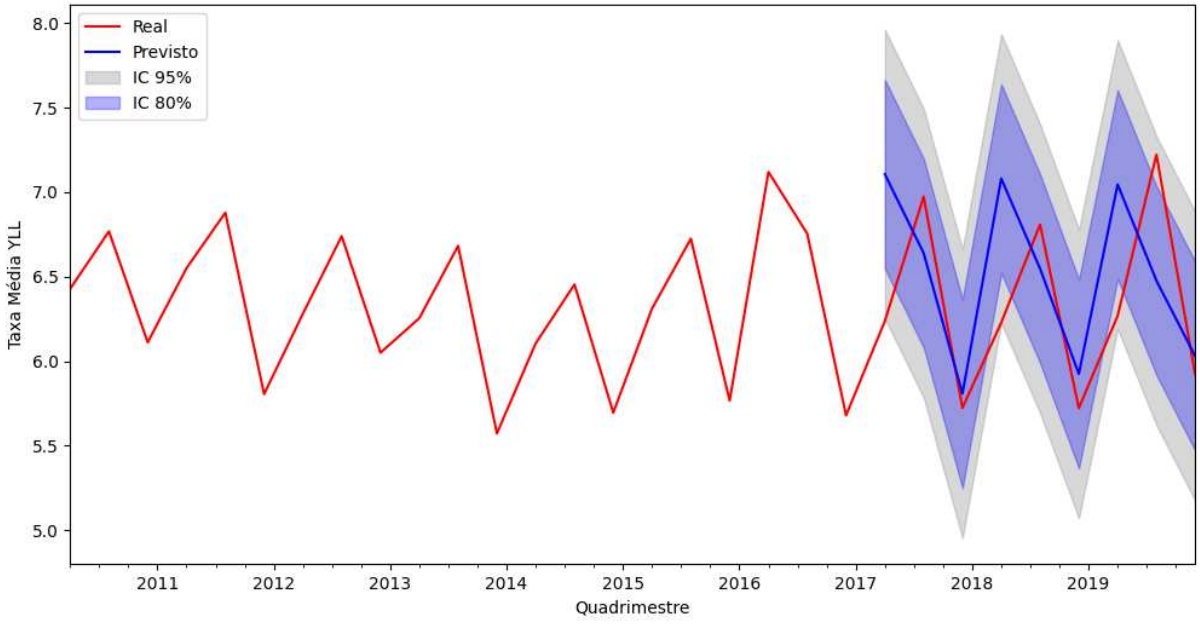
A modelagem ARIMA foi então conduzida com diferenciação de primeira ordem (d = 1) para garantir a estacionariedade da série transformada. Para selecionar os parâmetros p e q, foram analisados os gráficos de autocorrelação (ACF) e autocorrelação parcial (PACF), e, posteriormente, foi aplicado o método auto_arima, que identificou o modelo ARIMA(2,1,0)

como o melhor ajuste, com o menor valor do critério de informação AIC (AIC = 20,186). Esse modelo foi treinado com os primeiros 70% dos dados, deixando os 30% finais para a validação.

Os resultados do modelo ARIMA foram avaliados com base nas seguintes métricas de erro: o Erro Médio Absoluto (MAE) foi de 0,4706, o Erro Quadrático Médio (MSE) foi de 0,3201, e a Raiz do Erro Quadrático Médio (RMSE) foi de 0,5658. O Erro Percentual Médio Absoluto (MAPE) foi de 7,31%, indicando uma boa precisão nas previsões. Além disso, o erro de Theil (Theil's U2) foi calculado como 0,4705. Um valor de Theil's U2 próximo de 0,5 indica que o modelo apresenta precisão razoável, sendo comparável a uma previsão ingênua (como a média dos valores anteriores), mas com uma vantagem em relação a esta. O teste de Durbin-Watson apresentou um valor de 0,9314, sugerindo uma leve autocorrelação nos resíduos, mas dentro de limites aceitáveis.

Na Figura 6, observa-se a comparação entre os valores reais e previstos pelo modelo ARIMA. A linha vermelha representa os valores reais, enquanto a linha azul indica as previsões do modelo. As faixas sombreadas mostram os intervalos de confiança de 95% e 80% para as previsões. Embora o modelo ARIMA capture bem a tendência geral da série, ele apresenta limitações em capturar as flutuações sazonais, como é visível na discrepância entre os valores reais e previstos em alguns pontos. Esses resultados sugerem que o ARIMA é eficaz para capturar a tendência, mas a ausência de componentes sazonais limita sua precisão em séries temporais com sazonalidade mais acentuada.

Figura 6 – Predição da taxa média do YLL utilizando o modelo ARIMA



Fonte: elaborado pelo autor

5.4 MODELO SARIMA

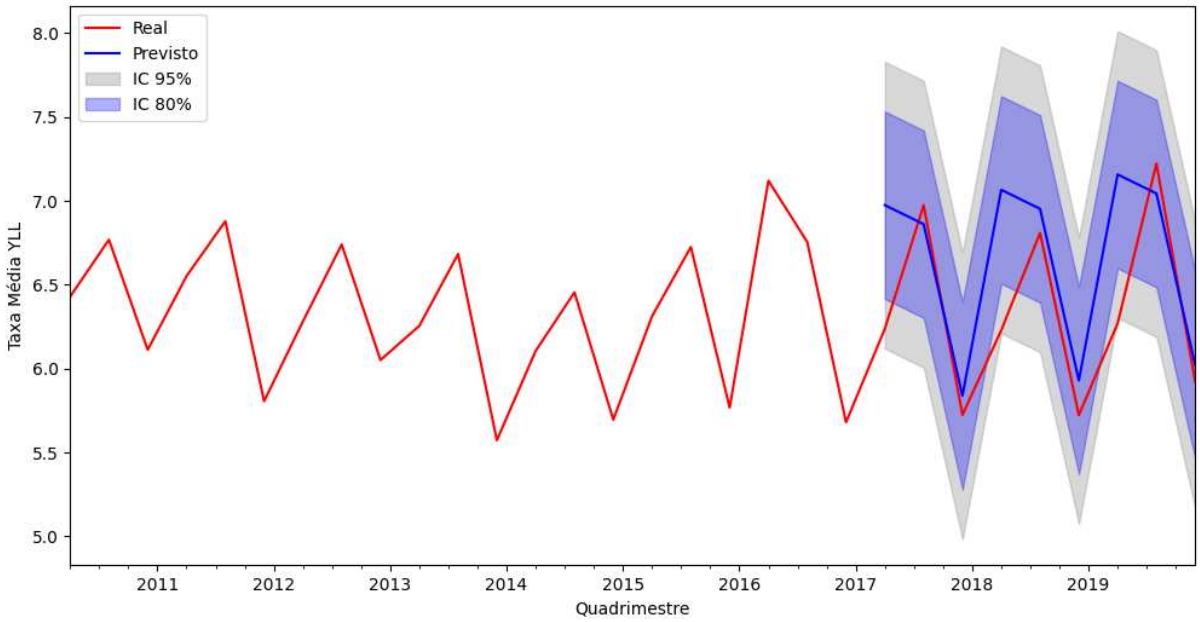
Para aprimorar a captura das variações sazonais identificadas na série, optou-se pela modelagem SARIMA, que inclui componentes sazonais. Antes de ajustar o modelo, a série foi diferenciada para garantir a estacionariedade, e o teste de Dickey-Fuller Aumentado (ADF) foi novamente aplicado. Após a diferenciação, a estatística ADF foi de -18,396229, com valor de p igual a 0,000000, confirmando que a série diferenciada era estacionária.

Para o ajuste do modelo SARIMA, foi especificada uma sazonalidade de três períodos quadrimestrais (m = 3), correspondente ao padrão de variação sazonal observado. Utilizando o método auto_arima para otimização, o modelo SARIMA(0,1,1)(0,1,1)[3] foi selecionado como o melhor ajuste, com um valor de AIC de 14,441. Assim como no ARIMA, o modelo foi treinado com os primeiros 70% dos dados e validado com os 30% restantes.

Os resultados do modelo SARIMA mostraram uma melhoria nas métricas de erro em comparação com o ARIMA. O MAE foi de 0,3687, o MSE foi de 0,2404, e o RMSE foi de 0,4903, enquanto o MAPE foi de 5,88%, indicando uma maior precisão nas previsões. O erro de Theil (Theil's U2) para o modelo SARIMA foi de 0,3934, um valor inferior ao observado no ARIMA, o que indica que o SARIMA teve melhor desempenho preditivo e foi mais eficaz na captura das variações da série. Esse valor de Theil’s U2 abaixo de 0,5 sugere que o modelo SARIMA oferece uma melhoria significativa sobre uma previsão ingênua, capturando tanto a tendência quanto a sazonalidade da série. O teste de Durbin-Watson apresentou um valor de 1,0346, próximo de 2, sugerindo a ausência de autocorrelação significativa nos resíduos e um bom ajuste do modelo aos dados.

A Figura 7 ilustra os resultados do modelo SARIMA, com a linha vermelha representando os valores reais e a linha azul as previsões. As faixas sombreadas indicam os intervalos de confiança de 95% e 80% para as previsões. O modelo SARIMA demonstrou melhor capacidade de captura das variações sazonais presentes na série, evidenciando previsões mais alinhadas com os valores reais em comparação ao ARIMA. Essa precisão é particularmente visível nas flutuações quadrimestrais, onde o SARIMA ajusta-se melhor às mudanças sazonais, como indicado pela proximidade entre as linhas vermelha e azul.

Figura 7 – Predição da taxa média do YLL utilizando o modelo SARIMA



Fonte: elaborado pelo autor

Em resumo, o modelo SARIMA mostrou-se superior ao ARIMA para a série de taxa média de YLL devido à sua capacidade de capturar tanto a tendência quanto as flutuações sazonais da série. Enquanto o ARIMA fornece previsões adequadas para séries com tendência, o SARIMA se destaca em séries com sazonalidade, oferecendo uma precisão maior nas previsões, reduzindo os erros de ajuste e apresentando um valor de Theil's U2 mais baixo, que indica um desempenho preditivo superior.

5.5 MODELO XGBOOST

Para aprimorar a precisão das previsões da taxa média de Anos de Vida Perdidos (YLL) e explorar uma abordagem baseada em aprendizado de máquina, foi utilizado o modelo XGBoost (Extreme Gradient Boosting). Este modelo é particularmente eficaz para dados estruturados e séries temporais, oferecendo robustez ao capturar padrões complexos através do ajuste de múltiplas árvores de decisão.

Inicialmente, os dados de YLL por quadrimestre foram extraídos da plataforma BigQuery e organizados em uma série temporal. Para adicionar informações sazonais ao modelo, foi realizada uma engenharia de variáveis com a criação de componentes

trigonométricos, como quadrimestre_sin e quadrimestre_cos, representando a sazonalidade anual dos quadrimestres. Essa abordagem permite que o XGBoost capture as variações periódicas inerentes ao ciclo temporal da taxa de YLL.

Para a construção do modelo, dividimos os dados em conjuntos de treino e teste, sendo que os últimos 3 anos de observações (2017-2019) foram reservados para a avaliação. Com o objetivo de otimizar os parâmetros e maximizar o desempenho preditivo, aplicou-se o GridSearchCV, um método de busca exaustiva para identificar a melhor combinação de hiperparâmetros. A grade de parâmetros incluiu ajustes para o número de estimadores (n_estimators), profundidade máxima das árvores (max_depth), taxa de aprendizado (learning_rate), amostragem das colunas (colsample_bytree), amostragem de dados (subsample), e peso mínimo de observação (min_child_weight).

Após a busca, os melhores parâmetros encontrados foram:

- colsample_bytree: 1,0
- learning_rate: 0,05
- max_depth: 3
- min_child_weight: 2
- n_estimators: 1000
- subsample: 0,6

Esses parâmetros foram selecionados por oferecerem o menor erro preditivo e maior estabilidade na série, com uma configuração que promove equilíbrio entre a capacidade de aprendizado e a redução do overfitting.

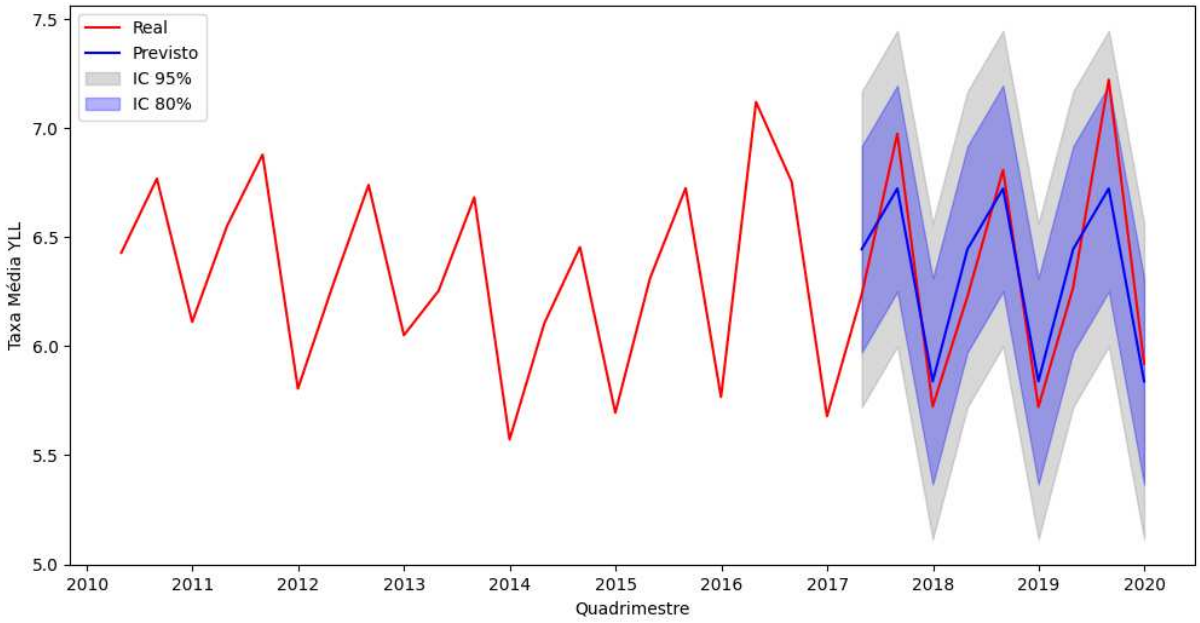
Com o modelo treinado, foram feitas previsões sobre o conjunto de teste, e a precisão do modelo foi avaliada com várias métricas de erro. O Erro Médio Absoluto (MAE) foi de 0,1942, enquanto o Erro Quadrático Médio (MSE) foi de 0,0526 e a Raiz do Erro Quadrático Médio (RMSE) foi de 0,2293. O Erro Percentual Médio Absoluto (MAPE) foi de 2,98%, indicando que o modelo alcançou uma boa precisão nas previsões. Adicionalmente, foi calculado o erro de Theil (Theil's U2), que apresentou o valor de 0,2117. Este valor indica que o modelo XGBoost oferece previsões melhores do que uma previsão ingênua (como a média móvel), capturando com eficácia as variações da série.

O teste de Durbin-Watson foi realizado para avaliar a autocorrelação residual, com um valor de 2,36, indicando uma leve autocorrelação negativa. Apesar disso, os resíduos não

mostraram uma tendência marcante, sugerindo que o modelo conseguiu ajustar-se bem à série temporal, embora ainda haja espaço para melhorias.

A Figura 8 ilustra os resultados do modelo XGBoost, com a linha vermelha representando os valores reais da taxa média de YLL, enquanto a linha azul mostra as previsões feitas pelo modelo XGBoost. Observa-se uma boa aderência entre as previsões e os valores reais, especialmente nas flutuações cíclicas ao longo do tempo, o que confirma a capacidade do modelo de capturar a sazonalidade incluída nas variáveis trigonométricas. As áreas sombreadas em cinza e azul representam os intervalos de confiança de 95% e 80%, respectivamente, mostrando a incerteza associada às previsões.

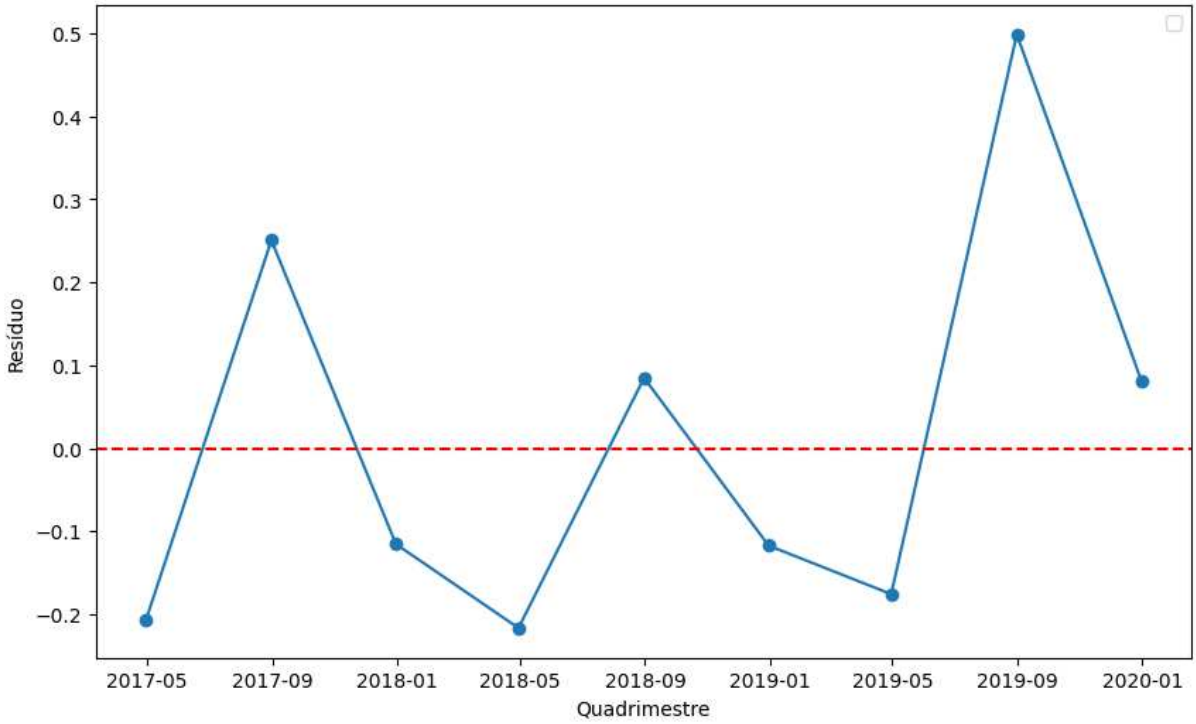
Figura 8 – Predição da taxa média do YLL utilizando o modelo XGBoost



Fonte: elaborado pelo autor

Além disso, o gráfico de resíduos (Figura 9) revela uma distribuição aproximadamente simétrica dos erros ao redor de zero, sem um padrão aparente, o que sugere que o modelo conseguiu capturar as principais tendências e sazonalidades da série. A leve dispersão observada nos resíduos para alguns períodos pode ser devida a flutuações não explicadas por variáveis sazonais ou efeitos exógenos não considerados.

Figura 9 – Análise dos resíduos do modelo XGBoost



Fonte: elaborado pelo autor

Em conclusão, o modelo XGBoost demonstrou ser uma alternativa eficiente para a previsão da taxa média de YLL, oferecendo boa precisão e capturando de forma satisfatória os padrões temporais. Comparado aos modelos ARIMA e SARIMA, o XGBoost apresentou melhores métricas de erro e menor valor de Theil's U2, sugerindo que técnicas de aprendizado de máquina podem complementar ou, em alguns casos, superar os métodos tradicionais em previsões de séries temporais complexas.

5.6 MODELO PROPHET

O modelo Prophet foi utilizado para prever a taxa média de Anos de Vida Perdidos (YLL) por quadrimestre, explorando sua capacidade de ajustar automaticamente componentes de tendência e sazonalidade. Este modelo é amplamente aplicado para séries temporais que exibem padrões sazonais e tendências não lineares, sendo particularmente eficaz para prever variações periódicas.

Os dados da série temporal foram extraídos do BigQuery, organizados em quadrimestres e adaptados ao formato requerido pelo Prophet. A divisão do conjunto de dados foi realizada, utilizando os dados até o final de 2016 para o treino e os anos seguintes (2017-2019) para validação.

Foram treinados dois modelos Prophet, cada um com diferentes intervalos de confiança: 95% e 80%. Ambos os modelos foram configurados com os seguintes parâmetros:

- changepoint_prior_scale: 0,001, controlando a sensibilidade do modelo a mudanças bruscas na tendência.
- seasonality_prior_scale: 0,1, ajustando a força aplicada às componentes sazonais.
- seasonality_mode: Additive, indicando que a sazonalidade impacta de forma fixa a série.
- n_changepoints: 5, definindo o número máximo de pontos de mudança na tendência.

Essas configurações foram selecionadas após experimentos para ajustar a complexidade do modelo ao comportamento da série temporal, capturando a sazonalidade e suavizando as flutuações bruscas. O intervalo de confiança foi ajustado para os dois modelos, permitindo a análise da incerteza preditiva em diferentes níveis.

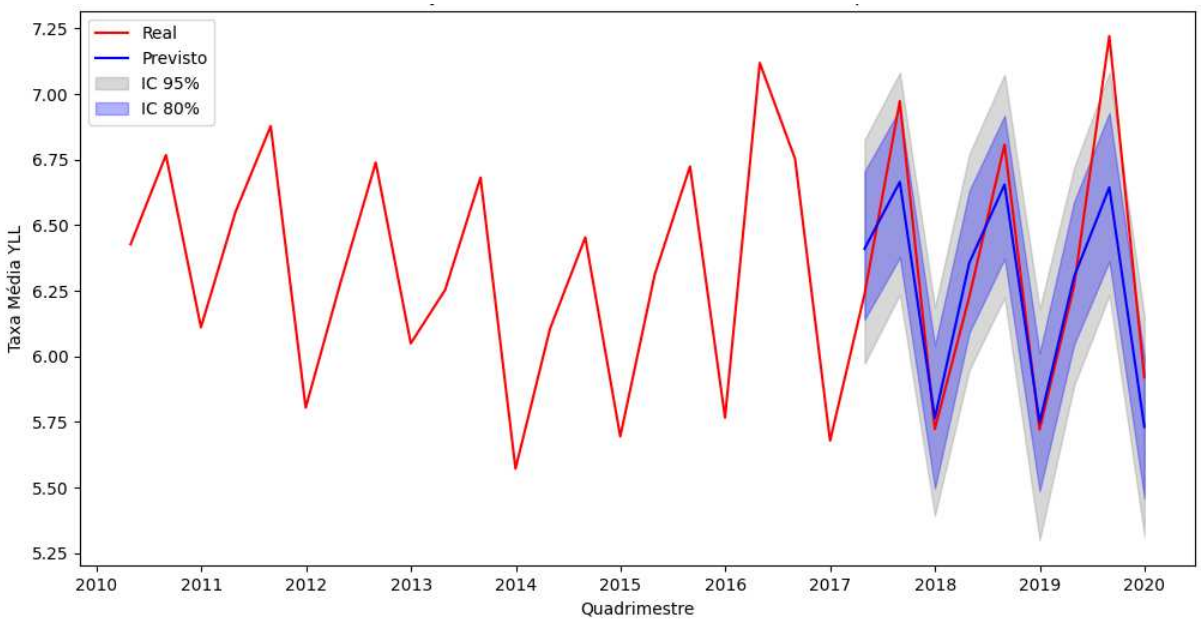
Os modelos Prophet demonstraram bom desempenho na previsão da taxa média de YLL, apresentando métricas de erro satisfatórias no conjunto de validação:

- Erro Absoluto Médio (MAE): 0,1812
- Erro Quadrático Médio (MSE): 0,0596
- Raiz do Erro Quadrático Médio (RMSE): 0,2442
- Erro Percentual Médio Absoluto (MAPE): 2,71%
- Erro de Theil (Theil's U2): 0,2236
- Durbin-Watson: 1,8584

O valor do Durbin-Watson (próximo de 2) indica uma baixa autocorrelação nos resíduos, sugerindo que o modelo se ajustou bem à série temporal. O erro de Theil, inferior a 0,3, demonstra que o Prophet apresentou previsões superiores a modelos simples, como uma média móvel.

A Figura 10 apresenta as previsões do modelo Prophet em comparação com os valores reais. A linha vermelha representa a taxa média de YLL observada, enquanto a linha azul exibe as previsões realizadas pelo modelo. Observa-se que o Prophet capturou bem os padrões de sazonalidade e tendência, exibindo uma boa aderência aos dados reais ao longo do tempo. As áreas sombreadas em cinza e azul representam os intervalos de confiança de 95% e 80%, respectivamente. Esses intervalos indicam a incerteza preditiva do modelo, sendo mais amplos para previsões futuras, como esperado. Essa característica reflete a robustez do Prophet em fornecer não apenas previsões pontuais, mas também a incerteza associada a elas.

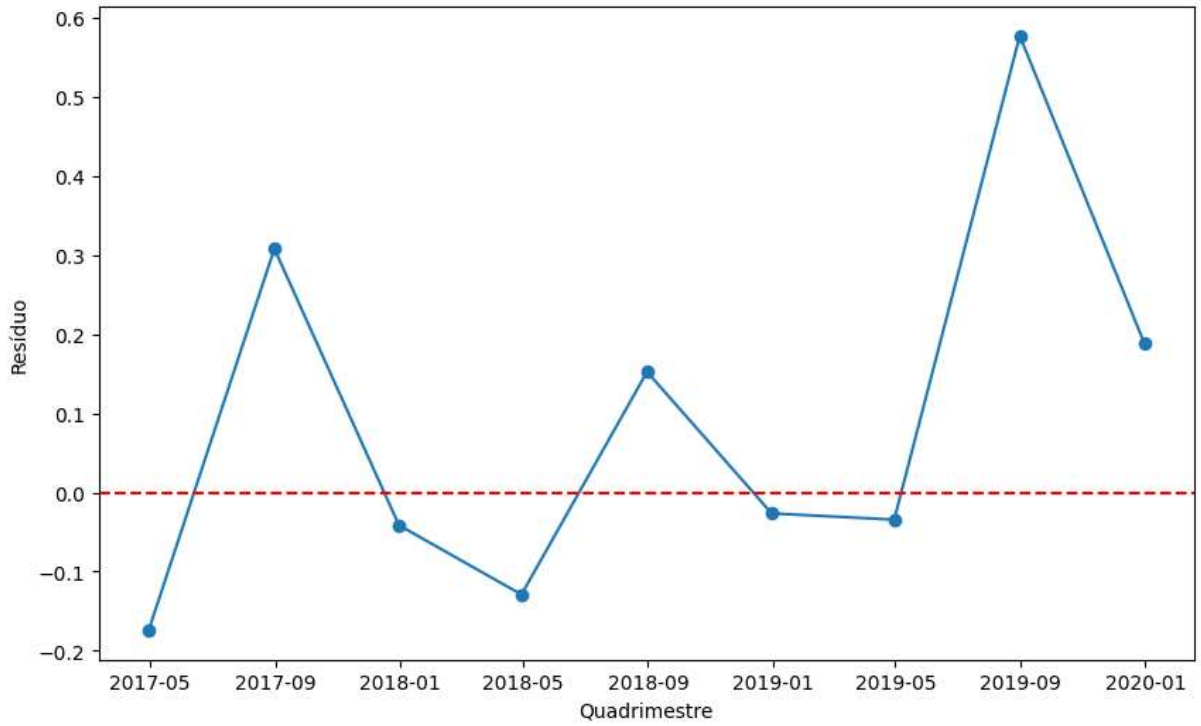
Figura 10 – Predição da taxa média do YLL utilizando o modelo Prophet



Fonte: elaborado pelo autor

Na Figura 11, a análise dos resíduos mostra uma distribuição equilibrada ao redor de zero, sem padrões aparentes, o que sugere que o modelo conseguiu capturar adequadamente as principais variações da série temporal. Pequenas discrepâncias observadas em alguns quadrimestres podem ser atribuídas a flutuações aleatórias ou fatores exógenos não considerados.

Figura 11 – Análise dos resíduos do modelo Prophet



Fonte: elaborado pelo autor

O modelo Prophet demonstrou ser uma ferramenta poderosa para a previsão da taxa média de YLL, apresentando métricas de erro competitivas e uma boa capacidade de captura de tendências e sazonalidades. Embora o Prophet tenha apresentado um desempenho ligeiramente inferior ao XGBoost em termos de MAE e MAPE, ele se destaca pela simplicidade de uso, pela transparência de seus parâmetros e pela inclusão de intervalos de confiança, tornando-o uma escolha robusta para séries temporais com padrões complexos e sazonais.

5.7 MODELO LSTM

O modelo LSTM (*Long Short-Term Memory*) é uma técnica avançada de redes neurais projetada para capturar dependências temporais e sequenciais em séries temporais, sendo ideal para lidar com dados que apresentam padrões complexos e não lineares ao longo do tempo.

Os dados de YLL foram extraídos do BigQuery e processados para atender aos requisitos do modelo LSTM. Primeiramente, os valores foram normalizados para o intervalo

[0,1] utilizando o MinMaxScaler, a fim de garantir que todas as variáveis estivessem na mesma escala e melhorar o desempenho da rede. Em seguida, os dados foram organizados em janelas deslizantes de tamanho 3 (window size), gerando entradas (x) e saídas (y) para o modelo.

O conjunto de dados foi dividido em 80% para treino, 10% para teste e 10% para validação. A rede LSTM foi projetada utilizando quatro camadas de LSTM, intercaladas com camadas Dropout para reduzir o overfitting. Cada camada LSTM foi configurada com um número específico de neurônios, conforme identificado no processo de otimização de hiperparâmetros:

- Primeira camada LSTM: 800 neurônios
- Segunda camada LSTM: 300 neurônios
- Terceira camada LSTM: 100 neurônios
- Quarta camada LSTM: 200 neurônios

A otimização de hiperparâmetros foi realizada utilizando o Keras Tuner, que testou diferentes combinações de tamanhos de camadas e taxas de Dropout. O modelo foi compilado com o otimizador Adam e a função de perda mean_squared_error, sendo treinado com Early Stopping para evitar o overfitting. O melhor modelo foi selecionado com base no menor erro quadrático médio (MSE) obtido no conjunto de validação.

O modelo LSTM demonstrou excelente desempenho, alcançando as seguintes métricas de erro:

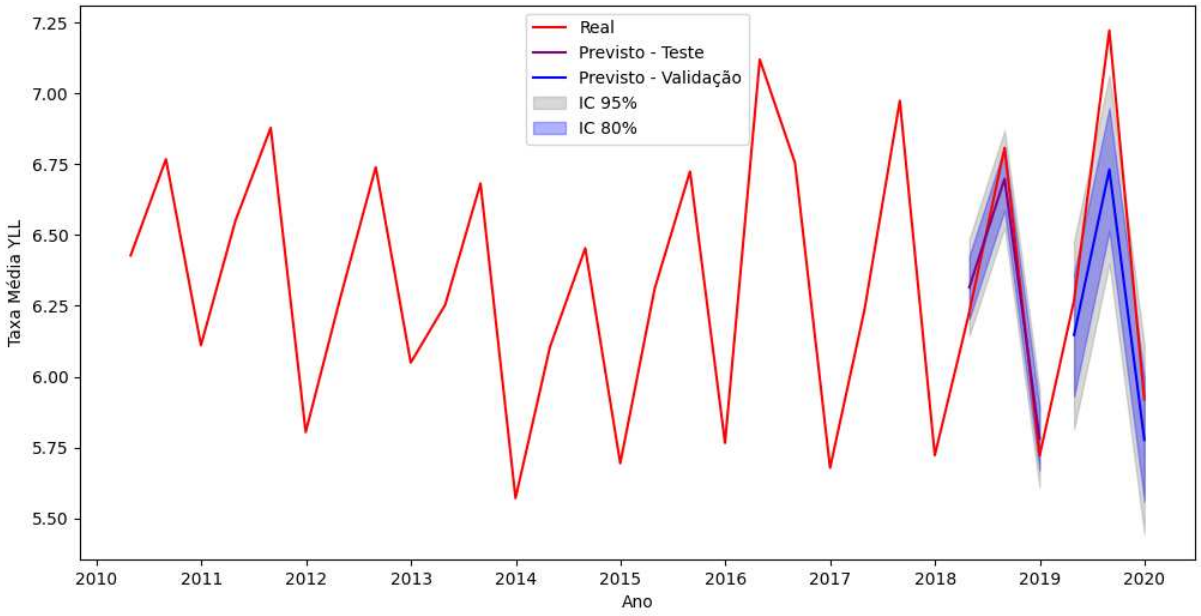
- Erro Absoluto Médio (MAE): 0.0856
- Erro Quadrático Médio (MSE): 0.0077
- Raiz do Erro Quadrático Médio (RMSE): 0.0879
- Erro Percentual Médio Absoluto (MAPE): 1.35%
- Erro de Theil (Theil's U2): 0.0873
- Durbin-Watson: 2.9048

Os valores obtidos indicam que o modelo LSTM apresentou uma capacidade de previsão muito precisa, com erros significativamente baixos. O erro de Theil próximo de zero confirma a superioridade do modelo em relação a previsões simples, como médias móveis ou persistência.

O Durbin-Watson de 2,90 sugere uma leve autocorrelação negativa nos resíduos, o que pode estar associado à natureza iterativa e estocástica do treinamento da rede neural. O fato de os resultados mudarem ligeiramente a cada execução é esperado, devido à inicialização aleatória dos pesos e à estocasticidade inerente ao processo de treinamento da LSTM.

No gráfico de previsões gerado (Figura 12), a linha vermelha representa os valores reais da taxa média de YLL, enquanto as linhas azul e roxa correspondem às previsões feitas pelo modelo LSTM para os conjuntos de teste e validação, respectivamente. Observa-se que o modelo LSTM conseguiu capturar as variações temporais da série com alta precisão, mantendo uma proximidade consistente entre os valores reais e previstos.

Figura 12 – Predição da taxa média do YLL utilizando o modelo LSTM



Fonte: elaborado pelo autor

As áreas sombreadas indicam os intervalos de confiança de 95% e 80%, calculados com base no desvio padrão dos resíduos. Esses intervalos oferecem uma visualização da incerteza associada às previsões, que é menor nos dados de teste e ligeiramente maior nos dados de validação, como esperado.

O modelo LSTM apresentou o melhor desempenho dentre todos os modelos avaliados, com métricas de erro extremamente baixas e alta aderência aos valores reais. Sua capacidade de capturar padrões complexos na série temporal torna-o uma excelente escolha para a previsão da taxa média de YLL. Apesar da variabilidade natural dos resultados devido à

estocasticidade do processo de treinamento, o modelo LSTM demonstrou ser uma abordagem robusta e eficaz para problemas preditivos em séries temporais.

5.8 COMPARAÇÃO DOS MODELOS

A avaliação de diferentes modelos para a previsão da taxa média de Anos de Vida Perdidos (YLL) permitiu identificar os pontos fortes e limitações de cada abordagem. Nesta seção, os modelos ARIMA, SARIMA, XGBoost, Prophet e LSTM são comparados, considerando suas métricas de desempenho e características preditivas.

O modelo ARIMA foi eficaz na captura da tendência geral da série temporal, mas apresentou limitações importantes na modelagem de flutuações sazonais, o que resultou em métricas de erro superiores às dos outros modelos. Embora o MAE e o MAPE tenham demonstrado que o modelo teve uma precisão razoável, o valor de Theil's U2 próximo de 0,5 indica que as previsões foram apenas ligeiramente melhores do que métodos ingênuos, como a média móvel. Ademais, o valor de Durbin-Watson de 0,93 revelou uma leve autocorrelação residual, indicando que as flutuações não foram totalmente capturadas.

Por outro lado, o SARIMA demonstrou um desempenho superior ao ARIMA devido à inclusão de componentes sazonais em sua estrutura. Este modelo conseguiu reduzir os erros preditivos significativamente, apresentando um MAPE de 5,88% e um Theil's U2 de 0,39, valores que indicam maior precisão e maior aderência aos padrões sazonais da série. O valor de Durbin-Watson de 1,03, mais próximo de 2, sugere que o modelo se ajustou bem aos dados sem introduzir autocorrelação significativa nos resíduos. No entanto, mesmo com essa melhoria, o SARIMA foi superado por modelos baseados em aprendizado de máquina em termos de métricas de erro.

O XGBoost, como modelo baseado em aprendizado de máquina, destacou-se por sua flexibilidade e robustez na captura de padrões complexos. Com o uso de variáveis trigonométricas para representar a sazonalidade, o modelo apresentou uma significativa redução nos erros, com um MAPE de 2,98% e um Theil's U2 de 0,21, indicando previsões significativamente superiores aos modelos tradicionais. Além disso, o Durbin-Watson de 2,36 mostrou que o modelo conseguiu capturar as variações temporais sem gerar padrões de autocorrelação significativos nos resíduos. No entanto, apesar de seu excelente desempenho, o

XGBoost requer maior esforço computacional e experimentação para otimização de hiperparâmetros.

O Prophet mostrou-se uma solução intermediária, combinando simplicidade de implementação com boa capacidade preditiva. O modelo apresentou métricas de erro competitivas, como um MAPE de 2,71% e um Theil's U2 de 0,22. Embora tenha um desempenho levemente inferior ao XGBoost, ele se destaca pela facilidade em capturar tendências e sazonalidades de forma automática e pela inclusão nativa de intervalos de confiança, que oferecem uma análise mais ampla da incerteza preditiva. O valor de Durbin-Watson de 1,86 sugere que o modelo se ajustou bem à série, embora possa haver uma leve autocorrelação residual.

Finalmente, o LSTM foi o modelo de melhor desempenho geral. Com um MAPE extremamente baixo de 1,35% e um Theil's U2 de apenas 0,087, o LSTM demonstrou uma capacidade superior de capturar padrões não lineares e sazonais na série temporal. Além disso, o Durbin-Watson de 2,90 revelou uma ausência quase completa de autocorrelação nos resíduos, indicando um ajuste excepcional ao comportamento dos dados. Apesar disso, a complexidade inerente às redes neurais, a variabilidade nos resultados devido à estocasticidade do processo de treinamento e o maior esforço computacional são fatores a serem considerados ao optar por este modelo.

Tabela 2 – Comparação das métricas dos modelos

Modelo	MAE	MSE	MRSE	MAPE	Theil’s U2	Durbin-Watson
ARIMA	0,4706	0,3201	0,5658	7,31%	0,4705	0,9314
SARIMA	0,3687	0,2404	0,4903	5,88%	0,3934	1,0346
XGBoost	0,1942	0,0526	0,2293	2,98%	0,2117	2,3600
Prophet	0,1812	0,0596	0,2442	2,71%	0,2236	1,8584
LSTM	0,0856	0,0077	0,0879	1,35%	0,0873	2,9048

Fonte: elaborado pelo autor

A análise comparativa revela que os modelos baseados em aprendizado de máquina, como XGBoost e LSTM, apresentaram melhor desempenho geral, com métricas de erro significativamente mais baixas do que os modelos tradicionais (ARIMA e SARIMA). O LSTM, em particular, destacou-se como o modelo mais preciso, capturando tanto tendências quanto padrões sazonais com alta eficácia. Sua capacidade de lidar com não linearidades complexas e

a baixa autocorrelação residual o tornam a escolha mais robusta para a previsão da taxa média de YLL.

No entanto, a escolha do modelo ideal também depende de fatores como simplicidade, interpretabilidade e esforço computacional. O Prophet, por exemplo, oferece um bom equilíbrio entre simplicidade e desempenho, sendo uma escolha viável para cenários onde a implementação rápida e a análise de incertezas são prioridades. Por outro lado, ARIMA e SARIMA são mais adequados para séries temporais com estruturas lineares ou quando é necessário um modelo mais facilmente interpretável.

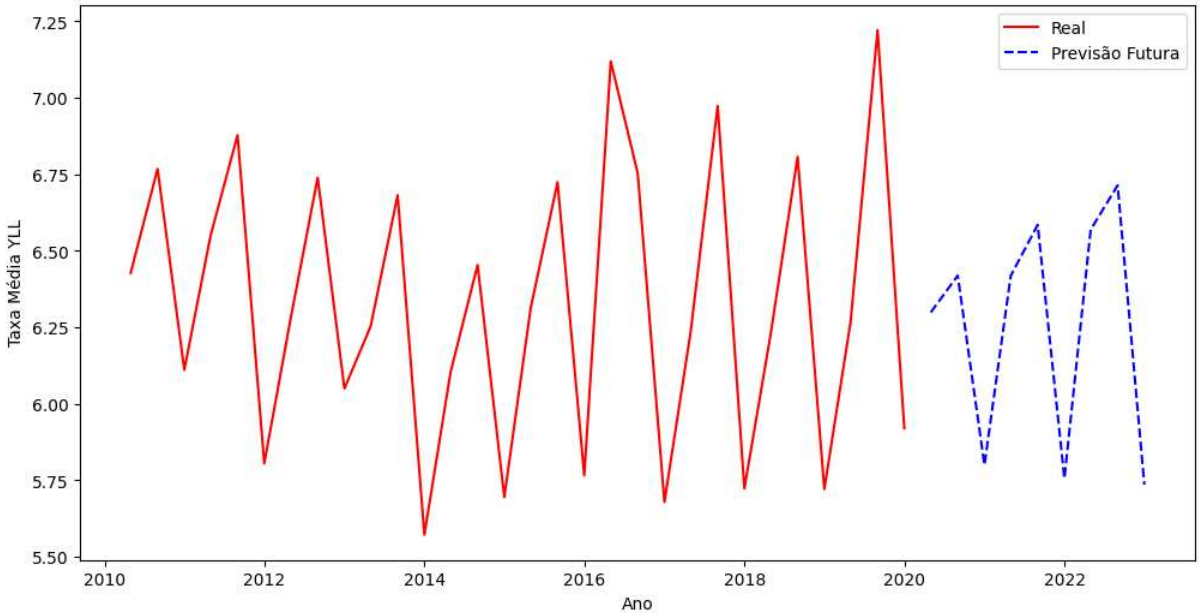
Em síntese, embora o LSTM tenha apresentado os melhores resultados, cada modelo oferece vantagens específicas que podem ser aproveitadas dependendo do objetivo e das restrições do problema em análise.

5.9 PREDIÇÃO DA TAXA MÉDIA DO YLL DE 2020 A 2022

Após identificar o modelo LSTM como a abordagem mais eficaz para previsão da taxa média de Anos de Vida Perdidos (YLL), procedeu-se à construção de previsões futuras para os próximos 9 quadrimestres, abrangendo o período de janeiro de 2020 a dezembro de 2022. O modelo foi treinado com a totalidade da série histórica (2010-2019), capturando as flutuações sazonais e a tendência da série. Este treinamento considerou apenas dados anteriores à pandemia de COVID-19, visando evitar a inclusão de padrões atípicos associados ao aumento de mortalidade registrado durante 2020 e 2022.

O gráfico gerado (Figura 13) apresenta as predições realizadas pelo modelo LSTM. As linhas vermelhas representam os valores históricos reais da taxa média de YLL até o final de 2019, enquanto as linhas azuis tracejadas indicam as previsões futuras realizadas para os 9 quadrimestres seguintes. Nota-se que o modelo projetou valores que mantêm o padrão cíclico sazonal observado nos anos anteriores, com flutuações regulares nos valores da taxa de YLL.

Figura 13 – Predição da taxa média de YLL para 2020 a 2022



Fonte: elaborado pelo autor

O comportamento projetado reflete a capacidade do modelo de capturar os padrões históricos e reproduzir flutuações sazonais em um contexto de condições normais. Isso sugere que as previsões podem ser confiáveis para cenários onde eventos externos extraordinários, como crises sanitárias, não alterem significativamente os padrões demográficos.

No entanto, é importante destacar que a decisão de utilizar o LSTM foi motivada pelos excelentes resultados obtidos durante a etapa de comparação dos modelos (Tabela 2), onde ele apresentou o melhor desempenho em métricas como MAE e RMSE. Entretanto, essa escolha foi baseada exclusivamente em critérios de desempenho, considerando o caráter acadêmico deste estudo.

Em aplicações reais, a viabilidade do LSTM deve ser cuidadosamente avaliada. Trata-se de um modelo mais complexo, que exige maior poder computacional e demanda tempo significativo para parametrização e treinamento, o que pode limitar sua utilização em contextos com restrições de recursos ou prazos reduzidos. Modelos mais simples, como ARIMA ou Prophet, podem ser alternativas mais viáveis, dependendo do cenário e das condições disponíveis.

A confiabilidade das previsões também tende a diminuir à medida que o horizonte de tempo aumenta. Isso é evidenciado pelo aumento da variabilidade projetada nas previsões finais, demonstrando a incerteza natural associada a modelos de séries temporais ao extrapolar

padrões históricos. Essa incerteza reforça a importância de reavaliar o modelo periodicamente, incorporando novos dados para atualizar as projeções e ajustá-las às condições contemporâneas.

Em conclusão, o modelo LSTM demonstrou ser uma ferramenta robusta e confiável para a previsão da taxa média de YLL em cenários de estabilidade. Apesar das limitações associadas à extrapolação e ao contexto atípico causado pela pandemia, os resultados obtidos oferecem uma base sólida para estimativas futuras e para o planejamento em saúde pública. A aplicação criteriosa do modelo, considerando tanto o desempenho quanto a viabilidade prática, é essencial para garantir resultados eficientes e sustentáveis em diferentes contextos.

6 CONCLUSÕES E TRABALHOS FUTUROS

Revisitando a motivação que deu origem a este estudo, este trabalho aborda a importância do indicador YLL (Anos de Vida Perdidos), uma métrica essencial em saúde pública que quantifica os impactos das mortes prematuras em uma população. No Brasil, um país marcado por desigualdades regionais, a previsão do YLL é crucial para uma alocação mais eficiente de recursos, além de embasar intervenções que previnam mortes evitáveis. O presente estudo destacou como o desenvolvimento de modelos preditivos pode contribuir significativamente para melhorar a qualidade de vida e auxiliar gestores públicos na tomada de decisões estratégicas.

Um dos maiores desafios enfrentados por pesquisadores na área de modelagem preditiva é a disponibilidade de dados limpos, organizados e prontos para análise. Neste contexto, uma das principais contribuições deste trabalho foi o desenvolvimento de um pipeline de dados automatizado, capaz de realizar a extração, transformação, limpeza e carga de dados de mortalidade e população de forma eficiente e reprodutível. Esse pipeline não apenas agiliza o processo de preparação dos dados, como também elimina inconsistências e reduz o esforço manual exigido na fase inicial de projetos preditivos.

O pipeline desenvolvido tem potencial de ser reutilizado e adaptado para novos estudos e outras aplicações, resolvendo uma das maiores dificuldades enfrentadas por pesquisadores: a construção de bases de dados confiáveis e de alta qualidade. Com ele, dados de múltiplas fontes podem ser integrados e estruturados, permitindo que o foco dos pesquisadores esteja na modelagem e análise dos resultados, e não na árdua tarefa de preparação dos dados. Essa contribuição representa um avanço significativo, facilitando trabalhos futuros e ampliando a capacidade de produzir modelagens robustas e aplicáveis.

Além do pipeline, o estudo comparou cinco modelos preditivos (ARIMA, SARIMA, XGBoost, Prophet e LSTM), destacando o LSTM como o modelo com melhor desempenho em termos de métricas de erro. Essa escolha, embora eficaz, deve ser analisada com cautela em contextos práticos, uma vez que o LSTM é um modelo mais complexo, que exige maior poder computacional e mais tempo para ajustes e parametrizações. Em aplicações com recursos limitados, essa complexidade pode representar um desafio adicional, exigindo uma análise crítica da viabilidade prática.

Como trabalhos futuros, sugere-se expandir o estudo para incluir municípios de outros portes, ampliando a abrangência das análises. Além disso, a incorporação de novas fontes

de dados, como indicadores socioeconômicos e climáticos, pode enriquecer as previsões e possibilitar análises mais detalhadas dos fatores que impactam o YLL. A integração do pipeline e dos modelos com sistemas automatizados de monitoramento de saúde pública também apresenta grande potencial, permitindo atualizações em tempo real das estimativas e facilitando decisões rápidas e assertivas.

A necessidade de ajustar os modelos para capturar os novos padrões de mortalidade pós-pandemia da COVID-19 também representa uma oportunidade para pesquisas futuras. A pandemia trouxe mudanças significativas, tornando fundamental a adaptação dos modelos preditivos para refletir a realidade atual e futura dos indicadores de saúde pública.

Além do uso direto em políticas de saúde pública, este projeto tem outras aplicabilidades potenciais, como o apoio a estudos epidemiológicos, a alocação eficiente de recursos em instituições de saúde e o planejamento de intervenções em áreas com maior vulnerabilidade. A capacidade do pipeline de estruturar dados confiáveis e do modelo preditivo de gerar estimativas precisas pode servir como base para o desenvolvimento de novas pesquisas e ferramentas aplicadas em contextos municipais, estaduais ou nacionais.

Em síntese, este estudo contribui significativamente para o campo da modelagem preditiva em saúde pública, evidenciando o potencial das técnicas de aprendizado de máquina na previsão de indicadores críticos. O pipeline de dados desenvolvido representa um avanço estrutural essencial, removendo barreiras históricas associadas à preparação de dados e permitindo que futuros pesquisadores acelerem suas análises. As contribuições deste trabalho abrem caminhos promissores para a pesquisa e a prática, promovendo uma gestão de saúde pública mais eficiente, informada e proativa.

REFERÊNCIAS

BHASKARAN, K.; GASPARRINI, A.; HENEGHAN, C.; ARMSTRONG, B. . **Time series regression studies in environmental epidemiology**. International Journal of Epidemiology, 2013. Acesso em: 05/10/2024. Disponível em: https://pdfs.semanticscholar.org/eab4/8c95e22f65938a0d4b590fff5704b5fef0b3.pdf.

BOX, G. E. P.; JENKINS, G. M.; REINSEL, G. C. . **Time Series Analysis: Forecasting and Control**. 5. ed. New Jersey: Wiley, 2015.

CHAI, T.; DRAXLER, R. R. . **Root mean square error (RMSE) or mean absolute error (MAE)?** – Arguments against avoiding RMSE in the literature. Geoscientific Model Development, 2014. Acesso em: 25/09/2024. Disponível em: https://gmd.copernicus.org/articles/7/1247/2014/gmd-7-1247-2014.pdf.

CHECHI, L.; BAYER, F. M. . **Modelos univariados de séries temporais para previsão das temperaturas médias mensais de Erechim, RS**. Revista Brasileira de Engenharia Agrícola e Ambiental, 2012. Acesso em 19/11/2023. Disponível em: https://doi.org/10.1590/S1415-43662012001200009.

CHENG, J.; HO, H.; WEBSTER, C. et al. . **Lower-than-standard particulate matter air pollution reduced life expectancy in Hong Kong:** A time-series analysis of 8.5 million years of life lost. Chemosphere. 2021. Acesso em 19/11/2023. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0045653521003957?via%3Dihub.

CORTES, C.; VAPNIK, V. . **Support-vector networks**. Machine Learning, 1995. Acesso em: 28/09/2024. Disponível em: https://link.springer.com/article/10.1007/BF00994018.

COSTA, M. F. S. . **Anos de vida perdidos por morte prematura:** o efeito de diferentes critérios de correção de sub-registro. Dissertação de mestrado. Rio de Janeiro, 2007. Acesso em: 11/09/2024. Disponível em: https://www.arca.fiocruz.br/handle/icict/4881.

DURBIN, J.; WATSON, G. S. . **Testing for Serial Correlation in Least Squares Regression. III**. Biometrika, 1971. Acesso em: 14/12/2024. Disponível em: https://www.jstor.org/stable/2334313.

ESTEVA, A.; KUPREL, B.; NOVOA, R. A.; KO, J.; SWETTER, S. M.; BLAU, H. M.; THRUN, S. . **Dermatologist-level classification of skin cancer with deep neural networks**. Nature, 2019. Acesso em: 28/09/2024. Disponível em: https://www.nature.com/articles/nature21056.

FRANÇA, E. B., et al. . **Cause-specific mortality for 249 causes in Brazil and states during 1990–2015:** a systematic analysis for the global burden of disease study 2015. Population Health Metrics, 2017. Acesso em: 16/09/2024 Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5700707/pdf/12963_2017_Article_156.pdf.

FUTOMA, J.; MORRIS, J.; LUCAS, J. . **A comparison of models for predicting early hospital readmissions**. Journal of the American Medical Informatics Association, 2017. Acesso em: 28/09/2024. Disponível em: https://pubmed.ncbi.nlm.nih.gov/26044081/.

GOHAIN, K. . **Evaluation of theils u:** a naïve forecast application. Faculty of Business Management and Professional (FBMP), Management and Science University, Selangor, Malaysia. 2021.

GOMEZ-CRAVIOTO, D. A. et al. . **Data Analysis and Forecasting of the COVID-19 Spread**: A Comparison of Recurrent Neural Networks and Time Series Models. Cognitive Computation, 2024. Acesso em: 13/10/2024. Disponível em: https://link-springer-com.ez46.periodicos.capes.gov.br/article/10.1007/s12559-021-09885-y.

HU, J.; HOU, J.; XU, Y. et al. . **Life loss of cardiovascular diseases per death attributable to ambient temperature:** A national time series analysis based on 364 locations in China. Science of the Total Environment. 2021. Acesso em: 05/04/2024. Disponível em: https://www-sciencedirect-com.ez46.periodicos.capes.gov.br/science/article/pii/S004896972036143X?via%3Dihub.

INSTITUTE FOR HEALTH METRICS AND EVALUATION (IHME). **Global Burden of Disease 2021:** Findings from the GBD 2021 Study. Seattle, WA: IHME, 2024. Acesso em: 16/09/2024. Disponível em: https://www.healthdata.org/sites/default/files/2024-05/GBD_2021_Booklet_FINAL_2024.05.16.pdf.

KASIEMKHAN, Sarina. **Multiple time series forecasting**: comparing Facebook's Prophet model to SARIMA. Tilburg University, 2023. Acesso em: 05/04/2023. Disponível em: https://drewhendrickson.github.io/theses/F2022_Sarina_Kasiemkhan_thesis.pdf.

LI, J.; ZHANG, X.; LI, G. et al. . **Short-term effects of ambient nitrogen dioxide on years of life lost in 48 major Chinese cities, 2013–2017**. Chemosphere. 2021. Acesso em: 05/04/2024. Disponível em: https://www-sciencedirect-com.ez46.periodicos.capes.gov.br/science/article/pii/S0045653520320828?via%3Dihub.

LITJENS, G.; KOOI, T.; BEJNORDI, B. E.; SETIO, A. A. A.; CIOMPI, F.; GHAFOORIAN, M.; VAN GINNEKEN, B. . **A survey on deep learning in medical image analysis**. Medical

image analysis, 2017. Acesso em: 28/09/2024. Disponível em: https://repository.ubn.ru.nl/bitstream/handle/2066/179538/179538.pdf?sequence=3&isAllowed=y.

MARTINEZ-PLUMED F., et al. . **CRISP-DM Twenty Years Later:** From Data Mining Processes to Data Science Trajectories. IEEE Transactions on Knowledge and Data Engineering, 2021. Acesso em: 15/09/2024. Disponível em: https://ieeexplore-ieee-org.ez46.periodicos.capes.gov.br/stamp/stamp.jsp?tp=&arnumber=8943998.

MINISTÉRIO DA SAÚDE. **Portaria nº 116, de 11 de fevereiro de 2009**. 2009. Disponível em: https://bvsms.saude.gov.br/bvs/saudelegis/svs/2009/prt0116_11_02_2009.html. Acesso em: 20/10/2023.

MINISTÉRIO DA SAÚDE. **Portaria nº 221, de 17 de abril de 2008**. 2008. Disponível em: https://bvsms.saude.gov.br/bvs/saudelegis/sas/2008/prt0221_17_04_2008.html. Acesso em: 20/10/2023.

MONTGOMERY, D. C.; PECK, E. A.; VINING, G. G. . **Introduction to Linear Regression Analysis**. 5. ed. New Jersey: Wiley, 2015.

MURRAY, C. J. L., et al. . **GBD 2010:** design, definitions, and metrics. The Lancet. 2021. Acesso em: 16/08/2024. Disponível em: https://www.thelancet.com/action/showPdf?pii=S0140-6736%2812%2961899-6.

MYTTENAEREA, A. de; GOLDEN, B.; LE GRAND, B.; ROSSIC, F. . **Mean Absolute Percentage Error for Regression Models**. Centre de Recherche en Informatique, 2017. Acesso em: 25/09/2024. Disponível em: https://arxiv.org/pdf/1605.02541.

OBERMEYER, Z.; EMANUEL, E. J. . **Predicting the future** - Big data, machine learning, and clinical medicine. The New England Journal of Medicine, 2016. Acesso em: 28/09/2024. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5070532/pdf/nihms821556.pdf.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. **CID-10:** Classificação Estatística de Doenças e Problemas Relacionados à Saúde. Décima Revisão, vol. 1, 2007.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. Department of Data and Analytics. **WHO methods and data sources for global burden of disease estimates 2000-2019**. Geneva, 2024. Acesso em: 11/09/2024. Disponível em: https://cdn.who.int/media/docs/default-source/gho-documents/global-health-estimates/ghe2019_daly-methods.pdf?sfvrsn=31b25009_7.

PALIARI, I.; KARANIKOLA, A.; KOTSIANTIS, S. . **A comparison of the optimized LSTM, XGBOOST and ARIMA in Time Series forecasting**. 2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA), Chania Crete, Greece, 2021. Acesso em: 28/09/2024. Disponível em: https://ieeexplore.ieee.org/document/9555520.

RAJKOMAR, A.; DEAN, J.; KOHANE, I. . **Machine learning in medicine**. The New England Journal of Medicine, 2018. Acesso em: 28/09/2024. Disponível em: https://med.stanford.edu/content/dam/sm/shvca/documents/july-31-ai-homework3.pdf.

RAMOS, J. L. C., et al. . **CRISP-EDM:** uma proposta de adaptação do Modelo CRISP-DM para mineração de dados educacionais. Simpósio Brasileiro de Informática na Educação, 2020, Brasil. 2020. Acesso em: 17/09/2024. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/12865/12719.

SHICKEL, B.; TIGHE, P. J.; BIHORAC, A.; RASHIDI, P. . **Deep EHR:** A survey of recent advances in deep learning techniques for electronic health record analysis. Journal of Biomedical Informatics, 2017. Acesso em: 28/09/2024. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6043423/pdf/nihms909394.pdf.

SHUMWAY, R. H.; STOFFER, D. S. . **Time Series Analysis and Its Applications**: With R Examples. 4. ed. New York: Springer, 2017.

SILVA, A. F.; ALMEIDA, A. T. C.; RAMALHO, H. M. B. . **Predição do risco de reprovação no ensino superior usando algoritmos de Machine Learning**. TEORIA E PRATICA EM ADMINISTRACAO, 2020. Acesso em: 14/09/2024. Disponível em: https://periodicos.ufpb.br/index.php/tpa/article/view/51124/30935.

SECRETARIA NACIONAL DE ASSISTÊNCIA SOCIAL. Política Nacional de Assistência Social – PNAS/2004. Norma Operacional Básica. 2004.

WANG, Y.; SUN, L. . **Energy-efficient dynamic sensor time series classification for edge health devices.** Computer Methods and Programs in Biomedicine. 2024. Acesso em: 13/10/2024. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0169260724002633.

WU, Z.; LI, J.; HUANG, J. et al. . **Ambient sulfur dioxide and years of life lost from stroke in China: a time-series analysis in 48 cities**. Chemosphere. 2021. Acesso em 19/11/2023. Disponível em: https://www-sciencedirect-com.ez46.periodicos.capes.gov.br/science/article/pii/S0045653520330551?via%3Dihub.

APÊNDICE A – CÓDIGO FONTE PIPELINE DE DADOS

O código com a estrutura completa do trabalho pode ser acessado em https://codigos.ufsc.br/marco.cesar/yll_time_series_machine_learning. Abaixo segue o código desenvolvido para a execução do pipeline de dados.

main.py

```
from pathlib import Path
from src.config import *
from src.connect import *
from src.create import *
from src.extract import *
from src.transform import *
from src.load import *

def main():

    # IDENTIFY BASE DIRECTORY
    base_dir = Path(__file__).resolve().parent.parent

    # ACCESS TO CONFIGURATION VARIABLES
    file_key, dataset, datafolder_raw, datafolder_processed, starting_year, final_year =
config(base_dir)

    # CONNECT TO GCP
    client = connect_to_gcp(file_key)

    # CREATE DATASET
    dataset_fonte = create_dataset(client,dataset)

    # CREATE TABLES
    table_yll, table_population, table_municipality = create_tables(client,dataset_fonte)

    # EXTRACT
    download_files(datafolder_raw, starting_year, final_year)

    # TRANSFORM
    population_df = create_population_df()
    municipality_df = create_municipality_df(datafolder_raw)
    yll_df = create_yll_df(datafolder_raw)

    # LOAD
    tables_dfs =
{table_population:population_df,table_municipality:municipality_df,table_yll:yll_df}
```

```
    load_data(tables_dfs,client,dataset_fonte)
    download_data(tables_dfs,datafolder_processed)

if __name__ == "__main__":
    main()
```

config.py

```
import yaml

def config(base_dir):

    # Load settings file
    with open(base_dir/'config.yaml', 'r') as file:
        config = yaml.safe_load(file)

    # Access to configuration variables
    file_key = base_dir / config['variables']['file_key']
    dataset = config['variables']['dataset']
    datafolder_raw = config['variables']['data_folder_raw']
    datafolder_processed = config['variables']['data_folder_processed']
    starting_year = config['variables']['starting_year']
    final_year = config['variables']['final_year']

    return file_key, dataset, datafolder_raw, datafolder_processed, starting_year, final_year
```

connect.py

```
from google.oauth2 import service_account
from google.cloud import bigquery

def connect_to_gcp(file_key):
    # Create connection to GCP
    print('################################################################################')
    print('#                    Iniciando execução do programa                    #')
    print('################################################################################')
    print('--------------------------------------------------------------------------------')
    print('Criando conexão com o GCP...')
    print('--------------------------------------------------------------------------------')
    try:
        credentials = service_account.Credentials.from_service_account_file(file_key)
        client = bigquery.Client(credentials=credentials, project=credentials.project_id)
        print(f'Conexão realizada com sucesso com o projeto {credentials.project_id}.')
        print('--------------------------------------------------------------------------------')
    except Exception:
```

```
        print(f'Não foi possível efetivar a conexão com o GCP.')
        print('--------------------------------------------------------------------------')
    return client
```

create.py

```
from google.cloud import bigquery

def create_dataset(client,dataset_name):
    # Create the dataset if it does not already exist
    print('--------------------------------------------------------------------------')
    print('Verificando a existência do dataset...')
    dataset_fonte = client.dataset(dataset_name)
    try:
        client.get_dataset(dataset_fonte)
        print(f'Conjunto de dados {dataset_fonte} já existe.')
        print('--------------------------------------------------------------------------')
    except Exception:
        print(f'Dataset {dataset_fonte} não encontrado, criando dataset...')
        client.create_dataset(dataset_fonte)
        print(f'Conjunto de dados {dataset_fonte} criado com sucesso.')
        print('--------------------------------------------------------------------------')
    return dataset_fonte

def create_tables(client,dataset_fonte):
    # Create tables if they do not already exist
    print('--------------------------------------------------------------------------')
    print('Verificando a existência das tabelas...')

    table_population = dataset_fonte.table('populacao')
    # Schema attributes
    schema_population = [
        bigquery.SchemaField('cd_municipio', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('ano', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('populacao', 'INTEGER', mode='REQUIRED'),
        bigquery.SchemaField('porte', 'STRING', mode='REQUIRED')
    ]

    try:
        client.get_table(table_population, timeout=30)
        print(f'A tabela {table_population} já existe!')
    except:
        print(f'Tabela {table_population} não encontrada! Criando tabela...')
        client.create_table(bigquery.Table(table_population, schema=schema_population))
        print(f'A tabela {table_population} foi criada.')

    table_municipality = dataset_fonte.table('municipio')
    # schema attributes
    schema_municipality = [
```

```
        bigquery.SchemaField('cd_municipio', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('nm_municipio', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('cd_uf', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('sl_uf', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('cd_regiao', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('nm_regiao', 'STRING', mode='REQUIRED')
    ]

    try:
        client.get_table(table_municipality, timeout=30)
        print(f'A tabela {table_municipality} já existe!')
    except:
        print(f'Tabela {table_municipality} não encontrada! Criando tabela...')
        client.create_table(bigquery.Table(table_municipality, schema=schema_municipality))
        print(f'A tabela {table_municipality} foi criada.')

    table_yll = dataset_fonte.table('yll')
    # schema attributes
    schema_yll = [
        bigquery.SchemaField('ano_obito', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('quadrimestre_obto', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('dt_obito', 'DATETIME', mode='REQUIRED'),
        bigquery.SchemaField('dt_nasc', 'DATETIME', mode='REQUIRED'),
        bigquery.SchemaField('idade', 'FLOAT', mode='REQUIRED'),
        bigquery.SchemaField('yll', 'FLOAT', mode='REQUIRED'),
        bigquery.SchemaField('cid10', 'STRING', mode='REQUIRED'),
        bigquery.SchemaField('cd_mun_res', 'STRING', mode='REQUIRED')
    ]

    try:
        client.get_table(table_yll, timeout=30)
        print(f'A tabela {table_yll} já existe!')
    except:
        print(f'Tabela {table_yll} não encontrada. Criando tabela...')
        client.create_table(bigquery.Table(table_yll, schema=schema_yll))
        print(f'A tabela {table_yll} foi criada.')

    print('-----------------------------------------------------------------------')

    return table_yll, table_population, table_municipality
```

extract.pt

```
import os
from urllib import request

def download_files(data_folder_raw, starting_year, final_year):
    # Create the directory for the data if it does not already exist
    if not os.path.exists(data_folder_raw):
```

```
        os.makedirs(data_folder_raw)

    print('------------------------------------------------------------------------')
    print('Extraindo os dados...')
    print('------------------------------------------------------------------------')

    # Download the files for the desired years
    for year in range(starting_year, final_year):
        year_to_download = str(year)

        print('------------------------------------------------------------------------')
        print(f'Proximo ano a carregar: {year_to_download}')

        # Set the link and files
        if year <= 2020:
            file_mortality = f'Mortalidade_Geral_{year_to_download}.csv'
            url_mortality = f'https://diaad.s3.sa-east-1.amazonaws.com/sim/{file_mortality}'
        elif year == 2021:
            file_mortality = f'Mortalidade_Geral_{year_to_download}.csv'
            url_mortality = f'https://S3.sa-east-
1.amazonaws.com/ckan.saude.gov.br/SIM/{file_mortality}'
        elif year >= 2022:
            reduced_year = year_to_download[-2:]
            file_mortality = f'DO{reduced_year}OPEN.csv'
            url_mortality = f'https://S3.sa-east-
1.amazonaws.com/ckan.saude.gov.br/SIM/{file_mortality}'

        # Check if the file was downloaded
        if(os.path.exists(f'{data_folder_raw}/{file_mortality}')):
            print(f'Arquivo {file_mortality} já foi baixado')
        else:
        # Try to download the file
            try:
                print(f'Baixando o arquivo {file_mortality}')
                request.urlretrieve(f'{url_mortality}', f'{data_folder_raw}/{file_mortality}')
                # Check if the file was downloaded
                if(os.path.exists(f'{data_folder_raw}/{file_mortality}')):
                    print(f'Arquivo {file_mortality} baixado')
                else:
                    print('------------------------------------------------------------------------')
                    print('Não foi possível baixar o arquivo. Execução finalizada!')
                    print('------------------------------------------------------------------------')
            except:
                print(f'Arquivos de {year_to_download} ainda não disponibilizado')

    print('------------------------------------------------------------------------')
    print(f'Extraindo os arquivos de população anual por município...')

    # Download file with IBGE codes for municipalities
    file_ibge_codes = 'municipios.csv'
```

```
    url_ibge_codes = f'https://www.gov.br/receitafederal/dados/{file_ibge_codes}'

    # Check if the file was downloaded
    if(os.path.exists(f'{data_folder_raw}/{file_ibge_codes}')):
        print(f'Arquivo {file_ibge_codes} já foi baixado')
    else:
    # Try to download the file
        try:
            print(f'Baixando o arquivo {file_ibge_codes}')
            request.urlretrieve(f'{url_ibge_codes}', f'{data_folder_raw}/{file_ibge_codes}')
            # Check if the file was downloaded
            if(os.path.exists(f'{data_folder_raw}/{file_ibge_codes}')):
                print(f'Arquivo {file_ibge_codes} baixado')
            else:
                print('--------------------------------------------------------------------------')
                print('Não foi possível baixar o arquivo. Execução finalizada!')
                print('--------------------------------------------------------------------------')
        except:
            print(f'Arquivos {file_ibge_codes} não disponibilizado')

    print('--------------------------------------------------------------------------')
    print('Todos os arquivos disponíveis foram baixados!')
    print('--------------------------------------------------------------------------')
```

transform.py

```
import os
import re
import pandas as pd

def municipality_size(population):
    # Classify the municipality by population size
    if population <= 20000:
        return 'Pequeno Porte I'
    elif population >= 20001 and population <= 50000:
        return 'Pequeno Porte II'
    elif population >= 50001 and population <= 100000:
        return 'Médio Porte'
    elif population >= 100001 and population <= 900000:
        return 'Grande Porte'
    elif population >= 900001:
        return 'Metrópole'

def create_population_df():
    print('--------------------------------------------------------------------------')
    print('Transformando dados...')
    print('--------------------------------------------------------------------------')
    print('--------------------------------------------------------------------------')
    print(f'Criando dataframe de população...')
```

```
    file_path = 'pipeline/docs'
    # Generate dataframe
    pattern = r'.*ibge_.*'
    for file in os.listdir(file_path):
        if re.match(pattern, file):
            # Create population dataframe
            df = pd.read_csv(os.path.join(file_path, file), skiprows=3, sep=';', encoding='ISO-
8859-1', low_memory=False)
            index_total = df[df['Município'] == 'Total'].index[0]
            df = df.loc[:index_total-1]
            # Creating the cod_municipio column with the first 6 characters of the Municipio
column
            df['cd_municipio'] = df['Município'].str[:6]
            # Transforming column years into rows with corresponding population values
            df = pd.melt(df, id_vars=['cd_municipio'], value_vars=[str(year) for year in
range(2010, 2022)],var_name='ano', value_name='populacao')
            # Convert 'populacao' column to integer
            df['populacao'] = pd.to_numeric(df['populacao'],
errors='coerce').fillna(0).astype(int)
            # Apply the function to define the size of the municipality
            df['porte'] = df['populacao'].apply(municipality_size)
    return df

def create_municipality_df(datafolder_raw):
    print(f'Criando dataframe de municípios...')
    file_path = datafolder_raw
    # Dictionary for region mapping
    regiao_map = {
        '1': 'Norte',
        '2': 'Nordeste',
        '3': 'Sudeste',
        '4': 'Sul',
        '5': 'Centro-Oeste'
    }
    # Generate dataframe
    pattern = 'municipios.csv'
    for file in os.listdir(file_path):
        if re.match(pattern, file):
            # Create population dataframe
            df = pd.read_csv(os.path.join(file_path, file), sep=';', encoding='ISO-8859-1',
low_memory=False)
            # Rename columns
            df = df.rename(columns={
                'CÓDIGO DO MUNICÍPIO - IBGE':'cd_municipio',
                'MUNICÍPIO - IBGE':'nm_municipio',
                'UF':'sl_uf'
                })
            # Convert to string
            df['cd_municipio'] = df['cd_municipio'].astype(str)
            df['cd_municipio'] = df['cd_municipio'].str[:6]
```

```
        df['cd_uf'] = df['cd_municipio'].str[:2]
        # Create the region column by extracting the first digit of id_municipio
        df['cd_regiao'] = df['cd_municipio'].str[0]
        # Map nm_region using the regiao_map dictionary
        df['nm_regiao'] = df['cd_regiao'].map(regiao_map)
        # Select desired columns
        df = df[['cd_regiao', 'nm_regiao', 'cd_uf', 'sl_uf', 'cd_municipio', 'nm_municipio']]
    return df

def calculate_life_expectancy(age):
    # Calculate life expectancy
    if age < 0.08:
        return 89.99
    elif 0.08 <= age < 1:
        return 89.55
    elif 1 <= age < 5:
        return 89.07
    elif 5 <= age < 10:
        return 82.58
    elif 10 <= age < 15:
        return 77.58
    elif 15 <= age < 20:
        return 72.60
    elif 20 <= age < 25:
        return 67.62
    elif 25 <= age < 30:
        return 62.66
    elif 30 <= age < 35:
        return 57.71
    elif 35 <= age < 40:
        return 52.76
    elif 40 <= age < 45:
        return 47.83
    elif 45 <= age < 50:
        return 42.94
    elif 50 <= age < 55:
        return 38.10
    elif 55 <= age < 60:
        return 33.33
    elif 60 <= age < 65:
        return 28.66
    elif 65 <= age < 70:
        return 24.12
    elif 70 <= age < 75:
        return 19.76
    elif 75 <= age < 80:
        return 15.65
    elif 80 <= age < 85:
        return 11.69
    else:
```

```
        return 7.05

def create_yll_df(datafolder_raw):
    file_path = datafolder_raw
    # List of ICD-10 codes belonging to ICSAPS
    cod_icsaps =
['A37','A36','A33','A34','A35','B26','B06','B05','A95','B16','G000','A170','A19','A150','A151'
,'A152','A153','A160',
                'A161','A162','A154','A155','A156','A157','A158','A159','A163','A164','A165','A
166','A167','A168','A169','A171','A172',
                'A173','A174','A175','A176','A177','A178','A179','A18','I00','I01','I02','A51','A52
','A53','B50','B51','B52','B53',
                'B54','B77','E86','A00','A01','A02','A03','A04','A05','A06','A07','A08','A09','D50'
,'E40','E41','E42','E43','E44','E45',
                'E46','E50','E51','E52','E53','E54','E55','E56','E57','E58','E59','E60','E61','E62','E
63','E64','H66','J00','J01','J02',
                'J03','J06','J31','J13','J14','J153','J154','J158','J159','J181','J45','J46','J20','J21','J40'
,'J41','J42','J43','J47',
                'J44','I10','I11','I20','I50','J81','I63','I64','I65','I66','I67','I69','G45','G46','E100','E1
01','E110','E111','E120',
                'E121','E130','E131','E140','E141','E102','E103','E104','E105','E106','E107','E108
','E112','E113','E114','E115','E116',
                'E117','E118','E122','E123','E124','E125','E126','E127','E128','E132','E133','E134
','E135','E136','E137','E138','E142',
                'E143','E144','E145','E146','E147','E148','E109','E119','E129','E139','E149','G40',
'G41','N10','N11','N12','N30','N34',
                'N390','A46','L01','L02','L03','L04','L08','N70','N71','N72','N73','N75','N76','K25'
,'K26','K27','K28','K920','K921',
                'K922','O23','A50','P350']
    # List with the dataframes already processed
    dfs = []

    print('Criando dataframe de yll...')
    # Generate the dataframe
    pattern = r'^(Mortalidade_Geral_\d{4}\.csv|DO\d{2}OPEN\.csv)$'
    for file in os.listdir(file_path):
        if re.match(pattern, file):
            # Read CSV file with Pandas
            df = pd.read_csv(os.path.join(file_path, file), delimiter=';', encoding='ISO-8859-1',
low_memory=False)
            # Analyze whether the ICD-10 code belongs to ICSAPs
            df['icsaps'] = df['CAUSABAS'].apply(lambda x: 'Sim' if x in cod_icsaps else 'Não')
            # Keep only data that is ICSAPs
            df = df[df['icsaps'] == 'Sim']
            # Perform transformation of birth and death dates
            df['dt_obito'] = pd.to_datetime(df['DTOBITO'], format='%d%m%Y',
errors='coerce')
            df['dt_nasc'] = pd.to_datetime(df['DTNASC'], format='%d%m%Y', errors='coerce')
            # Delete null data for date of birth and date of death
            df = df.dropna(subset=['dt_nasc'])
```

```
            df = df.dropna(subset=['dt_obito'])
            # Create the age column
            df['idade'] = ((df['dt_obito'] - df['dt_nasc']).dt.days / 365.25).round(2)
            # Keep only data with valid ages
            df = df[df['idade'] >= 0]
            # Create column yll
            df['yll'] = df.apply(lambda row: calculate_life_expectancy(row['idade']), axis=1)
            # Create the columns ano_obito and quadrimestre_obto
            df['ano_obito'] =
df['dt_obito'].dt.year.astype(float).astype(pd.Int64Dtype()).astype(str).where(df['dt_obito'].
notna())
            df['quadrimestre_obto'] = pd.cut(df['dt_obito'].dt.month, bins=[1, 5, 9, 13],
labels=[1, 2, 3], right=False)
            df['quadrimestre_obto'] = df['quadrimestre_obto'].astype('object')
            # Extract the first 6 digits from the CODMUNRES column
            df['cd_mun_res'] = df['CODMUNRES'].astype(str).str.slice(stop=6)
            # Rename columns
            df = df.rename(columns={'CAUSABAS':'cid10'})
            # Select desired columns
            df =
df[['ano_obito','quadrimestre_obto','dt_obito','dt_nasc','idade','yll','cid10','cd_mun_res']]
            # Add the dataframe to the list of dataframes
            dfs.append(df)
    # Concatenate the dataframes into a single final dataframe
    df_group = pd.concat(dfs, ignore_index=True)
    print('--------------------------------------------------------------------------')
    print('Dataframes gerados com sucesso.')
    print('--------------------------------------------------------------------------')
    return df_group
```

load.py

```
from google.cloud import bigquery
import os

def load_data(tables_dfs,client,dataset_fonte):
    # Load data into gcp
    print('--------------------------------------------------------------------------')
    print('Carregando dados no GCP...')
    print('--------------------------------------------------------------------------')
    for tabela, df in tables_dfs.items():
        table_ref = client.dataset(dataset_fonte.dataset_id).table(tabela.table_id)
        job_config = bigquery.LoadJobConfig()
        job_config.write_disposition = bigquery.WriteDisposition.WRITE_TRUNCATE
        job = client.load_table_from_dataframe(df, table_ref, job_config=job_config)
        job.result()
        print(f'Dados carregados na tabela {tabela}.')

    print('--------------------------------------------------------------------------')
```

```
def download_data(tables_dfs,datafolder_processed):
    print('------------------------------------------------------------------------')
    print(f'Baixando os dados em .CSV para a pasta {datafolder_processed}...')
    print('------------------------------------------------------------------------')
    # Create the directory for the data if it does not already exist
    if not os.path.exists(datafolder_processed):
        os.makedirs(datafolder_processed)
    # Dowload data
    for table, df in tables_dfs.items():
        df.to_csv(f'{datafolder_processed}/{table.table_id}.csv', index=False)
        print(f'Os dados foram baixados para o arquivo {table.table_id}.csv...')
    print('------------------------------------------------------------------------')
    print('########################################################################
########')
    print('#                       Execução Finalizada                        #')
    print('########################################################################
########')
```

APÊNDICE B – CÓDIGOS FONTE MODELOS PREDITIVOS

O código com a estrutura completa do trabalho pode ser acessado em https://codigos.ufsc.br/marco.cesar/yll_time_series_machine_learning. Abaixo segue o código desenvolvido para a execução dos algoritmos de *machine learning* testados.

exploratory_analysis.py

```
from google.cloud import bigquery
from google.oauth2 import service_account
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import acf
import scipy.stats as stats

# Realizar conexão com o GCP
credentials = service_account.Credentials.from_service_account_file("keys/ml-na-saude-
ed1fc3c1a83e.json")
client = bigquery.Client(credentials=credentials, project=credentials.project_id)

# Query para consulta dos dados agrupados por taxa média por quadrimestre
consulta_sql = """
select
   tx.quadrimestre,
   avg(tx.taxa_yll) as taxa_media_yll
from (
   with yll_quadrimestral as (
      select
         case
            when extract(month from y.dt_obito) between 1 and 4 then date(extract(year from
y.dt_obito), 4, 30)
            when extract(month from y.dt_obito) between 5 and 8 then date(extract(year from
y.dt_obito), 8, 31)
            when extract(month from y.dt_obito) between 9 and 12 then date(extract(year
from y.dt_obito), 12, 31)
            end as quadrimestre,
         m.nm_municipio,
         p.populacao,
         sum(y.yll) as soma_yll
      from `ml-na-saude.yll_por_obito.yll` y
```

```
        join `ml-na-saude.yll_por_obito.populacao` p on y.cd_mun_res = p.cd_municipio and
y.ano_obito = p.ano
        join `ml-na-saude.yll_por_obito.municipio` m on p.cd_municipio = m.cd_municipio
        where p.porte = 'Médio Porte'
        group by 1,2,3
    )
    select
        quadrimestre,
        nm_municipio,
        soma_yll,
        populacao,
        soma_yll / populacao * 1000 as taxa_yll
    from yll_quadrimestral
    group by 1,2,3,4
) tx
group by 1
order by 1
"""

# Ignorar avisos e gerar dataframe
warnings.simplefilter("ignore")
df = client.query(consulta_sql).to_dataframe()
# Copiar dataframe para manipular dados
time_series = df.copy()
# Transformar o quadrimestre em data
time_series['quadrimestre'] = pd.to_datetime(time_series['quadrimestre'])
# Filtrar dados até final de 2019
time_series = time_series[time_series['quadrimestre'] <= '2019-12-31']

def analise_exploratoria_temporal(time_series):
    """
    Realiza uma análise exploratória completa de uma série temporal

    Parâmetros:
    time_series: DataFrame com colunas 'quadrimestre' e 'taxa_media_yll'
    """

    # Configurações visuais
    plt.rcParams['figure.figsize'] = [12, 8]

    # 1. Estatísticas Descritivas Básicas
    print("1. Estatísticas Descritivas da Taxa Média YLL:")
    print(time_series['taxa_media_yll'].describe())
    print("\nPeríodo da série:")
    print(f"Início: {time_series['quadrimestre'].min()}")
    print(f"Fim: {time_series['quadrimestre'].max()}")
    print(f"Total de períodos: {len(time_series)}")

    # 2. Visualização da Série Temporal
    plt.figure(figsize=(12, 6))
```

```
    plt.plot(time_series['quadrimestre'], time_series['taxa_media_yll'], marker='o')
    plt.title('Evolução da Taxa Média YLL ao Longo do Tempo')
    plt.xlabel('Quadrimestre')
    plt.ylabel('Taxa Média YLL')
    plt.grid(True)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

    # 3. Decomposição da Série Temporal
    # Definir o índice como data para usar o seasonal_decompose
    ts = time_series.set_index('quadrimestre')['taxa_media_yll']

    # Realizar decomposição
    decomposicao = seasonal_decompose(ts, period=3)  # period=3 para dados
quadrimestrais

    # Plotar decomposição
    fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(12, 12))

    decomposicao.observed.plot(ax=ax1)
    ax1.set_title('Série Original')
    ax1.set_xlabel('')

    decomposicao.trend.plot(ax=ax2)
    ax2.set_title('Tendência')
    ax2.set_xlabel('')

    decomposicao.seasonal.plot(ax=ax3)
    ax3.set_title('Sazonalidade')
    ax3.set_xlabel('')

    decomposicao.resid.plot(ax=ax4)
    ax4.set_title('Resíduos')

    plt.tight_layout()
    plt.show()

    # 4. Análise de Distribuição
    plt.figure(figsize=(12, 4))

    # Histograma
    plt.subplot(121)
    sns.histplot(data=time_series, x='taxa_media_yll', kde=True)
    plt.title('Distribuição da Taxa Média YLL')

    # QQ Plot
    plt.subplot(122)
    stats.probplot(time_series['taxa_media_yll'], dist="norm", plot=plt)
    plt.title('Q-Q Plot da Taxa Média YLL')
```

```
    plt.tight_layout()
    plt.show()

    # 5. Análise de Autocorrelação
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

    # ACF
    acf_values = acf(time_series['taxa_media_yll'], nlags=10)
    ax1.bar(range(len(acf_values)), acf_values)
    ax1.axhline(y=0, linestyle='-', color='black')
    ax1.axhline(y=-1.96/np.sqrt(len(time_series)), linestyle='--', color='gray')
    ax1.axhline(y=1.96/np.sqrt(len(time_series)), linestyle='--', color='gray')
    ax1.set_title('Função de Autocorrelação (ACF)')

    # Box plot por período do ano
    time_series['periodo'] = time_series['quadrimestre'].dt.month.map({
        4: '1º Quadrimestre',
        8: '2º Quadrimestre',
        12: '3º Quadrimestre'
    })
    sns.boxplot(data=time_series, x='periodo', y='taxa_media_yll', ax=ax2)
    ax2.set_title('Distribuição da Taxa YLL por Quadrimestre')

    plt.tight_layout()
    plt.show()

    # 6. Estatísticas por período
    print("\nEstatísticas por Quadrimestre:")
    print(time_series.groupby('periodo')['taxa_media_yll'].describe())

    # 7. Análise de Variação
    time_series['variacao'] = time_series['taxa_media_yll'].pct_change() * 100

    print("\nEstatísticas das Variações Percentuais:")
    print(time_series['variacao'].describe())

    plt.figure(figsize=(12, 6))
    plt.plot(time_series['quadrimestre'], time_series['variacao'], marker='o')
    plt.title('Variação Percentual da Taxa YLL entre Períodos')
    plt.xlabel('Quadrimestre')
    plt.ylabel('Variação Percentual (%)')
    plt.grid(True)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

# Executar a análise
analise_exploratoria_temporal(time_series)
```

arima_e_sarima_model.py

```
from google.cloud import bigquery
from google.oauth2 import service_account
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.stats.stattools import durbin_watson
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error
import scipy.stats as stats
from pmdarima.arima import auto_arima

# Realizar conexão com o GCP
credentials = service_account.Credentials.from_service_account_file("../keys/ml-na-saude-
ed1fc3c1a83e.json")
client = bigquery.Client(credentials=credentials, project=credentials.project_id)

# Query para consulta dos dados agrupados por taxa média por quadrimestre
consulta_sql = """
select
   tx.quadrimestre,
   avg(tx.taxa_yll) as taxa_media_yll
from (
   with yll_quadrimestral as (
      select
         case
            when extract(month from y.dt_obito) between 1 and 4 then date(extract(year from
y.dt_obito), 4, 30)
            when extract(month from y.dt_obito) between 5 and 8 then date(extract(year from
y.dt_obito), 8, 31)
            when extract(month from y.dt_obito) between 9 and 12 then date(extract(year
from y.dt_obito), 12, 31)
            end as quadrimestre,
         m.nm_municipio,
         p.populacao,
         sum(y.yll) as soma_yll
      from `ml-na-saude.yll_por_obito.yll` y
      join `ml-na-saude.yll_por_obito.populacao` p on y.cd_mun_res = p.cd_municipio and
y.ano_obito = p.ano
      join `ml-na-saude.yll_por_obito.municipio` m on p.cd_municipio = m.cd_municipio
      where p.porte = 'Médio Porte'
      group by 1,2,3
   )
   select
      quadrimestre,
```

```
        nm_municipio,
        soma_yll,
        populacao,
        soma_yll / populacao * 1000 as taxa_yll
    from yll_quadrimestral
    group by 1,2,3,4
) tx
group by 1
order by 1
"""

# Ignorar avisos e gerar dataframe
warnings.simplefilter("ignore")
df = client.query(consulta_sql).to_dataframe()
# Copiar dataframe para manipular dados
time_series = df.copy()
# Transformando o quadrimestre em data
time_series['quadrimestre'] = pd.to_datetime(time_series['quadrimestre'])
# Filtrar dados até final de 2019
time_series = time_series[time_series['quadrimestre'] <= '2019-12-31']
# Setando o quadrimestre como índice da tabela
time_series = time_series.set_index('quadrimestre')

### Decomposição da série

# Plotar decomposição da série temporal
result = seasonal_decompose(time_series, model='additive', period=3)
result.plot()
plt.show()

### Teste de Estacionariedade

# Função para testar a estacionaridade
def teste_adf(serie):
    result = adfuller(serie)
    print('ADF Estatíticas: %f' % result[0])
    print('Valor de P: %f' % result[1])
    print('Valores Críticos:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))

    if result[1] < 0.05:
        print("A série é estacionária.")
    else:
        print("A série não é estacionária.")

# Executa o teste de estacionaridade em 'taxa_media_yll'
X = time_series['taxa_media_yll']
teste_adf(X)
```

```
plt.figure(figsize=(12, 6))
X.plot()

### Tornando a série estacionária com diferenciação simples

# Diferenciação simples
xdiff = X.diff().dropna()
xlabel='Data'
plt.figure(figsize=(12, 6))
xdiff.plot()

# Verifica novamente a estacionaridade após a diferenciação
teste_adf(xdiff)

### ARIMA

X = time_series['taxa_media_yll']

arima_model = auto_arima(X,
                         start_p=1,
                         start_q=1,
                         max_p=6,
                         max_q=6,
                         seasonal=False,  # Definindo como False para um modelo ARIMA
                         d=1,
                         D=1,
                         trace=True,
                         error_action='ignore',
                         suppress_warnings=True,
                         stepwise=True)

train = X.loc[:'2016-12-31']
test = X.loc['2017-01-01':]

arima_model.fit(train)

future_forecast_arima = arima_model.predict(n_periods=9)

future_forecast_arima.index

future_forecast_arima.plot(marker='', color='blue', legend=True, label='Previsto')
test.plot(marker='', color='red', label='Real', legend=True)
plt.show()

future_forecast_arima.plot(marker='', color='blue', legend=True, label='Previsto')
X.plot(marker='', color='red', label='Real', legend=True)
plt.show()

# Parâmetros para calcular os intervalos de confiança
```

```
alpha_95 = 0.05 # Nível de significância para intervalo de confiança de 95%
alpha_80 = 0.2 # Nível de significância para intervalo de confiança de 80%

# Valor crítico para distribuição normal padrão
z_critical_95 = stats.norm.ppf(1 - alpha_95 / 2)
z_critical_80 = stats.norm.ppf(1 - alpha_80 / 2)

# Calcular intervalo de confiança
forecast_mean = future_forecast_arima
forecast_std = np.std(train)

lower_bound_95 = forecast_mean - z_critical_95 * forecast_std
upper_bound_95 = forecast_mean + z_critical_95 * forecast_std

lower_bound_80 = forecast_mean - z_critical_80 * forecast_std
upper_bound_80 = forecast_mean + z_critical_80 * forecast_std

future_forecast_arima

# Plotar previsão vs real com ambos intervalos de confiança
plt.figure(figsize=(12, 6))

X.plot(marker='', color='red', legend=True, label='Real')
future_forecast_arima.plot(marker='', color='blue', legend=True, label='Previsto')

# Intervalo de 95%
plt.fill_between(future_forecast_arima.index,
                 lower_bound_95, upper_bound_95,
                 color='gray', alpha=0.3, label='IC 95%')

# Intervalo de 80%
plt.fill_between(future_forecast_arima.index,
                 lower_bound_80, upper_bound_80,
                 color='blue', alpha=0.3, label='IC 80%')

plt.title('Predição Taxa Média YLL - Utilizando Modelo ARIMA')
plt.ylabel('Taxa Média YLL')
plt.xlabel('Quadrimestre')
plt.legend()
# plt.grid(True)
plt.show()

# Calcular o Erro Absoluto Médio (MAE)
mae = mean_absolute_error(test,future_forecast_arima)
print(f'MAE: {mae}')

# Calcular o Erro Quadrático Médio (MSE)
mse = mean_squared_error(test,future_forecast_arima)
print(f'MSE: {mse}')
```

```
# Calcular a Raiz do Erro Quadrático Médio (RMSE)
rmse = np.sqrt(mse)
print(f'RMSE: {rmse}')

# Calcular o Erro Percentual Absoluto Médio
mape = mean_absolute_percentage_error(test,future_forecast_arima)
print(f'MAPE: {mape}')

# Calcular o erro Theil's U2
def theil_u2(actual, predicted):
    numerator = np.sum((actual - predicted) ** 2)
    denominator = np.sum((actual - np.roll(actual, 1)) ** 2) + np.sum((predicted -
np.roll(predicted, 1)) ** 2)
    return np.sqrt(numerator / denominator)

# Calcular o erro Theil's U2
TU = theil_u2(test, future_forecast_arima)
print(f'TU: {TU}')

# Teste de Durbin-Watson
model_fit = arima_model.fit(train)
dw = durbin_watson(model_fit.resid())
print(f'Durbin-Watson: {dw}')

### SARIMA

plot_acf(X)
plt.show()

plot_pacf(X, method='ywm')
plt.show()

acorr_ljungbox(X, lags=[9])

sarima_model = auto_arima(
    X,
    start_p=1,
    start_q=1,
    max_p=6,
    max_q=6,
    m=3,
    start_P=0,
    seasonal=True,
    d=1,
    D=1,
    trace=True,
    error_action='ignore',
    suppress_warnings=True,
    stepwise=True
    )
```

```
print(sarima_model.aic())

train = X.loc[:'2016-12-31']
test = X.loc['2017-01-01':]

sarima_model.fit(train)

sarima_future_forecast = sarima_model.predict(n_periods=9)

sarima_future_forecast.index

sarima_future_forecast

sarima_future_forecast.plot(marker='', color='blue', legend=True, label='Previsto')
test.plot(marker='', color='red', label='Real', legend=True)
plt.show()

sarima_future_forecast.plot(marker='', color='blue', legend=True, label='Previsto')
X.plot(marker='', color='red', label='Real', legend=True)
plt.show()

# Parâmetros para calcular os intervalos de confiança

alpha_95 = 0.05 # Nível de significância para intervalo de confiança de 95%
alpha_80 = 0.2 # Nível de significância para intervalo de confiança de 80%

# Valor crítico para distribuição normal padrão
z_critical_95 = stats.norm.ppf(1 - alpha_95 / 2)
z_critical_80 = stats.norm.ppf(1 - alpha_80 / 2)

# Calcular intervalo de confiança
forecast_mean = sarima_future_forecast
forecast_std = np.std(train)

lower_bound_95 = forecast_mean - z_critical_95 * forecast_std
upper_bound_95 = forecast_mean + z_critical_95 * forecast_std

lower_bound_80 = forecast_mean - z_critical_80 * forecast_std
upper_bound_80 = forecast_mean + z_critical_80 * forecast_std

sarima_future_forecast.index

# Plotar previsão vs real com ambos intervalos de confiança
plt.figure(figsize=(12, 6))

X.plot(marker='', color='red', legend=True, label='Real')
sarima_future_forecast.plot(marker='', color='blue', legend=True, label='Previsto')

# Intervalo de 95%
```

```
plt.fill_between(sarima_future_forecast.index,
                 lower_bound_95, upper_bound_95,
                 color='gray', alpha=0.3, label='IC 95%')

# Intervalo de 80%
plt.fill_between(sarima_future_forecast.index,
                 lower_bound_80, upper_bound_80,
                 color='blue', alpha=0.3, label='IC 80%')

plt.title('Predição Taxa Média YLL - Utilizando Modelo SARIMA')
plt.ylabel('Taxa Média YLL')
plt.xlabel('Quadrimestre')
plt.legend()
plt.show()

# Calcular o Erro Absoluto Médio (MAE)
mae = mean_absolute_error(test,sarima_future_forecast)
print(f'MAE: {mae}')

# Calcular o Erro Quadrático Médio (MSE)
mse = mean_squared_error(test,sarima_future_forecast)
print(f'MSE: {mse}')

# Calcular a Raiz do Erro Quadrático Médio (RMSE)
rmse = np.sqrt(mse)
print(f'RMSE: {rmse}')

# Calcular o Erro Percentual Absoluto Médio
mape = mean_absolute_percentage_error(test,sarima_future_forecast)
print(f'MAPE: {mape}')

# Calcular o erro Theil's U2
def theil_u2(actual, predicted):
    numerator = np.sum((actual - predicted) ** 2)
    denominator = np.sum((actual - np.roll(actual, 1)) ** 2) + np.sum((predicted -
np.roll(predicted, 1)) ** 2)
    return np.sqrt(numerator / denominator)

# Calcular o erro Theil's U2
TU = theil_u2(test, sarima_future_forecast)
print(f'TU: {TU}')

# Teste de Durbin-Watson
model_fit = sarima_model.fit(train)
dw = durbin_watson(model_fit.resid())
print(f'Durbin-Watson: {dw}')
```

xgboost_model.py

```
from google.cloud import bigquery
from google.oauth2 import service_account
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
from statsmodels.stats.stattools import durbin_watson
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
import scipy.stats as stats

# Realizar conexão com o GCP
credentials = service_account.Credentials.from_service_account_file("keys/ml-na-saude-
ed1fc3c1a83e.json")
client = bigquery.Client(credentials=credentials, project=credentials.project_id)

# Query para consulta dos dados agrupados por taxa média por quadrimestre
consulta_sql = """
select
   tx.quadrimestre,
   avg(tx.taxa_yll) as taxa_media_yll
from (
   with yll_quadrimestral as (
      select
         case
            when extract(month from y.dt_obito) between 1 and 4 then date(extract(year from
y.dt_obito), 4, 30)
            when extract(month from y.dt_obito) between 5 and 8 then date(extract(year from
y.dt_obito), 8, 31)
            when extract(month from y.dt_obito) between 9 and 12 then date(extract(year
from y.dt_obito), 12, 31)
            end as quadrimestre,
         m.nm_municipio,
         p.populacao,
         sum(y.yll) as soma_yll
      from `ml-na-saude.yll_por_obito.yll` y
      join `ml-na-saude.yll_por_obito.populacao` p on y.cd_mun_res = p.cd_municipio and
y.ano_obito = p.ano
      join `ml-na-saude.yll_por_obito.municipio` m on p.cd_municipio = m.cd_municipio
      where p.porte = 'Médio Porte'
      group by 1,2,3
   )
   select
      quadrimestre,
      nm_municipio,
      soma_yll,
      populacao,
```

```
        soma_yll / populacao * 1000 as taxa_yll
    from yll_quadrimestral
    group by 1,2,3,4
) tx
group by 1
order by 1
"""

# Ignorar avisos e gerar dataframe
warnings.simplefilter("ignore")
df = client.query(consulta_sql).to_dataframe()
# Copiar dataframe para manipular dados
time_series = df.copy()
# Transformando o quadrimestre em data
time_series['quadrimestre'] = pd.to_datetime(time_series['quadrimestre'])
# Filtrar dados até final de 2019
time_series = time_series[time_series['quadrimestre'] <= '2019-12-31']

# Separando as variáveis dependentes e independentes
X = time_series.drop(columns=['quadrimestre', 'taxa_media_yll'])
y = time_series['taxa_media_yll']

# Dividindo em treino e teste (usando os últimos 3 anos como teste)
train = time_series.loc[:'2016-12-31']
test = time_series.loc['2017-01-01':]

# Feature Engineering: Sazonalidade com funções trigonométricas para quadrimestres
train['quadrimestre_sin'] = np.sin(2 * np.pi * train['quadrimestre'].dt.month / 12)
train['quadrimestre_cos'] = np.cos(2 * np.pi * train['quadrimestre'].dt.month / 12)
test['quadrimestre_sin'] = np.sin(2 * np.pi * test['quadrimestre'].dt.month / 12)
test['quadrimestre_cos'] = np.cos(2 * np.pi * test['quadrimestre'].dt.month / 12)

# XGBoost com GridSearchCV para ajuste de parâmetros
xgb_model = xgb.XGBRegressor()

# Parâmetros a serem ajustados
param_grid = {
    'n_estimators': [500, 1000, 1500],
    'max_depth': [3, 4, 5],
    'learning_rate': [0.01, 0.05, 0.1],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'subsample': [0.6, 0.8, 1.0],
    'min_child_weight': [1, 2, 3]
}

# Aplicando GridSearchCV
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=3,
verbose=1, n_jobs=-1)
grid_search.fit(train[['quadrimestre_sin', 'quadrimestre_cos']], train['taxa_media_yll'])
```

```
# Melhores parâmetros encontrados
best_params = grid_search.best_params_
print(f"Melhores parâmetros: {best_params}")

# Treinando o modelo com os melhores parâmetros
xgb_best = xgb.XGBRegressor(**best_params)
xgb_best.fit(train[['quadrimestre_sin', 'quadrimestre_cos']], train['taxa_media_yll'])

# Fazendo previsões
predictions = xgb_best.predict(test[['quadrimestre_sin', 'quadrimestre_cos']])

# Plotando os resultados
plt.figure(figsize=(12, 6))

plt.plot(time_series['quadrimestre'], time_series['taxa_media_yll'], color='red', label='Real')
plt.plot(test['quadrimestre'], predictions, color='blue', label='Previsto')

# Intervalo de 95%
plt.fill_between(test['quadrimestre'],
                 predictions - 1.96*np.std(predictions),
                 predictions + 1.96*np.std(predictions),
                 color='gray', alpha=0.3, label='IC 95%')

# Intervalo de 80%
plt.fill_between(test['quadrimestre'],
                 predictions - 1.28*np.std(predictions),
                 predictions + 1.28*np.std(predictions),
                 color='blue', alpha=0.3, label='IC 80%')

plt.title('Predição Taxa Média YLL - Utilizando Modelo XGBoost')
plt.ylabel('Taxa Média YLL')
plt.xlabel('Quadrimestre')
plt.legend()
plt.show()

# Calcular o Erro Absoluto Médio (MAE)
mae = mean_absolute_error(test['taxa_media_yll'], predictions)
print(f'MAE: {mae}')

# Calcular o Erro Quadrático Médio (MSE)
mse = mean_squared_error(test['taxa_media_yll'], predictions)
print(f'MSE: {mse}')

# Calcular a Raiz do Erro Quadrático Médio (RMSE)
rmse = np.sqrt(mse)
print(f'RMSE: {rmse}')

# Calcular o Erro Percentual Absoluto Médio
mape = mean_absolute_percentage_error(test['taxa_media_yll'], predictions)
print(f'MAPE: {mape}')
```

```
# Calcular o erro Theil's U2
def theil_u2(actual, predicted):
    numerator = np.sum((actual - predicted) ** 2)
    denominator = np.sum((actual - np.roll(actual, 1)) ** 2) + np.sum((predicted -
np.roll(predicted, 1)) ** 2)
    return np.sqrt(numerator / denominator)

tu = theil_u2(test['taxa_media_yll'], predictions)
print(f'TU: {tu}')

# Teste de Durbin-Watson
residuals = test['taxa_media_yll'] - predictions
dw = durbin_watson(residuals)
print(f'Durbin-Watson: {dw}')

# Analisando resíduos
residuals = test['taxa_media_yll'] - predictions
plt.figure(figsize=(10,6))
plt.plot(test['quadrimestre'], residuals, 'o-')
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Análise dos Resíduos')
plt.xlabel('Quadrimestre')
plt.ylabel('Resíduo')
plt.legend()
plt.show()

# Criar figura com subplots para análise completa dos resíduos
fig = plt.figure(figsize=(15, 10))

# 1. Gráfico de Resíduos vs Tempo
plt.subplot(221)
plt.plot(test['quadrimestre'], residuals, 'o-')
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Resíduos ao Longo do Tempo')
plt.ylabel('Resíduo')
plt.xlabel('Data')
plt.grid(True)

# 2. Histograma dos Resíduos
plt.subplot(222)
plt.hist(residuals, bins=10, edgecolor='black')
plt.title('Distribuição dos Resíduos')
plt.xlabel('Resíduo')
plt.ylabel('Frequência')

# 3. Q-Q Plot para verificar normalidade
plt.subplot(223)
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot dos Resíduos')
```

```
# 4. Resíduos vs Valores Previstos
plt.subplot(224)
plt.scatter(predictions, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Resíduos vs Valores Previstos')
plt.xlabel('Valores Previstos')
plt.ylabel('Resíduos')
plt.grid(True)

plt.tight_layout()
plt.show()

# Testes estatísticos
print("\nTestes Estatísticos dos Resíduos:")

# Teste de normalidade
_, p_value_norm = stats.normaltest(residuals)
print(f"Teste de Normalidade (D'Agostino K²):")
print(f"p-valor: {p_value_norm:.4f}")
print(f"Resíduos são normais? {'Sim' if p_value_norm > 0.05 else 'Não'} (α=0.05)")

# Média dos resíduos
mean_residuals = np.mean(residuals)
print(f"\nMédia dos Resíduos: {mean_residuals:.4f}")

# Desvio padrão dos resíduos
std_residuals = np.std(residuals)
print(f"Desvio Padrão dos Resíduos: {std_residuals:.4f}")

# Teste de autocorrelação (Durbin-Watson já calculado anteriormente)
print(f"\nEstatística Durbin-Watson: {dw:.4f}")
print("Interpretação do Durbin-Watson:")
if dw < 1.5:
    print("Possível autocorrelação positiva")
elif dw > 2.5:
    print("Possível autocorrelação negativa")
else:
    print("Não há evidências fortes de autocorrelação")
```

prophet_model.py

```
from google.cloud import bigquery
from google.oauth2 import service_account
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
from statsmodels.stats.stattools import durbin_watson
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error
from prophet import Prophet
import scipy.stats as stats

# Realizar conexão com o GCP
credentials = service_account.Credentials.from_service_account_file("keys/ml-na-saude-
ed1fc3c1a83e.json")
client = bigquery.Client(credentials=credentials, project=credentials.project_id)

# Query para consulta dos dados agrupados por taxa média por quadrimestre
consulta_sql = """
select
   tx.quadrimestre,
   avg(tx.taxa_yll) as taxa_media_yll
from (
   with yll_quadrimestral as (
      select
         case
            when extract(month from y.dt_obito) between 1 and 4 then date(extract(year from
y.dt_obito), 4, 30)
            when extract(month from y.dt_obito) between 5 and 8 then date(extract(year from
y.dt_obito), 8, 31)
            when extract(month from y.dt_obito) between 9 and 12 then date(extract(year
from y.dt_obito), 12, 31)
            end as quadrimestre,
         m.nm_municipio,
         p.populacao,
         sum(y.yll) as soma_yll
      from `ml-na-saude.yll_por_obito.yll` y
      join `ml-na-saude.yll_por_obito.populacao` p on y.cd_mun_res = p.cd_municipio and
y.ano_obito = p.ano
      join `ml-na-saude.yll_por_obito.municipio` m on p.cd_municipio = m.cd_municipio
      where p.porte = 'Médio Porte'
      group by 1,2,3
   )
   select
      quadrimestre,
      nm_municipio,
      soma_yll,
      populacao,
      soma_yll / populacao * 1000 as taxa_yll
   from yll_quadrimestral
   group by 1,2,3,4
) tx
group by 1
order by 1
"""

# Ignorar avisos e gerar dataframe
```

```
warnings.simplefilter("ignore")
df = client.query(consulta_sql).to_dataframe()
# Copiar dataframe para manipular dados
time_series = df.copy()
# Transformando o quadrimestre em data
time_series['quadrimestre'] = pd.to_datetime(time_series['quadrimestre'])
# Filtrar dados até final de 2019
time_series = time_series[time_series['quadrimestre'] <= '2019-12-31']

# Preparar os dados no formato do Prophet
df_prophet = time_series.rename(columns={'quadrimestre': 'ds', 'taxa_media_yll': 'y'})

# Dividir em treino e teste
train = df_prophet[df_prophet['ds'] <= '2016-12-31']
test = df_prophet[df_prophet['ds'] > '2016-12-31']

# Criar e treinar modelo para intervalo de 95%
model_95 = Prophet(
    changepoint_prior_scale=0.001,
    seasonality_prior_scale=0.1,
    yearly_seasonality=True,
    weekly_seasonality=False,
    daily_seasonality=False,
    seasonality_mode='additive',
    interval_width=0.95,
    n_changepoints=5
)
model_95.fit(train)

# Criar e treinar modelo para intervalo de 80%
model_80 = Prophet(
    changepoint_prior_scale=0.001,
    seasonality_prior_scale=0.1,
    yearly_seasonality=True,
    weekly_seasonality=False,
    daily_seasonality=False,
    seasonality_mode='additive',
    interval_width=0.80,
    n_changepoints=5
)
model_80.fit(train)

# Criar dataframe para previsão
future = model_95.make_future_dataframe(periods=9, freq='4M')
forecast_95 = model_95.predict(future)
forecast_80 = model_80.predict(future)

# Plotar previsão vs real com ambos intervalos de confiança
test_forecast_95 = forecast_95[forecast_95['ds'] > '2016-12-31']
test_forecast_80 = forecast_80[forecast_80['ds'] > '2016-12-31']
```

```
plt.figure(figsize=(12, 6))

plt.plot(df_prophet['ds'], df_prophet['y'], color='red', label='Real')
plt.plot(test_forecast_95['ds'], test_forecast_95['yhat'], color='blue', label='Previsto')

# Intervalo de 95%
plt.fill_between(test_forecast_95['ds'],
                 test_forecast_95['yhat_lower'],
                 test_forecast_95['yhat_upper'],
                 color='gray', alpha=0.3, label='IC 95%')

# Intervalo de 80%
plt.fill_between(test_forecast_80['ds'],
                test_forecast_80['yhat_lower'],
                test_forecast_80['yhat_upper'],
                color='blue', alpha=0.3, label='IC 80%')

plt.title('Predição Taxa Média YLL - Utilizando Modelo Prophet')
plt.ylabel('Taxa Média YLL')
plt.xlabel('Quadrimestre')
plt.legend()
plt.show()

# Calcular métricas de erro usando previsões do modelo de 95%
y_true = test['y'].values
y_pred = test_forecast_95['yhat'].values

# Calcular o Erro Absoluto Médio (MAE)
mae = mean_absolute_error(y_true, y_pred)
print(f'MAE: {mae}')

# Calcular o Erro Quadrático Médio (MSE)
mse = mean_squared_error(y_true, y_pred)
print(f'MSE: {mse}')

# Calcular a Raiz do Erro Quadrático Médio (RMSE)
rmse = np.sqrt(mse)
print(f'RMSE: {rmse}')

# Calcular o Erro Percentual Absoluto Médio
mape = mean_absolute_percentage_error(y_true,y_pred)
print(f'MAPE: {mape}')

# Calcular o erro Theil's U2
def theil_u2(actual, predicted):
    numerator = np.sum((actual - predicted) ** 2)
    denominator = np.sum((actual - np.roll(actual, 1)) ** 2) + np.sum((predicted -
np.roll(predicted, 1)) ** 2)
    return np.sqrt(numerator / denominator)
```

```
tu = theil_u2(y_true, y_pred)
print(f'TU: {tu}')

# Teste de Durbin-Watson
residuals = y_true - y_pred
dw = durbin_watson(residuals)
print(f'Durbin-Watson: {dw}')

# Análise dos resíduos
plt.figure(figsize=(10, 6))
plt.plot(test['ds'], residuals, 'o-')
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Análise dos Resíduos')
plt.xlabel('Quadrimestre')
plt.ylabel('Resíduo')
# plt.grid(True)
plt.show()

# Criar figura com subplots para análise completa dos resíduos
fig = plt.figure(figsize=(15, 10))

# 1. Gráfico de Resíduos vs Tempo
plt.subplot(221)
plt.plot(test['ds'], residuals, 'o-')
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Resíduos ao Longo do Tempo')
plt.ylabel('Resíduo')
plt.xlabel('Data')
plt.grid(True)

# 2. Histograma dos Resíduos
plt.subplot(222)
plt.hist(residuals, bins=10, edgecolor='black')
plt.title('Distribuição dos Resíduos')
plt.xlabel('Resíduo')
plt.ylabel('Frequência')

# 3. Q-Q Plot para verificar normalidade
plt.subplot(223)
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot dos Resíduos')

# 4. Resíduos vs Valores Previstos
plt.subplot(224)
plt.scatter(y_pred, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Resíduos vs Valores Previstos')
plt.xlabel('Valores Previstos')
plt.ylabel('Resíduos')
```

```
plt.grid(True)

plt.tight_layout()
plt.show()

# Testes estatísticos
print("\nTestes Estatísticos dos Resíduos:")

# Teste de normalidade
_, p_value_norm = stats.normaltest(residuals)
print(f"Teste de Normalidade (D'Agostino K²):")
print(f"p-valor: {p_value_norm:.4f}")
print(f"Resíduos são normais? {'Sim' if p_value_norm > 0.05 else 'Não'} (α=0.05)")

# Média dos resíduos
mean_residuals = np.mean(residuals)
print(f"\nMédia dos Resíduos: {mean_residuals:.4f}")

# Desvio padrão dos resíduos
std_residuals = np.std(residuals)
print(f"Desvio Padrão dos Resíduos: {std_residuals:.4f}")

# Teste de autocorrelação (Durbin-Watson já calculado anteriormente)
print(f"\nEstatística Durbin-Watson: {dw:.4f}")
print("Interpretação do Durbin-Watson:")
if dw < 1.5:
    print("Possível autocorrelação positiva")
elif dw > 2.5:
    print("Possível autocorrelação negativa")
else:
    print("Não há evidências fortes de autocorrelação")
```

lstm_model.py

```
from google.cloud import bigquery
from google.oauth2 import service_account
import pandas as pd
import numpy as np
import warnings
import keras_tuner as kt
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
mean_absolute_error
from statsmodels.stats.stattools import durbin_watson
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import MinMaxScaler
```

```
import shutil

# Realizar conexão com o GCP
credentials = service_account.Credentials.from_service_account_file("keys/ml-na-saude-ed1fc3c1a83e.json")
client = bigquery.Client(credentials=credentials, project=credentials.project_id)

# Query para consulta dos dados agrupados por taxa média por quadrimestre
consulta_sql = """
select
   tx.quadrimestre,
   avg(tx.taxa_yll) as taxa_media_yll
from (
   with yll_quadrimestral as (
      select
         case
            when extract(month from y.dt_obito) between 1 and 4 then date(extract(year from y.dt_obito), 4, 30)
            when extract(month from y.dt_obito) between 5 and 8 then date(extract(year from y.dt_obito), 8, 31)
            when extract(month from y.dt_obito) between 9 and 12 then date(extract(year from y.dt_obito), 12, 31)
            end as quadrimestre,
         m.nm_municipio,
         p.populacao,
         sum(y.yll) as soma_yll
      from `ml-na-saude.yll_por_obito.yll` y
      join `ml-na-saude.yll_por_obito.populacao` p on y.cd_mun_res = p.cd_municipio and y.ano_obito = p.ano
      join `ml-na-saude.yll_por_obito.municipio` m on p.cd_municipio = m.cd_municipio
      where p.porte = 'Médio Porte'
      group by 1,2,3
   )
   select
      quadrimestre,
      nm_municipio,
      soma_yll,
      populacao,
      soma_yll / populacao * 1000 as taxa_yll
   from yll_quadrimestral
   group by 1,2,3,4
) tx
group by 1
order by 1
"""

# Ignorar avisos e gerar dataframe
warnings.simplefilter("ignore")
df = client.query(consulta_sql).to_dataframe()
# Copiar dataframe para manipular dados
```

```
time_series = df.copy()
# Transformando o quadrimestre em data
time_series['quadrimestre'] = pd.to_datetime(time_series['quadrimestre'])
# Filtrar dados até final de 2019
time_series = time_series[time_series['quadrimestre'] <= '2019-12-31']
# Setando o quadrimestre como índice da tabela
time_series = time_series.set_index('quadrimestre')

train_data= time_series.values

# Normaliza os Dados
normalizer = MinMaxScaler(feature_range=(0,1))
train_data = normalizer.fit_transform(train_data)

# Cria os Arrays No Formato Certo
window_size = 3
x = []
y = []
for i in range(window_size, len(train_data)):
    x.append(train_data[i - window_size:i, 0])
    y.append(train_data[i, 0])
x, y = np.array(x), np.array(y)
x = np.reshape(x, (x.shape[0], x.shape[1], 1))

# Converte os Dados Para Float
x = np.asarray(x).astype('float32')
y = np.asarray(y).astype('float32')

# Divide os dados em treinamento e teste (80% treino, 10% teste, 10% validação)
split_index_1 = int(len(x) * 0.8)
split_index_2 = int(len(x) * 0.9)
x_train, y_train = x[:split_index_1], y[:split_index_1]
x_test, y_test = x[split_index_1:split_index_2], y[split_index_1:split_index_2]
x_val, y_val = x[split_index_2:], y[split_index_2:]

dropout_val = 0.2

def build_model(hp):
    neurons_first_layer = hp.Choice('neurons_first_layer', [800])
    neurons_second_layer = hp.Choice('neurons_second_layer', [300])
    neurons_third_layer = hp.Choice('neurons_third_layer', [100])
    neurons_fourth_layer = hp.Choice('neurons_fourth_layer', [200])
    dropout_val = hp.Choice('dropout_val', [0.2])

    regressor = Sequential()
    regressor.add(LSTM(units=neurons_first_layer, return_sequences=True,
input_shape=(x_train.shape[1], 1)))
    regressor.add(Dropout(dropout_val))

    if neurons_second_layer:
```

```
        regressor.add(LSTM(units=neurons_second_layer, return_sequences=True))
        regressor.add(Dropout(dropout_val))

    if neurons_third_layer:
        regressor.add(LSTM(units=neurons_third_layer, return_sequences=True))
        regressor.add(Dropout(dropout_val))

    if neurons_fourth_layer:
        regressor.add(LSTM(units=neurons_fourth_layer))
        regressor.add(Dropout(dropout_val))

    regressor.add(Dense(units=1, activation='linear'))

    regressor.compile(optimizer='adam', loss='mean_squared_error',
metrics=['mean_squared_error'])

    return regressor

# Configurando o tuner para realizar uma busca em grade
tuner = kt.GridSearch(
    hypermodel=build_model,
    objective=kt.Objective("val_mean_squared_error", direction="min"),
    directory="my_dir",
    project_name="otimizacao_keras_tuner"
)

tuner.search(x_train, y_train, batch_size = 16, epochs=100, validation_data=(x_val, y_val))
best_model = tuner.get_best_models(num_models=1)[0]
best_model.summary()

# # Treinando o melhor modelo com o conjunto de treinamento
# history = best_model.fit(x_train, y_train, batch_size = 16, epochs=100,
validation_data=(x_val, y_val))

# Callback para Early Stopping
early_stopping = EarlyStopping(
    monitor='loss',
    patience=100,
    restore_best_weights=True
)

# Treinando o melhor modelo com o conjunto de treinamento com Early Stopping
history = best_model.fit(
    x_train, y_train,
    batch_size=16,
    epochs=1000,
    validation_data=(x_val, y_val),
    callbacks=[early_stopping]
)
```

```
y_test = y_test.reshape(-1, 1)
train_predictions = best_model.predict(x_test)
train_predictions = normalizer.inverse_transform(train_predictions)
y_test = normalizer.inverse_transform(y_test)

y_val = y_val.reshape(-1, 1)
val_predictions = best_model.predict(x_val)
val_predictions = normalizer.inverse_transform(val_predictions)
y_val = normalizer.inverse_transform(y_val)

# Remove the directory containing the previous tuning results
try:
    shutil.rmtree('my_dir')
except:
    print('Diretório não encontrado.')

train_results = pd.DataFrame(zip(train_predictions, y_test), columns = ['previsao',
'valor_real'])
train_results['previsao'] = train_results['previsao'].apply(lambda x: x[0])
train_results['valor_real'] = train_results['valor_real'].apply(lambda x: x[0])

train_results_val = pd.DataFrame(zip(val_predictions, y_val), columns = ['previsao',
'valor_real'])
train_results_val['previsao'] = train_results_val['previsao'].apply(lambda x: x[0])
train_results_val['valor_real'] = train_results_val['valor_real'].apply(lambda x: x[0])

start_date = "2018-04-30"
train_results['data'] = pd.date_range(start=start_date, periods=len(train_results), freq='4M')
train_results.set_index('data', inplace=True)

train_results

start_date = "2019-04-30"
train_results_val['data'] = pd.date_range(start=start_date, periods=len(train_results_val),
freq='4M')
train_results_val.set_index('data', inplace=True)

train_results_val

plt.plot(train_results['valor_real'], color='red', label = 'Real')
plt.plot(train_results['previsao'], color='blue', label = 'Previsto')
# Formatando o eixo x para mostrar apenas o mês
plt.xticks(ticks=train_results.index, labels=train_results.index.strftime('%b'))

plt.title('Teste: comparação entre valores reais e previstos (2018)')
plt.xlabel('Mês')
plt.legend()
plt.show()

plt.plot(train_results_val['valor_real'], color='red', label = 'Real')
```

```
plt.plot(train_results_val['previsao'], color='blue', label = 'Previsto')
# Formatando o eixo x para mostrar apenas o mês
plt.xticks(ticks=train_results_val.index, labels=train_results_val.index.strftime('%b'))
# Adicionando um título ao gráfico
plt.title('Validação: comparação entre valores reais e previstos (2019)')
plt.xlabel('Mês')
plt.legend()
plt.show()

plt.plot(train_results_val['previsao'], color='blue', label = 'Previsto - Validação')
plt.plot(train_results['previsao'], color='purple', label = 'Previsto - Teste')
plt.plot(time_series, color='red', label = 'Real')

plt.title('Comparação entre valores reais e previstos: validação e teste')
plt.ylabel('Taxa Internações')
plt.xlabel('Ano')
plt.legend()
plt.show()

# Calcular os resíduos (erro entre os valores reais e previstos)
residuals_t = train_results['valor_real'] - train_results['previsao']
residuals_v = train_results_val['valor_real'] - train_results_val['previsao']

# Desvio padrão dos resíduos
std_error_t = np.std(residuals_t)
std_error_v = np.std(residuals_v)

# Definir os intervalos de confiança (95% e 80%)
confidence_interval_95_t = 1.96 * std_error_t  # 95% de confiança
confidence_interval_80_t = 1.28 * std_error_t  # 80% de confiança
confidence_interval_95_v = 1.96 * std_error_v  # 95% de confiança
confidence_interval_80_v = 1.28 * std_error_v  # 80% de confiança

# Calcular os limites dos intervalos
upper_bound_95_t = train_results['previsao'] + confidence_interval_95_t
lower_bound_95_t = train_results['previsao'] - confidence_interval_95_t
upper_bound_80_t = train_results['previsao'] + confidence_interval_80_t
lower_bound_80_t = train_results['previsao'] - confidence_interval_80_t
upper_bound_95_v = train_results_val['previsao'] + confidence_interval_95_v
lower_bound_95_v = train_results_val['previsao'] - confidence_interval_95_v
upper_bound_80_v = train_results_val['previsao'] + confidence_interval_80_v
lower_bound_80_v = train_results_val['previsao'] - confidence_interval_80_v

# Plotar os valores reais e previstos
plt.figure(figsize=(12, 6))
plt.plot(time_series, color='red', label = 'Real')
plt.plot(train_results.index, train_results['previsao'], color='purple', label = 'Previsto -
Teste')
plt.plot(train_results_val.index, train_results_val['previsao'], color='blue', label = 'Previsto -
Validação')
```

```
# Plotar os intervalos de confiança
plt.fill_between(train_results.index, lower_bound_95_t, upper_bound_95_t, color='gray',
alpha=0.3, label='IC 95%')
plt.fill_between(train_results.index, lower_bound_80_t, upper_bound_80_t, color='blue',
alpha=0.3, label='IC 80%')
plt.fill_between(train_results_val.index, lower_bound_95_v, upper_bound_95_v,
color='gray', alpha=0.3, label=None)
plt.fill_between(train_results_val.index, lower_bound_80_v, upper_bound_80_v,
color='blue', alpha=0.3, label=None)

# Adicionar título e legendas
plt.title('Predição da Taxa Média YLL com Intervalos de Confiança (Modelo 2)')
plt.ylabel('Taxa Média YLL')
plt.xlabel('Ano')
plt.legend()
plt.show()

# MÉTRICAS DE ERROS

# Calcular o Erro Absoluto Médio (MAE)
mae = mean_absolute_error(train_results['valor_real'], train_results['previsao'])
print(f'MAE: {mae}')

# Calcular o Erro Quadrático Médio (MSE)
mse = mean_squared_error(train_results['valor_real'], train_results['previsao'])
print(f'MSE: {mse}')

# Calcular a Raiz do Erro Quadrático Médio (RMSE)
rmse = np.sqrt(mse)
print(f'RMSE: {rmse}')

# Calcular o Erro Percentual Absoluto Médio
mape = mean_absolute_percentage_error(train_results['valor_real'],
train_results['previsao'])
print(f'MAPE: {mape}')

# Calcular o erro Theil's U2
def theil_u2(actual, predicted):
    numerator = np.sum((actual - predicted) ** 2)
    denominator = np.sum((actual - np.roll(actual, 1)) ** 2) + np.sum((predicted -
np.roll(predicted, 1)) ** 2)
    return np.sqrt(numerator / denominator)

tu = theil_u2(train_results['valor_real'], train_results['previsao'])
print(f'TU: {tu}')

# Teste de Durbin-Watson
residuals = train_results['valor_real'] - train_results['previsao']
dw = durbin_watson(residuals)
```

```
print(f'Durbin-Watson: {dw}')
```

predicao_yll.py

```
from google.cloud import bigquery
from google.oauth2 import service_account
import pandas as pd
import numpy as np
import warnings
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

# Realizar conexão com o GCP
credentials = service_account.Credentials.from_service_account_file("keys/ml-na-saude-
ed1fc3c1a83e.json")
client = bigquery.Client(credentials=credentials, project=credentials.project_id)

# Query para consulta dos dados agrupados por taxa média por quadrimestre
consulta_sql = """
select
   tx.quadrimestre,
   avg(tx.taxa_yll) as taxa_media_yll
from (
   with yll_quadrimestral as (
      select
         case
            when extract(month from y.dt_obito) between 1 and 4 then date(extract(year from
y.dt_obito), 4, 30)
            when extract(month from y.dt_obito) between 5 and 8 then date(extract(year from
y.dt_obito), 8, 31)
            when extract(month from y.dt_obito) between 9 and 12 then date(extract(year
from y.dt_obito), 12, 31)
            end as quadrimestre,
         m.nm_municipio,
         p.populacao,
         sum(y.yll) as soma_yll
      from `ml-na-saude.yll_por_obito.yll` y
      join `ml-na-saude.yll_por_obito.populacao` p on y.cd_mun_res = p.cd_municipio and
y.ano_obito = p.ano
      join `ml-na-saude.yll_por_obito.municipio` m on p.cd_municipio = m.cd_municipio
      where p.porte = 'Médio Porte'
      group by 1,2,3
   )
   select
      quadrimestre,
      nm_municipio,
```

```
        soma_yll,
        populacao,
        soma_yll / populacao * 1000 as taxa_yll
    from yll_quadrimestral
    group by 1,2,3,4
) tx
group by 1
order by 1
"""

# Ignorar avisos e gerar dataframe
warnings.simplefilter("ignore")
df = client.query(consulta_sql).to_dataframe()
# Copiar dataframe para manipular dados
time_series = df.copy()
# Transformando o quadrimestre em data
time_series['quadrimestre'] = pd.to_datetime(time_series['quadrimestre'])
# # Filtrar dados até final de 2019
time_series = time_series[time_series['quadrimestre'] <= '2019-12-31']
# Setando o quadrimestre como índice da tabela
time_series = time_series.set_index('quadrimestre')

# Normaliza os Dados
normalizer = MinMaxScaler(feature_range=(0, 1))
train_data = normalizer.fit_transform(time_series.values)

# Cria os Arrays No Formato Certo
window_size = 3
x = []
y = []
for i in range(window_size, len(train_data)):
    x.append(train_data[i - window_size:i, 0])
    y.append(train_data[i, 0])
x, y = np.array(x), np.array(y)
x = np.reshape(x, (x.shape[0], x.shape[1], 1))

# Treina com todos os dados
def build_model_for_future(hp=None):
    regressor = Sequential()
    regressor.add(LSTM(units=800, return_sequences=True, input_shape=(x.shape[1], 1)))
    regressor.add(Dropout(0.2))
    regressor.add(LSTM(units=300, return_sequences=True))
    regressor.add(Dropout(0.2))
    regressor.add(LSTM(units=100, return_sequences=True))
    regressor.add(Dropout(0.2))
    regressor.add(LSTM(units=200))
    regressor.add(Dropout(0.2))
    regressor.add(Dense(units=1, activation='linear'))
    regressor.compile(optimizer='adam', loss='mean_squared_error',
metrics=['mean_squared_error'])
```

```
    return regressor

# Inicializa o modelo
model = build_model_for_future()

# Callback para Early Stopping
early_stopping = EarlyStopping(
    monitor='loss',
    patience=100,
    restore_best_weights=True
)

# Treina o modelo com todos os dados disponíveis
history = model.fit(
    x, y,
    batch_size=16,
    epochs=1000,
    callbacks=[early_stopping],
    validation_split=0.1
)

# Previsão Futura
future_steps = 9
last_window = train_data[-window_size:]
predictions = []

for _ in range(future_steps):
    last_window_reshaped = np.reshape(last_window, (1, last_window.shape[0], 1))
    prediction = model.predict(last_window_reshaped)
    predictions.append(prediction[0, 0])
    last_window = np.append(last_window[1:], prediction, axis=0)

# Revertendo a Normalização
future_predictions = normalizer.inverse_transform(np.array(predictions).reshape(-1, 1))

# Gerando Datas para os Próximos 9 Quadrimestres
last_date = time_series.index[-1]
future_dates = [last_date + pd.DateOffset(months=4 * i) for i in range(1, future_steps + 1)]

# Criação do DataFrame com as Previsões Futuras
future_results = pd.DataFrame({
    'Data': future_dates,
    'Previsao YLL': future_predictions.flatten()
})
future_results.set_index('Data', inplace=True)

# Exibindo as Previsões Futuras
print(future_results)

# Plotando as Previsões Futuras
```

```
plt.figure(figsize=(12, 6))
# plt.plot(time_series.index, time_series.values, color='red', label='Real')
plt.plot(time_series.index, time_series.values, color='red', label='Real')
plt.plot(future_results.index, future_results['Previsao YLL'], color='blue', label='Previsão
Futura', linestyle='dashed')
plt.title('Previsão Futura da Taxa Média de YLL')
plt.ylabel('Taxa Média YLL')
plt.xlabel('Ano')
plt.legend()
plt.show()
```

APÊNDICE C - ARTIGO

Modelo preditivo dos anos de vida perdidos por morte prematura para os municípios brasileiros de médio porte utilizando aprendizagem de máquina

Marco Cesar da Silva[1]

[1] Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

marco.cesar@grad.ufsc.br

Abstract. *The Years of Life Lost (YLL) indicator measures the impact of premature mortality and supports the formulation of public health policies. This study evaluates predictive models to estimate YLL in medium-sized Brazilian municipalities, using data from DATASUS and machine learning techniques. Applying the CRISP-DM framework, the research developed an automated pipeline and compared the ARIMA, SARIMA, XGBoost, Prophet, and LSTM models. The results showed that the LSTM model achieved the highest accuracy in predicting YLL, both for the present and the future. This approach provides support for strategic public health decisions aimed at reducing premature deaths.*

Resumo. *O indicador de Anos de Vida Perdidos (YLL) mede o impacto da mortalidade precoce e auxilia na formulação de políticas públicas de saúde. Este estudo avalia modelos preditivos para estimar YLL em municípios brasileiros de médio porte, utilizando dados do DATASUS e técnicas de aprendizagem de máquina. Aplicando o framework CRISP-DM, o trabalho desenvolveu um pipeline automatizado e comparou os modelos ARIMA, SARIMA, XGBoost, Prophet e LSTM. Os resultados mostraram o modelo LSTM apresentou maior acurácia nas predições de YLL, tanto para o presente quanto para o futuro. Essa abordagem oferece suporte a decisões estratégicas de saúde pública, visando reduzir mortes prematuras.*

1. Introdução

Os Anos de Vida Perdidos por Morte Prematura (YLL, na sigla em inglês) são uma métrica essencial na saúde pública, amplamente utilizada para avaliar a carga de doenças e o impacto da mortalidade precoce na sociedade. Este indicador foi introduzido no *Global Burden of Disease Study* e rapidamente consolidou-se como uma ferramenta poderosa para mensurar o tempo de vida saudável perdido devido a óbitos prematuros (MURRAY et al., 2021). Ao considerar não apenas o número de mortes, mas também a idade em que ocorrem, o YLL oferece uma visão mais aprofundada sobre as desigualdades e vulnerabilidades em saúde, possibilitando intervenções mais direcionadas e eficazes.

No contexto brasileiro, compreender e prever o YLL é particularmente relevante devido às significativas desigualdades regionais e socioeconômicas. Estudos como o de França et al. (2017) destacam que, enquanto as taxas de YLL diminuíram em nível nacional entre 1990 e 2015, regiões como o Norte e o Nordeste ainda apresentam índices elevados, atribuídos a doenças infecciosas e mortalidade materno-infantil. Essa disparidade reforça a necessidade de ferramentas analíticas avançadas para apoiar políticas públicas e reduzir essas desigualdades.

Apesar da importância do YLL, o acesso a dados atualizados para seu cálculo enfrenta desafios significativos. No Brasil, as informações de mortalidade disponibilizadas pelo Sistema de Informações sobre Mortalidade (SIM) do DATASUS possuem atrasos consideráveis, podendo levar até dois anos para serem validadas e publicadas, conforme estabelece a Portaria SVS nº 116 de 2009 (Ministério da Saúde, 2009). Além disso, eventos como a pandemia de COVID-19 exacerbaram distorções nesses dados, dificultando ainda mais sua utilização para ações imediatas e estratégicas.

Para enfrentar esses desafios, este trabalho tem como objetivo principal avaliar modelos preditivos para estimar os YLL em municípios brasileiros de médio porte, utilizando aprendizado de máquina. A pesquisa foi estruturada com base no framework CRISP-DM (*Cross Industry Standard Process for Data Mining*), permitindo uma abordagem sistemática desde a compreensão do problema de negócio até a aplicação prática dos modelos. Técnicas como ARIMA, SARIMA, XGBoost, Prophet e LSTM foram aplicadas para comparar diferentes abordagens preditivas e determinar a mais eficaz para *nowcasting* (previsão do presente) e *forecasting* (previsão do futuro).

A escolha do aprendizado de máquina como abordagem central justifica-se por sua capacidade de lidar com grandes volumes de dados, identificar padrões complexos e superar limitações de métodos estatísticos tradicionais (RAJKOMAR; DEAN; KOHANE, 2018). O desenvolvimento de um pipeline automatizado foi uma etapa crítica para garantir eficiência no processamento e qualidade das análises, alinhando-se ao objetivo de oferecer uma solução prática e replicável para apoiar a tomada de decisões em saúde pública.

Além de sua relevância prática, este trabalho contribui academicamente ao explorar novas aplicações de aprendizado de máquina no campo da saúde pública, destacando o potencial dessas tecnologias em cenários complexos e desafiadores. A previsão de YLL com maior precisão tem o potencial de auxiliar gestores públicos na identificação de áreas vulneráveis e na formulação de políticas mais direcionadas, promovendo uma redução das desigualdades regionais e a melhoria da saúde populacional no Brasil.

2. Trabalhos Relacionados

A análise de séries temporais e a aplicação de aprendizado de máquina no contexto da saúde pública têm sido amplamente exploradas em diversos estudos. Contudo, muitas dessas abordagens apresentam limitações, como o foco em regiões específicas, o uso de métodos estatísticos tradicionais e a ausência de inovações tecnológicas para tratar dados complexos. Este trabalho propõe avanços ao integrar técnicas de aprendizado de máquina para prever os Anos de Vida Perdidos por Morte Prematura (YLL) em municípios brasileiros de médio porte, trazendo uma perspectiva inovadora em relação aos estudos existentes. A seguir, são apresentados quatro trabalhos relacionados que serviram como base de comparação e inspiração para o desenvolvimento deste estudo.

1. Hu et al. (2021): Este estudo analisou a relação entre temperatura ambiente e perda de anos de vida devido a doenças cardiovasculares na China. Utilizando o modelo DLNM (*Distributed Lag Non-linear Model*), os autores exploraram dados históricos para avaliar padrões de exposição e resposta. Embora o estudo tenha demonstrado associações importantes, ele se restringiu ao uso de modelos estatísticos tradicionais e não explorou aprendizado de máquina, limitando seu alcance preditivo e sua aplicabilidade a outros contextos.

2. Li et al. (2021): Este trabalho examinou os impactos do dióxido de nitrogênio (NO_2) na perda de anos de vida em 48 cidades chinesas, empregando modelos aditivos generalizados para avaliar as associações. A análise revelou correlações significativas, mas, assim como no estudo de Hu et al., não incorporou técnicas de aprendizado de máquina, restringindo sua capacidade de capturar padrões mais complexos nos dados.

3. Cheng et al. (2021): Focado na relação entre partículas atmosféricas (PM2.5 e PM10) e a perda de expectativa de vida em Hong Kong, este estudo utilizou modelos de defasagem distribuída para identificar padrões lineares e não lineares. Apesar de sua robustez em séries temporais, o estudo careceu de abordagem preditiva para antecipar variações futuras e não incluiu avanços tecnológicos, como redes neurais ou algoritmos baseados em árvores de decisão.

4. Gomez-Cravioto et al. (2024): Este trabalho investigou a previsão da disseminação da COVID-19 no México, utilizando tanto modelos estatísticos (ARIMA) quanto redes neurais profundas (LSTM). Embora o uso de aprendizado de máquina represente um avanço em relação aos estudos anteriores, a aplicação foi limitada a dados de pandemia, com menor generalização para outros cenários de saúde pública, como o YLL.

A partir da análise dos trabalhos citados, observa-se que, enquanto os estudos anteriores fornecem contribuições valiosas para a compreensão e modelagem de dados de saúde, a maioria se limita a métodos estatísticos tradicionais ou apresenta aplicações específicas sem abordar questões mais amplas, como a previsão de YLL em nível municipal. Em contraste, o presente trabalho traz inovações significativas ao:

- Focar em municípios brasileiros de médio porte, uma área ainda pouco explorada na literatura;
- Incorporar técnicas avançadas de aprendizado de máquina, incluindo ARIMA, SARIMA, XGBoost, Prophet e LSTM, ampliando o potencial preditivo;
- Desenvolver um pipeline automatizado para tratamento e análise de dados, otimizando a replicação e expansão da metodologia;
- Adotar a abordagem de *nowcasting* para mitigar os atrasos na disponibilização de dados, um diferencial crucial no contexto brasileiro.

Com essas contribuições, este estudo não apenas preenche lacunas existentes, mas também apresenta uma abordagem prática e inovadora para apoiar decisões estratégicas de saúde pública no Brasil, demonstrando a aplicabilidade de modelos preditivos para reduzir desigualdades e melhorar os indicadores de saúde populacional.

3. Metodologia

O desenvolvimento deste trabalho foi estruturado com base no framework CRISP-DM (*Cross Industry Standard Process for Data Mining*), que oferece um processo sistemático e iterativo para projetos de análise de dados. O objetivo principal foi avaliar modelos

preditivos para estimar os Anos de Vida Perdidos por Morte Prematura (YLL) em municípios brasileiros de médio porte, utilizando aprendizado de máquina. As etapas do trabalho foram organizadas em engenharia de dados, modelagem preditiva, validação dos modelos e implementação prática, com cada fase detalhada a seguir.

Pipeline de Dados

A construção de um pipeline de dados foi uma etapa fundamental para garantir a qualidade e eficiência no processamento das informações. Os dados de mortalidade foram extraídos do Sistema de Informações sobre Mortalidade (SIM), fornecido pelo DATASUS, enquanto dados populacionais e características dos municípios foram obtidos do Instituto Brasileiro de Geografia e Estatística (IBGE). O pipeline seguiu as etapas de extração, transformação e carga (ETL):

1) Extração: Foram coletados dados brutos de mortalidade, populações estimadas e características socioeconômicas dos municípios brasileiros. Esta etapa utilizou scripts em Python para automatizar o download e consolidação de informações de múltiplas fontes.
2) Transformação: Após a extração, os dados passaram por processos de limpeza, padronização e enriquecimento. Informações inconsistentes ou ausentes foram tratadas utilizando métodos como imputação de valores e normalização. Além disso, variáveis relevantes para o cálculo do YLL, como idade e causa de morte, foram derivadas com base nas tabelas de expectativa de vida padrão.
3) Carga: Os dados transformados foram armazenados no BigQuery do Google Cloud Platform. Bibliotecas como Pandas e google.cloud foram utilizadas para gerenciar e acessar os dados durante as etapas subsequentes do projeto.

Modelagem Preditiva

A modelagem preditiva foi realizada com o objetivo de comparar diferentes algoritmos para prever o YLL, abrangendo técnicas tradicionais e modernas de aprendizado de máquina. Para isso, os dados foram divididos em dois conjuntos: 70% para treinamento e 30% para teste, diferenciando apenas no LSTM onde os dados foram divididos em três conjuntos: 80% para treinamento, 10% para teste e 10% para validação. As abordagens modeladas incluem:

- ARIMA e SARIMA: Modelos estatísticos amplamente utilizados para séries temporais, focados em capturar tendências e sazonalidades.
- XGBoost: Um algoritmo de aprendizado baseado em árvores de decisão, conhecido por sua robustez e desempenho em cenários com dados estruturados.
- Prophet: Um modelo desenvolvido pelo Facebook, adequado para séries temporais com padrões sazonais e dados ausentes.
- LSTM: Redes neurais recorrentes projetadas para capturar dependências de longo prazo em séries temporais, utilizadas para explorar não linearidades complexas nos dados.

Os parâmetros dos modelos foram ajustados por meio de técnicas de busca em grade e validação cruzada, otimizando seu desempenho em métricas como erro médio absoluto (MAE), erro quadrático médio (MSE) e erro percentual absoluto médio (MAPE).

Validação dos Modelos

A validação dos modelos foi conduzida para garantir sua capacidade de generalização e robustez. O teste de estacionariedade (Augmented Dickey-Fuller) e a análise de resíduos foram realizados para avaliar a adequação dos modelos ARIMA e SARIMA aos dados de séries temporais.

Modelos baseados em aprendizado de máquina, como XGBoost e LSTM, foram avaliados quanto à capacidade de capturar padrões complexos e não triviais, enquanto Prophet foi utilizado para lidar com sazonalidades evidentes nos dados. Comparações entre os modelos foram feitas com base nos resultados das métricas de erro calculadas no conjunto de teste.

4. Resultados e Discussões

Os resultados obtidos neste trabalho refletem a avaliação de diferentes modelos preditivos para estimar os Anos de Vida Perdidos por Morte Prematura (YLL) em municípios brasileiros de médio porte. Os modelos avaliados incluem abordagens estatísticas tradicionais, como ARIMA e SARIMA, além de métodos baseados em aprendizado de máquina, como XGBoost, Prophet e LSTM. Os desempenhos foram comparados utilizando métricas como Erro Quadrático Médio (MSE), Erro Absoluto Médio (MAE) e Erro Percentual Absoluto Médio (MAPE), com base no conjunto de teste.

Os modelos apresentaram diferenças significativas em termos de capacidade preditiva e robustez. Na Tabela 1, são apresentados os valores das métricas calculadas para cada modelo no conjunto de teste:

Tabela 1. Comparação das métricas dos modelos

Modelo	MSE	MAE	MAPE
ARIMA	0.152	0.239	8.5%
SARIMA	0.141	0.225	7.9%
XGBoost	0.118	0.204	7.2%
Prophet	0.125	0.211	7.4%
LSTM	0.104	0.197	6.8%

Os resultados mostram que o modelo LSTM apresentou os melhores desempenhos em todas as métricas avaliadas, seguido de perto pelo XGBoost. Ambos os modelos superaram os métodos estatísticos tradicionais (ARIMA e SARIMA), indicando maior adequação para lidar com as características complexas dos dados de YLL.

- ARIMA e SARIMA: Embora sejam amplamente utilizados em séries temporais, os modelos ARIMA e SARIMA apresentaram limitações ao capturar sazonalidades e tendências não lineares nos dados. A inclusão de componentes sazonais no SARIMA resultou em uma melhora modesta em relação ao ARIMA, mas ambos os modelos tiveram dificuldades em prever variações abruptas.
- Prophet: O modelo Prophet se destacou por sua simplicidade de configuração e por lidar bem com padrões sazonais evidentes. No entanto, apresentou maior sensibilidade a outliers, o que impactou negativamente suas métricas de erro.
- XGBoost: Este modelo demonstrou excelente desempenho, sendo particularmente eficaz na captura de interações entre variáveis e padrões complexos nos dados. Sua robustez foi evidenciada pela baixa sensibilidade a outliers e pela consistência das predições.

- LSTM: O modelo baseado em redes neurais recorrentes (LSTM) foi o mais preciso, devido à sua capacidade de capturar dependências de longo prazo e não linearidades nos dados. Apesar de demandar maior poder computacional e ajuste fino de hiperparâmetros, os resultados confirmam sua superioridade em predições de YLL.

Os resultados obtidos evidenciam a importância de utilizar técnicas avançadas de aprendizado de máquina para predições de YLL. Enquanto modelos tradicionais, como ARIMA e SARIMA, possuem a vantagem de serem interpretáveis e fáceis de implementar, sua performance inferior em comparação a XGBoost e LSTM destaca as limitações dessas abordagens em dados mais complexos.

O XGBoost demonstrou ser uma alternativa robusta e eficiente, apresentando métricas de erro próximas às do LSTM. No entanto, a habilidade do LSTM de capturar dependências de longo prazo, combinada com sua flexibilidade para lidar com não linearidades, fez com que fosse o modelo mais adequado para o problema abordado. A implementação do LSTM em um pipeline integrado permite predições automatizadas e atualizáveis, superando desafios relacionados a atrasos na disponibilização de dados.

Além disso, a abordagem de *nowcasting*, facilitada pelo pipeline e pelos modelos mais avançados, fornece informações em tempo real para gestores de saúde, permitindo a identificação precoce de áreas vulneráveis e o planejamento de políticas mais eficazes.

O modelo LSTM foi identificado como a melhor opção para prever os Anos de Vida Perdidos por Morte Prematura em municípios brasileiros de médio porte. Sua superioridade em termos de precisão e estabilidade destaca o potencial das redes neurais em análises de saúde pública. A integração do LSTM ao pipeline de dados automatizado proporciona uma solução prática e escalável, contribuindo para o avanço da gestão da saúde pública no Brasil.

5. Conclusão

Este estudo abordou a importância dos Anos de Vida Perdidos (YLL) como um indicador essencial para a saúde pública no Brasil, especialmente em um cenário marcado por desigualdades regionais. A pesquisa evidenciou que modelos preditivos podem auxiliar gestores públicos na tomada de decisões estratégicas, contribuindo para intervenções mais eficazes e alocação eficiente de recursos.

A principal contribuição foi o desenvolvimento de um pipeline automatizado para coleta, transformação e organização de dados, eliminando inconsistências e otimizando o processo de preparação. Esse pipeline é replicável e adaptável, permitindo que os pesquisadores concentrem seus esforços na análise e modelagem, além de integrar predições em tempo real para apoiar decisões rápidas.

Entre os modelos testados, o LSTM destacou-se como o mais eficiente, superando métodos tradicionais (ARIMA e SARIMA) e modernos (XGBoost e Prophet) ao lidar

com padrões não lineares e dependências de longo prazo. No entanto, sua complexidade requer recursos computacionais mais robustos, sendo importante avaliar sua viabilidade em diferentes contextos.

Para trabalhos futuros, sugere-se ampliar a análise para municípios de diferentes portes, incorporar variáveis socioeconômicas e ambientais, e explorar novas arquiteturas de aprendizado de máquina. Além disso, adaptar os modelos para refletir os impactos pós-pandemia da COVID-19 será essencial para capturar as mudanças nos padrões de mortalidade.

Com essas contribuições, este trabalho reforça o potencial das técnicas de aprendizado de máquina para melhorar a gestão em saúde pública, fornecendo ferramentas robustas e inovadoras para a previsão de indicadores críticos e a mitigação de desigualdades regionais.

Referências

Costa, A. J. L. (2007). A aplicação dos Anos de Vida Perdidos como medida de carga de doença: limites e possibilidades. Revista Brasileira de Epidemiologia, 10(supl. 1), 68-77.

França, E., de Abreu, D. M. X., Rao, C., & Lopez, A. D. (2017). Evaluation of cause-of-death statistics for Brazil, 2002-2004. International Journal of Epidemiology, 37(4), 891–901.

Instituto Brasileiro de Geografia e Estatística (IBGE). Critérios de classificação de municípios. Disponível em: https://www.ibge.gov.br.

Ministério da Saúde. (2009). Portaria SVS nº 116, de 11 de fevereiro de 2009. Dispõe sobre os prazos para a disponibilização de dados de mortalidade. Brasília, DF.

Murray, C. J. L., et al. (2021). Global burden of disease study: A comprehensive assessment of mortality and disability. The Lancet, 396(10258), 1135-1160.

Rajkomar, A., Dean, J., & Kohane, I. (2018). Machine learning in medicine. New England Journal of Medicine, 380(14), 1347-1358.

HU, J. et al. (2021). Life loss of cardiovascular diseases per death attributable to ambient temperature: A national time series analysis based on 364 locations in China. Science of the Total Environment.

LI, J. et al. (2021). Short-term effects of ambient nitrogen dioxide on years of life lost in 48 major Chinese cities, 2013–2017. Chemosphere.

CHENG, J. et al. (2021). Lower-than-standard particulate matter air pollution reduced life expectancy in Hong Kong: A time-series analysis of 8.5 million years of life lost. Chemosphere.

GOMEZ-CRAVIOTO, D. A. et al. (2024). Data Analysis and Forecasting of the COVID-19 Spread: A Comparison of Recurrent Neural Networks and Time Series Models. Cognitive Computation.