INSTITUTO FEDERAL DE SANTA CATARINA

ANDRÉ LUIZ FARACO MAZUCHELI

Aplicação dos Conceitos de Engenharia de Aprendizado de Máquina em Produção em um Sistema de Detecção de Anomalias

São José - SC

dezembro/2022

RESUMO

Com o avanço da tecnologia, cada vez mais fica evidente o seu impacto em nosso cotidiano. Dentre estas tecnologias, uma que se destaca é o *Machine Learning*. Em sua imensidão de aplicações, existem muitos desafios a serem enfrentados quando se trata da engenharia dos dados utilizados em cada sistema, e também muitos obstáculos que surgem quando implantado em produção. Estima-se que finalizar a modelagem e definição dos dados, pode representar menos da metade do que seria a primeira versão estável, confiável e eficiente do sistema. Este trabalho visa aplicar conceitos de MLOPs, em cenário de produção, explorando cada etapa do ciclo de vida de um sistema de Machine Learning, com o intuído de definir boas práticas e ferramentas para auxiliar no processo.

Palavras-chave: MLOps. Machine Learning. Engenharia de dados.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API Application Programming Interface.

ECG Eletrocardiograma.

HLP Human Level Performance.

IA Inteligência Artificial.

LGPD Lei Geral de Proteção de Dados Pessoais.

ML Machine Learning.

MLOps Machine Learning Operations.

POC Proof of Concept.

SO Sistema Operacional.

TFX TensorFlow Extended.

VM Virtual Machine.

SUMÁRIO

1 INTRODUÇÃO

Modelos de aprendizado de máquina, do inglês Machine Learning (ML), quando implantados em sistemas do mundo real, necessitam de flexíbilidade, pois existe uma variação dos dados de entrada com a variação do tempo em ambiente de produção. O desempenho do modelo em produção se degrada devido a estas frequentes nos dados mudanças. Monitorar o modelo se torna uma perspectiva essencial e responsável por definir se existe a necessidade de treiná-lo novamente e garantir que os modelos estejam funcionando conforme o esperado. Porém, acompanhar o desempenho de um modelo trás diversos desafios ao longo do ciclo de vida do aprendizado de máquina (GARG et al., 2021).

Sistemas de aprendizado de máquina, possuem um ciclo de vida que tende a melhorar gradualmente sua eficiência. O ciclo pode ser dividido em 4 estágios básicos, sendo eles: **escopo**, **dados**, **modelagem** e **implantação**, podendo os três últimos, ser iterativos entre si. Cada etapa, possui uma série de processos e análises, que conforme são aperfeiçoados, melhoram a precisão e o desempenho do sistema.

Dentro deste contexto, podemos trabalhar com o conceito de engenharia de aprendizado de máquina, que também pode ser chamado de Machine Learning Operations (MLOps). Ele agrega um conjunto de ferramentas e princípios para apoiar o progresso ao longo do ciclo de vida de um projeto de ML, principalmente referente às etapas de dados, modelagem e implantação. A ideia central em MLOps é encontrar maneiras de pensar sobre o escopo de modelagem e implantação de dados e também ferramentas de software para sustentar as melhores práticas. Ele permite que os desenvolvedores envolvidos no projeto colaborem e aumentem o ritmo em que os modelos de IA podem ser desenvolvidos, implantados, modelados, monitorados e retreinados (ASHMORE; CALINESCU; PATERSON, 2021).

A importância deste conceito, está no fato de que quando um algoritmo de predição é utilizado em um sistema de ML, ele ainda precisa passar por um processo de amadurecimento em produção, por fluxo iterativo do algoritmo, sendo ele retreinado e ajustado para eventuais alterações em parâmetros não considerados previamente. Este processo, pode não ser gerido e nem implementado muitas vezes de forma coerente.

A forma como é realizado o processo de implantação do sistema de ML em produção, esta diretamente associada ao seu nível de desempenho, portanto sua estrutura deve ser cuidadosamente modelada para o cenário de aplicação.

Definir se o seu funcionamento ocorrerá em tempo real ou não, ou se ele será executado em nuvem, ou em *edge*, com o hardware e software em algum servidor local.

Nestes cenários, é relevante considerar questões como latência de comunicação, que de fato pode impactar consideravelmente, segurança de dados também devem ser consideradas relevantes, já que o sistema trabalha com a análise de dados das mais diversas naturezas(NG, 2022a).

1.1 Objetivo geral

Este trabalho possui o objetivo geral de aplicar conceitos de engenharia de aprendizado de máquina em produção, utilizando um modelo de detecção de anomalias, definindo boas práticas para gerenciar e realizar ajustes em toda a infraestrutura definida.

1.2 Objetivos específicos

Visando atingir o objetivo geral, os seguintes objetivos específicos deverão se aplicados:

1. Pesquisar e entender os conceitos de MLOps;
2. Descrever o ciclo de vida de um sistema de ML em produção;
3. Criar a infraestrutura para implantação do sistema;
4. Definir aplicação e base de dados utilizada;
5. Gerar o modelo de detecção de anomalias;
6. Colocar o modelo em um cenário de produção;

2 FUNDAMENTAÇÃO TEÓRICA

Este capitulo tem a finalidade de fundamentar e explorar os conceitos e tecnologias utilizadas neste trabalho, para contextualizar a ideia e consolidar as motivações presentes nesta aplicação.

2.1 MLOps

Conforme a comunidade de ML evolui e se consolida, manifesta-se uma tendência de que desenvolver e implantar sistemas que utilizam desta tecnologia é relativamente rápido e barato, mas mantê-los ao longo do tempo, encarece e aumenta a complexidade do projeto (SCULLEY et al., 2015).

A implantação de modelos em aprendizado de máquina vem evoluindo consideravelmente nos últimos anos, e tem sido uma área crescente de estudos. este processo pode ser visto como semelhante ao estabelecido para a implantação no desenvolvimento de *software* tradicional (GARG et al., 2021).

Assim que desenvolvido, o sistema em produção deve funcionar ininterruptamente com o custo mínimo e, simultaneamente, produzir o desempenho máximo. Quando o sistema é implantado, surge um novo fator que afeta diretamente todo o sistema, que são os casos práticos problemas do mundo real.

Compreender os conceitos de ML é essencial, mas construir uma carreira de IA eficaz e consolidada, existe a necessidade de compreender e dominar os conceitos e recursos da engenharia de aprendizado de máquina em produção MLOps(NG, 2022a).

O MLOps combina os conceitos de ML com conceitos funcionais de desenvolvimento de software para nortear a elaboração do projeto de ML, ja o preparando para a implantação em produção.

Desta forma, MLOps abrange como conceituar, construir e manter sistemas operados continuamente em produção utilizando ferramentas e metodologias bem estabelecidas para garantir a eficácia e eficiência. Diferentemente de modelagens de aprendizado de máquina padrão, concebidas em ambiente de desenvolvimento, os sistemas, quando em produção, precisam lidar com dados em constante evolução e mudanças, sejam elas graduais ou repentinas.

Objetivo do MLOps:

1. Rápida definição e desenvolvimento de modelo;

2. Rápida implantação de modelos atualizados em produção;
3. Garantia de qualidade;
4. Facilitar a comunicação entre o cientista de dados e o Engenheiro de ML;

MLOps é um procesos que permite que cientistas de dados e equipes de desenvolvedores colaborem de forma otimizada para aumentar o ritmo de desenvolvimento de modelos, juntamente com CI/CD, monitoramento e validação.

A complexidade em MLOps esta no fato de não existir processos estáticos, necessita de um engajamento com as necessidades específicas de cada projeto tendo como referência, os conceitos, as boas práticas e as ferramentas de apoio.

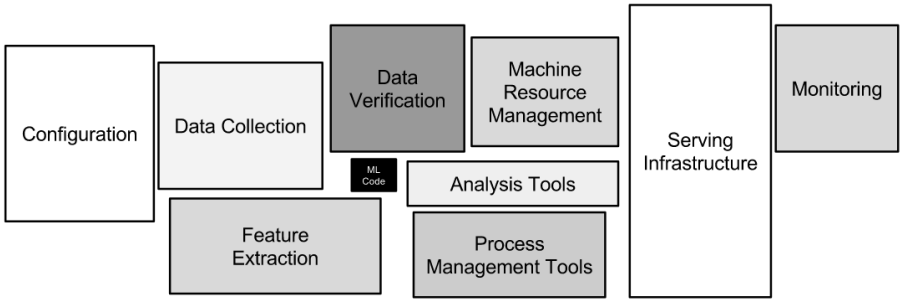
2.1.1 Componentes da infraestrutura de um sistema de aprendizado de máquina

Para garantir um sistema de aprendizado de máquina ML robusto e eficiente em produção, é necessário desenvolver uma infraestrutura para o projeto que garanta tais características.

O núcleo do sistema de fato é entregue pelo parte de codigo ML, e existem diversos algoritmos já implementados e testados pela comunidade científica para diferentes tipos de aprendizado de máquina, cada um com sua vantagens, desvantagens e peculiaridades.

Porém confomre ilustra a Figura 1, quando se trata de um sistema de ML em produção, o código em si representa uma pequena fração de toda a infraestrutura, podendo represenstar algo entre 5-10%, de todo o conteúdo implantado, diferentemente de um ambiente de desenvolvimento ou mesmo um ***notebook Jupyter*** por exemplo (SCULLEY et al., 2015).

Figura 1 – *Infraestrutura de um sistema de aprendizado de máquina*



Fonte:Sculley et al. (2015)

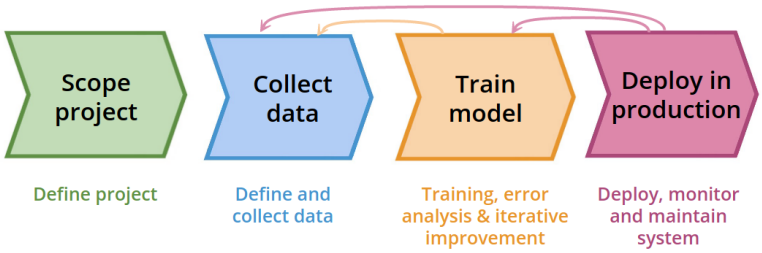
Isso pode impactar em um grande diferencial de realizar uma Proof of Concept (POC) em ambiente de desenvolvimento, ou seja, quando apenas o algoritmo em si esta

aprovado antes de ser implantado em produção, sem consider outros fatores que afetam o desempenho do projeto.

2.1.2 Ciclo de vida de um sistema de aprendizado de máquina em produção

O ciclo de vida de um sistema de ML, conforme ilustra a Figura 2, é um processo iterativo e complexo cujo papel é seguimentar e nortear o processo de ingestão de dados, que se inicia com a coleta dos dados usados para treinar um algoritmo de ML e se encerra um ciclo com a implantação desse sistema em produção (ASHMORE; CALINESCU; PATERSON, 2021). A contextualização detalhada de cada etapa do ciclo é fundamental para consolidação dos conceitos aplicados neste trabalho.

Figura 2 – *Ciclo de vida*



Fonte:Ng (2022b)

2.1.2.1 Escopo

Conforme ilustra a Figura 2, o ciclo é iniciado pelo **escopo do projeto**, esta etapa tem por objetivo a especificação e o planejamento do projeto. A maior importância nela é realizar questionamentos fundamentais para o seu desenvolvimento.

Questões como *"Quais são os recursos em termos de dados, tempo, pessoas são necessários para o desenvolvimento do projeto?*, ou *"Quais são as métricas de sucesso do projeto?"*. Desta forma, o processo de **Escopo do projeto**, esta diretamente vinculada a procurar um problema de negócio para resolver com Inteligência Artificial (IA), sempre visando analisar a viabilidade técnica e valor.

As métricas mais importantes a serem analisadas estão relacionadas ao sistema de ML, que estão associadas a precisão do algoritmo ao realizar seu proposito, de *software*, que estão relacionadas com cenário técnico, latência na comunicação, recursos da infraestrutura, e também os recursos necessários em termos de mão de obra qualificada, e os próprios dados a serem adquiridos, que muitas vezes não são obtidos em um processo simples ou barato.

Além disso, deve ser realizada uma análise minuciosa em relação às métricas de negócio, visando definir se o projeto de fato é viável em termos de investimento.

Definir estes parâmetros nos estágios iniciais do desenvolvimento de um sistema de aprendizado de máquina, resulta em maiores simplificações e otimizações nas etapas adjacentes do ciclo de vida, reduzindo a possibilidade de ocorrer imprevistos (NG, 2022a).

2.1.2.2 Dados

Tradicionalmente, esta é a etapa do ciclo mais consome tempo do desenvolvimento. Os dados passam por um processo de limpeza e rotulagem, que podem vir de várias fontes e formatos diferentes, e geralmente, necessita de um grau de normalização rígido.

Existe um processo de otimização na coleta dos dados, mesmo após o sistema já estar em produção, a necessidade a medida em que a qualidade dos dados melhora, é desejável utilizá-los para criar um modelo. Esta é a primeira necessidade de MLOps.

Pode ser realizada uma divisão, definindo um limiar que determina se o conjunto de dados é pequeno, do inglês *small data* ou grande, do inglês *big data*. O valor é relativo, porém para este trabalho, foi estabelecido que conjuntos com mais de 10.000 dados, são considerados *big data*.

Para cenários em que se aplique um *conjunto pequeno de dados*, é uma boa prática estabelecer um processo de rotulagem bem definido. O conjunto pode ser examinado manualmente e os rótulos corrigidos e refatorados facilmente devido ao pequeno volume de dados. Nos casos de se utilizar um *big data*, deve-se focar no processamento dos dados, por ser mais complexo o processo de rotulagem.

O problema da rotulagem, é que dependendo da situação, pode ser que torne a análise dos dados um processo mais complexo e custoso para o sistema. Para evitar este problema, é uma boa prática identificar possíveis redundâncias em certos cenários rotulados.

Para exemplificar, pode ser considerado um cenário em que um sistema de predição analisa *smartphones* em uma fábrica, sendo o seu objetivo, identificar possíveis imperfeições nos produtos ao final do seu ciclo de produção. Imaginando que os dados possuem duas rotulagens que identificam arranhões profundos e arranhões suaves, como a rotulagem identifica a mesma classe de imperfeição, é indicado em alguns cenários que seja realizada uma mesclagem, para otimizar a leitura dos dados, simplificando o processo, eliminando uma redundância de rótulos atacando o mesmo cenário.

Nesta etapa, é altamente recomendado não perder muito tempo com os dados, e fazer o mais rapidamente possível o ciclo de interação ilustrado na Figura 3, pois provavelmente mais inconsistências e situações não previstas surgirão nele.

Para realizar o pré-processamento dos dados, é recomendado desenvolver e implementar *scripts* que sejam replicáveis, de modo que sua validação seja flexível e ajustes sejam executados de forma ágil, lembrando que este é um processo iterativo, que pode ser repetido inúmeras vezes.

Outra metodologia importante presenta nesta etapa é o uso de metadados, que para contexto pode ser representados como os dados que referenciam os dados. Como exemplo, pode-se imaginar, no cenário da fábrica de *smartphones* que analisa imperfeições nos produtos utilizando fotos deles, os dados seriam as fotos e os rótulos dos arranhões. Os metadados seriam os dados que informam, por exemplo o horário em que a foto foi tirada, o número de linha do produto na fábrica, configurações da câmera que bateu a foto.

Através deles é possível identificar algum tipo específico de situação que está resultando em erro, como, por exemplo, identificar que uma linha específica de produtos, e uma única fábrica está dando mais erros do que os demais produtos analisados no algoritmo, ajuda a solucionar e otimizar o modelo na análise de erros, e detecção de efeitos inesperados.

2.1.2.3 Modelagem

Sistemas de IA, em sua essência, são compostos de código, que representa o algoritmo e o modelo, em conjunto com os dados. Ao construir um sistema de ML, existe a necessidade de ter um modelo, e então o modelo é treinado nos dados, gerando o algoritmo de predição.

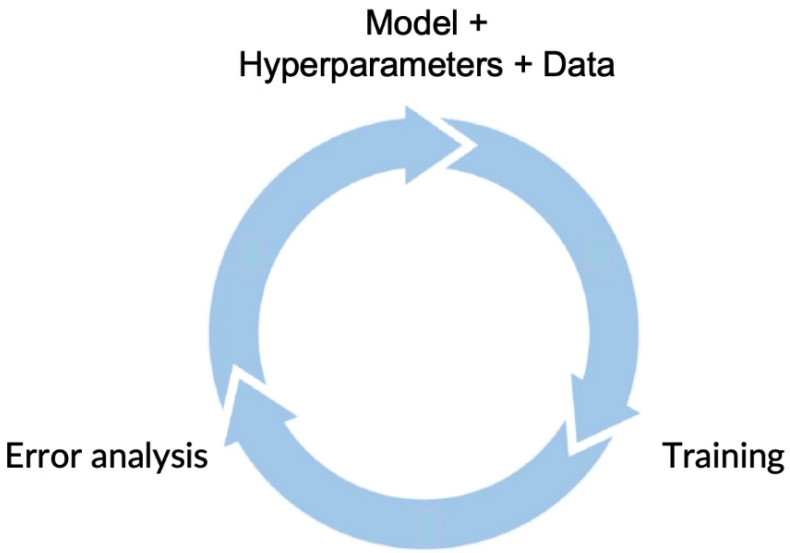
Existem também os *hiperparâmetros*, entradas adicionais, importantes para validar se existe uma taxa de aprendizado bem ajustada e um parâmetro de regularização adequado, este processo é chamado também de *tuning*.

Na Figura 3, esta sendo ilustrado um ciclo de modelagem que ajuda a identificar como melhorar ou os dados, ou os hiperparâmetros, ou os modelos. O fato desse ciclo ser executado muito rapidamente, é a chave para melhorar o desempenho, e aqui se encontra o grande papel do MLOps, otimizar este processo. Após validado que um modelo adequado foi alcançado, é então feita a implantação.

Mesmo possuindo uma média baixa de erros no ciclo, nem sempre implica no sistema atender às regras de negócio para o projeto. Este problema pode ser ilustrado através de um exemplo.

No caso de realizar uma pesquisa no ***Google***, sobre alguma receita que se deseja fazer, não é garantido que todos os resultados, realizados através do motor de busca, sejam os das melhores receitas, ou que retorne a melhor receita que você procura, porém, não houve erro no processo, a busca foi feita baseada nos dados. Agora imaginando uma busca pela palavra *YouTube* no *Google*, o motor de busca identifica que você procura por

Figura 3 – *Ciclo de modelagem*



Fonte:Ng (2022b)

www.youtube.com.

Quando o usuário possui uma busca clara, fica fácil para o algoritmo identificar o melhor resultado da busca para exibir, aqui então se aplica o conceito de desproporcionalidades de relevância das informações, então neste caso, mesmo se a média de erro de precisão estiver baixa, que trata todos os parâmetros como iguais. Que no caso, o acerto no cenário de busca por *YouTube*, que foi muito mais preciso, teve o mesmo peso do cenário da receita, mesmo sendo um resultado de busca menos preciso.

Para tratar este problema, existe uma metodologia cujo objetivo é realizar análises em fatias-chave, do inglês *key slices* dos dados. Digamos que foi feito um sistema de ML para decidir quais usuários provavelmente vão pagar um empréstimo, e assim então recomendar o empréstimo a instituição financeira que o fará.

Este sistema, muitas vezes por lei, não considera dados de cliente como sexo, etnia, cor, mesmo que coincidentemente, com a média de erros baixa, o sistema criar tendencias de indicar que pessoas negras, por exemplo, possuem mais probabilidade de não pagar o empréstimo, e consequentemente o sistema não indicar o empréstimo, isso indica um problema no negócio.

Outra analogia pode ser feita com um sistema de *e-commerce* que pode indicar produtos de apenas marcas maiores e não as menores, o que pode ser prejudicial para o negócio, considerando que existe demanda para todas as marcas.

Uma situação semelhante, esta associada ao conceito de classes raras, do inglês, *rare classes*, o aparecimento de situações raras, em que mesmo com uma boa precisão da análise, podem ser ignoradas, pois ocorrem muito raramente.

Este caso esta diretamente associado a ideia de distribuição distorcida dos dados. Contextualizando para um sistema que visa identificar uma doença através de dados dos exames de pessoas, onde 99% da população não possui ela e 1% possui, o algoritmo mesmo sendo muito preciso, com margem de erro de apenas 1%, por exemplo, poderá afirmar que 0% da população possuem a doença, pela margem de erro, segue uma precisão alta, porém a informação pode ser desastrosa e se torna de certa forma inútil. O objetivo do MLOps nestes cenários, é garantir que o sistema de ML desenvolvido, atenderá as regras de negócio.

Em todo o processo de modelagem, observa-se que o objetivo de aumentar a precisão do algoritmo é constante. Porém, é necessário ter cautela ao definir qual limiar identifica que o nível de acurácia é adequado.

Para isso, deve-se observar novamente o tipo de dados com que se esta trabalhando, e as formas em que estão organizados, que podem ser basicamente divididas em duas, estruturados e não estruturados, e esta informação ajuda a parametrizar o uso de uma *baseline* para ter referências de precisão em um algoritmo, indicada ser utilizada para ter sucesso a longo prazo com o projeto de ML. Este projeto terá foco na análise de dados não estruturados.

Dados estruturados são os *BigDatas*, planilhas com grandes volumes de informação, em que humanos não são tão confiáveis para analisar. E dados não estruturados, tendem a ser dados que humanos são bons em interpretar, como imagens, áudios e textos. Neste caso, utilizando como *baseline* o conceito de Human Level Performance (HLP), tende a ser uma ótima ideia, pois humanos são confiáveis nestas interpretações e serão uma ótima referência para validar a precisão do algoritmo.

Para estabelecer uma *baseline* adequada para cada caso, além de priorizar o HLP para dados não estruturados, também é uma boa prática, realizar pesquisas na literatura, ou examinar resultados em testes implementados em projetos de código aberto. Também pode ser construída por testes em produção do mesmo modelo com versões mais simples e compactas, para através dos resultados, ter como referência neveis adequados de precisão. Para este caso, a agilidade ao implantar o sistema em produção facilita de forma considerável o processo.

É importante compreender que antes de realizar o treinando do algoritmo com milhares de dados, é mais eficiente treinar com conjuntos de dados menores, que podem ser realizados mais rapidamente e assim encontrar *bugs* mais genéricos de forma mais rápida, pois, para o caso de um sistema que realiza reconhecimento de imagens, se o algoritmo não funcionar com 100 imagens no treinamento, ele certamente não funcionará com 1000 imagens, ou 10000.

Então, uma das coisas mais importantes antes de treinar o modelo de ML, é como

realizar a análise de erros para ajudar a decidir como melhorar o desempenho do algoritmo.

Figura 4 – *Rotulagem de erros*

Example	Label	Prediction	Car noise	People noise	Low bandwidth
1	"Stir fried lettuce recipe"	"Stir fry lettuce recipe"	1		
2	"Sweetened coffee"	"Swedish coffee"		1	1
3	"Sail away song"	"Sell away some"		1	
4	"Let's catch up"	"Let's ketchup"	1	1	1

Fonte:Ng (2022b)

Na Figura 4, esta sendo ilustrado uma boa prática para realizar a análise dos erros. O contexto é a rotulagem de erros que ocorrem, em um sistema que faz o reconhecimento de voz, e transforma em texto.

Em cada caso, em que foi identificado que o algoritmo, realizou a interpretação da frase de forma equivocada. Existem rótulos que podem estar associados ao possível motivo da distorção, sendo eles ruido de carros, do inglês *car noise*, ruido de pessoas, do inglês *people noise* ou baixa largura de banda, do inglês *low bandwidth*, que pode representar perda na taxa de dados da comunicação.

Esta rotulagem serve para mapear as causas dos erros do algoritmo, e facilitar sua resolução, vale ressaltar que mais de um ruído, pode estar envolvido e um mesmo erro neste cenário. Para segmentar em categorias e analisar cada caso, e definir prioridades, considerando a quantidade de vezes que um determinado ruído aparece.

Por exemplo, se em todo o conjunto de dados, a tag *car noise* representar 2% do conjunto, é mais irrelevante do que se a tag *people noise* representar 80%. Esse processo é iterativo, e pode ocorrer diversas vezes, e conforme se aprofunda na análise, podem ser adicionadas novos rótulos.

Para parametrizar esta análise, alguns cenários devem ser questionados como, quanto espaço para melhoria existe, ou com que frequência essa categoria de rótulo aparece, o quão fácil é melhorar a precisão nessa categoria ou o quão importante é melhorar essa categoria. São questionamentos que auxiliam na tomada de decisão.

Quando no caso de selecionar um rótulo para trabalhar na melhoria, é indicado coletar mais dados associados a esta categoria, como, por exemplo, se ocorrer muitas falhas por ruído de carros, é desejável ter mais dados de áudio com ruídos de carros, pois em ML, sempre se deseja possuir mais dados, o que em muitos casos pode ser custoso conseguir, porém, com estas práticas, os dados que se desejam são mais seletivos, podendo ser coletados em menos volume, focado na categoria em que desejamos aumentar a precisão. A classificação das categorias de rótulos fornecem um norte para concentrar o aumento de entrada de dados nos campos que realmente farão mais efeito.

Nas últimas décadas, muito tem se discutido por estudiosos da área de como melhorar o código, de fato, muitas pesquisas de foram desenvolvidas realizando *downloads* de *datasets* existentes, e trabalhando na escolha de um modelo que funcione bem neles.

Mas para muitas aplicações, existe a flexibilidade de alterar os dados. Existem muitos projetos em que o algoritmo ou o modelo são basicamente um problema, um modelo pronto e genérico, muitas vezes funcionará bem o suficiente, e é mais eficiente gastar mais tempo aprimorando os dados, pois geralmente eles terão de ser mais customizados para cada modelo de negócio e cenário.

Nesta etapa, está presente a escolha do algoritmo de ML que melhor atenderá as necessidades, bem como seu processo de treinamento. Também nesta etapa, é definido se existe a necessidade da inserção de mais dados no sistema, indicando que retorne para a etapa anterior de 2.1.2.2. Este processo iterativo de entre os dados e a modelagem é extremamente influente no resultado final do sistema a ser desenvolvido.

Este conjunto de processos executados nesta etapa também é chamado de "Experimentation", é semelhante ao de experimentação de tentativa e erro, com várias combinações de algoritmos, recursos e hiperparâmetros diferentes até obter a combinação que resulte em um modelo adequado para o trabalho proposto.

Neste caso, as tentativas que resultarem em erros, são extremamente importantes, pois através da análise delas é que são definidos os novos conjuntos de combinações a serem testados. Aqui surge um novo conjunto de necessidade para MLOps, rastrear as métricas, do inglês *track metrics*, das execuções do experimento, que representa, registrar as informações em cada teste, e mapear métricas baseado na ideia de ação e reação.

Estas tarefas, desde a experimentação, até a otimização estão fundamentadas no pipeline de construção do sistema como parte do MLOps. Uma vez que o cientista de dados cria o modelo com uma eficácia aceitável, o modelo precisa ser implantado em produção.

2.1.2.4 Implantação

A etapa de implantação em produção, além de ser fundamental para o processo de amadurecimento do projeto, é também a fase em que surgem diversos desafios, que podem ser divididos em duas categorias. A primeira está associada ao surgimento de problemas estatísticos. Na segunda, problemas associados ao software e ferramentas utilizadas para desenvolver o projeto.

Nesta etapa, é possível analisar de forma empírica se existe a necessidade de realizar mais ajustes em etapas anteriores do ciclo de vida. O mais importante é compreender como os dados variam com o tempo, e dentro desta variação, podem existir dois cenários. O primeiro, remete a variação gradual, do inglês *gradual change*, que representa uma

mudança suave ao longo de um período, pode-se fazer uma analogia com a variação que um idioma sofre ao longo do tempo, com a adição de termos, e ajustes linguísticos.

O segundo, caracteriza uma mudança repentina, do inglês *Sudden Shock*, que representa uma variação súbita em um curto período. Pode-se fazer uma analogia com uma pessoa anonima, que viraliza na internet, e passa a se tornar famosa muito rapidamente.

Outra analogia, é que com o incidente da pandemia do Covid-19, muitas pessoas começaram a fazer mais compras online, utilizando muito mais o cartão de crédito, caracterizando uma mudança repentina em pessoas que não utilizavam com muita frequência. Desta forma, alguns sistemas antifraude baseado em ML, realizaram alerta de fraude de forma equivocada, pois estavam em produção, e em seu desenvolvimento, a pandemia não havia sido um parâmetro de análise.

A presença destes problemas estatísticos de variação nos dados, são melhor compreendida através dos conceitos de *desvio de dados* e *desvio de conceito*.

O termo *desvio de conceito*, do inglês *concept drift*, refere-se ao mapeamento desejado de um dado **X** para **Y**. Analogamente pode-se definir que **X** seja o tamanho de uma determinada residência, e **Y** seja o seu preço. Com a presença da inflação e outros indicadores econômicos, o preço da casa (Y), sofrerá variação, porém seu tamanho (**X**) será constante.

Outro exemplo para melhor consolidação do conceito de *concept drift*. O comportamento de compra de clientes ao longo do tempo pode ser influenciado pela força da economia de seu país ou região, porém, o poder aquisitivo não é especificado nos dados, e sua variação pode impactar na frequência de compra do cliente. Esses elementos definidos como *contexto oculto*, do inglês *hidden context*.

O conceito de *desvio de dados*, do inglês *data drift*, é usado para descrever a variação na distribuição dos dados de entrada. No exemplo, com o mapeamento em que **X** seja o tamanho de uma determinada residência, e **Y** seja o seu preço, um cenário de *data drift*, pode ser considerado quando o tamanho das casas (**X**), de fato passe a variar, porém, o preço (**Y**) se mantém constante.

É importante prever e gerenciar essas duas variações, e como auxílio, existe um *checklist* a se fazer para melhor modelar a estrutura do sistema de predição em produção:

Tempo real ou não: Extremamente importante parametrizar se a predição será feita na em tempo real ou se será executada através de uma coleta de dados durante um período, e após isso, elaborado um relatório preditivo pelo sistema.

Cloud VS Edge: Definir se o projeto será executado em nuvem, ou em *edge*, em algum servidor local, é vital para melhor especificar a infraestrutura, e definir quais ferramentas de apoio utilizar, bem como suas configurações. A forma de implementar a

metodologia CI/CD, em cada caso, se torna diferente, e a parametrização de indicadores da latência na comunicação, por exemplo, também sofre grande variação em cada respectivo cenário. Considerando que o sistema trabalha com a análise de dados das mais diversas naturezas, a manipulação e segurança deles, deve ser estruturada de forma coerente em cada caso, em nuvem ou em servidor local, as melhores práticas se tornam distintas e peculiares, apresentando a necessidade da especificação detalhada do cenário.

Recursos da hospedagem do sistema: A definição e monitoramento dos recursos da infraestrutura do sistema deve ser um dos pontos de mais atenção, afinal, o sistema utilizará destes recursos para executar seu propósito. Sem o gerenciamento e a especificação adequada, o desempenho do projeto, e até mesmo a precisão das análises preditivas, podem ser comprometidas. Três parâmetros são essenciais, sendo eles **armazenamento permanente**, **armazenamento temporário** e a quantidade de **processamento** necessária.

Definir a quantidade de disco para o armazenamento permanente e memória RAM para armazenamento temporário, bem como o processamento necessário para que o sistema exerça seu proposito sem gargalos, que pode ser monitorado e especificado baseado na carga do Sistema Operacional (SO), do inglês *load average*, são essenciais.

Logging: A implementação de uma estrutura de logs, em um projeto de ML, é vital para o MLOps, para identificar problemas ou retreinar o algorítimo. Segue-se a mesma proposta de um sistema de logs estruturado em um software padrão, utilizado para detecção de erros. Porém, com o conceito de MLOps, as informações existentes neste logs, servirão para a reestruturação do sistema, e para tomadas de decisões sobre definição de retreinar ou não o modelo em produção, sendo um dos princípios do ciclo de vida em produção.

Segurança e privacidade: Com o surgimento da Lei Geral de Proteção de Dados Pessoais (LGPD) no Brasil, e a manifestação de outras regulamentações de dados no mundo digital, a sensibilidade do armazenamento e a segurança, tornou-se um parâmetro crucial na especificação de projetos, principalmente do ponto de vista jurídico. Com isso, como um sistema de ML é baseado em dados, sejam eles de qualquer natureza, este ponto torna-se ainda mais crítico, necessitando de uma validação cuidadosa e detalhada (NG, 2022a).

Desta forma, a implantação de um sistema de ML ocorre em dois processos, mover o projeto do ambiente de desenvolvimento, para a produção, onde ele atuará de fato, e o processo de monitoramento, com a manutenção do sistema, já em produção.

Este último esta diretamente associada a análise de desempenho do sistema de forma automatizada e com o processo de identificação de anomalias na ingestão de dados no sistema. É através deste monitoramento, que conforme a Figura 2, ocorre o retreinamento, representados por setas fluindo da etapa de 2.1.2.4 para a etapa de 2.1.2.3, que por sua

vez, flui para 2.1.2.2.

Pode ocorrer também, o fluxo da etapa de 2.1.2.4 diretamente para 2.1.2.2, definindo qual dinâmica o processo vai executar, é o tipo de conclusão que o monitoramento do sistema vai chegar baseado na análise de *logs*.

Após implantar o sistema desenvolvido em produção, é importante definir alguma proposta de validação do algoritmo de predição, antes de colocá-lo para tomar decisões em cenário real.

Para isso, existem algumas estratégias que podem mitigar riscos de modelos em produção estarem inadequados.

Uma delas é chamada de *shadow mode*, ou modo sombra em tradução literal, que consiste em utilizar de algum julgamento externo para validar conclusões do algoritmo. Esta validação é executada em paralelo ao algoritmo, e comparada baseada nas decisões. O instrumento de comparação pode ser um julgamento humano através de uma inspeção visual do cenário, por exemplo.

Definir o grau apropriado de automação do pipeline de ML é também importante, nem sempre ser totalmente automazizado, caracteriza o cenário ideal para a aplicação. A IA por exemplo, pode encaminhar informações para um humano decidir, quando ela identificar que não possui um decisão tãoo precisa. Neste cenário, existe um trabalho a ser realizado para o próprio sistema identificar estes cenários, e tomar a decisão correta.

Para isso, é importante definir o monitoramento do sistema, uma das melhores formas de realizar isso, é monitorando *dashboards*, elaborados pelo próprio sistema indicando parâmetros importantes para as tomadas de decisão, ou que definem se o projeto esta sendo executado de forma adequada.

Estes *dashboards* podem realizar comparações como carga do servidor, ou uso de recursos do ambiente, para validar a questão da infraestrutura, que pode caracterizar algum tipo de erro no software.

Para identificar possíveis problemas nas métricas de entrada e de saída, análises como comparação entre frações de valores nulos na saida, ou coeficiente de perda de dados na saída, em comparação com a entrada, pordem caracterizar falhas no algoritmo.

Após definir as métricas para analisar, é necessário definir alerta para estes limiares. Ao definir se o problema é proveniente do ambiente em que o sistema está hospedado, como carga do servidor, ou uso de memória, é necessário refatorar o software, porém no caso do problema ocorrer no algoritmo, será necessário retreiná-lo, podendo ser de forma manual ou automática.

Com isso, a complexidade e importância da etapa de **Implantação**, se mostra evidente, tornando indispensável a execução de boas práticas e ferramentas de apoio.

2.2 Ferramentas para implantação

Para implementar os conceitos de MLOps neste trabalho, serão utilizadas diversas ferramentas de apoio, cada uma associada a uma camada da estrutura definida do projeto.

2.2.1 Docker

A necessidade de ciclos de desenvolvimento cada vez mais curtos e entrega contínua cada vez mais ágil, o uso de contêineres se mostra cada vez mais convidativo em infraestruturas de sistemas de software, por serem mais flexíveis que as máquinas virtuais e fornecem desempenho quase nativo. (COMBE; MARTIN; PIETRO, 2016).

O desenvolvimento e as operações que incorporam o conceito de Integração Contínua e Entrega Contínua, do inglês, *Continuous Integration / Continuous Delivery* (CI/CD), atendem diretamente esta necessidade de agilidade na entrega de atualizações no software (GARG et al., 2021).

Integração Contínua (CI): Ajuda na gestão do tempo e permite curtos ciclos de atualização para melhorar a qualidade do software e aumentar a produtividade geral da equipe.

Implantação Contínua (CD): Ajuda na automação da implantação do software em produção e realizar verificações de qualidade através de monitoramento.

Empregar pipelines de CI/CD em uma aplicação que incorpora componentes de MLOps resulta em grandes desafios. Porém, incorporar este conceito para implantação automatizada de modelos de IA resulta em agilidade, escalabilidade, modularidade e consequentemente em grande agregação de valor ao projeto (GARG et al., 2021).

Conforme ilustra a Figura 2, no ciclo de vida do aprendizado de máquina existe um processo de reutilização dos componentes, combinações distintas de compartilhamento de informações entre eles.

A conteinerização do pipeline do sistema de aprendizado de máquina, o torna modular, escalável e facilita a personalização independente de cada componente do ciclo, sem comprometer o compartilhamento de recursos e desempenho.

Como existe o envolvimento de múltiplos *containers*, é convidativa a utilização de algum recurso para gerenciá-los, desta forma, o Docker se apresenta amplamente utilizado para a administração deles, permitindo que o desenvolvimento utilize recursos de automação de contêiner, implantação, e redes de forma otimizada.

O Docker implementa uma camada de virtualização nos *containers* basicamente fazendo o papel de uma Virtual Machine (VM), porém muito mais muito leve, e utilizando menos recursos. Ele implementa uma interface de comunicação entre o container e o *host*

em que o sistema está hospedado, controlando o uso de seus recursos, como processamento, *RAM*, *bitrate*, disponibilizando todas as bibliotecas necessárias.

Cada container esta associado a uma imagem Docker, que representa basicamente um modelo de sistema de arquivos, um container específico, gerado a partir desta imagem. Uma subcamada virtualizada é criada de processos abaixo do Docker, que está associado ao processo inicial que iniciou um determinado container.

2.2.2 TensorFlow Extended

Para implementação dos conceitos de MLOps, será utilizado o TensorFlow Extended (TFX). Como este trabalho possui objetivo de implantar um sistema de ML em produção, a relevância de utilizar uma ferramenta de auxílio para gestão dos componentes se mostra indispensável.

O TFX foi desenvolvido pela Google, e estabelece um *pipeline* específico para tarefas de aprendizado de máquina de alto desempenho em produção através de componentes que são construídos através de bibliotecas customizáveis, para melhor atender cada cenário (GOOGLE, 2022).

Seu uso será direcionado para integrar os componentes do ciclo de vida de um sistema de ML, discutidos no capítulo 2.1.2 em uma plataforma única, no qual se deseja aumentar a padronização dos componentes envolvidos, simplificar a configuração, realizar experimentos de forma mais rápida, e reduzir o tempo de operação em produção, fornecendo estabilidade e minimizando interrupções nos processos (BAYLOR et al., 2017).

Diversas empresas renomadas no mercado utilizam o TFX, entre elas serviços da Google como Gmail ou Play Store, bem como Spotify, Airbus entre outras, caracterizando grande consistência no meio de atuação (GOOGLE, 2022).

2.2.3 Fast API

Para uma interface e simplificar o gerenciamento do acesso aos resultados do sistema de detecção de anomalias desenvolvido, será utilizada a ferramenta Fast API.

É um *framework web* utilizado para construção de Application Programming Interface (API), com características de alto desempenho. Sua proposta é ser simples e com característica de viabilizar o desenvolvimento rápido de funcionalidades, muito adequada para o cenário que este trabalho propõe.

Seu desempenho é semelhante a de ferramentas conceituadas no mercado como *NodeJS* e *GO* conforme a documentação oficial (RAMíREZ, 2022), porém com caráter mais simplista.

Segundo o trabalho (BANSAL; OUDA, 2022), foi realizada uma avaliação em

resultados experimentais que mostraram que o desempenho do modelo de ML utilizando FastAPI foi melhorado em quase 45% em comparação ao Flask, outra ferramenta com proposito semelhante muito utilizada no mercado.

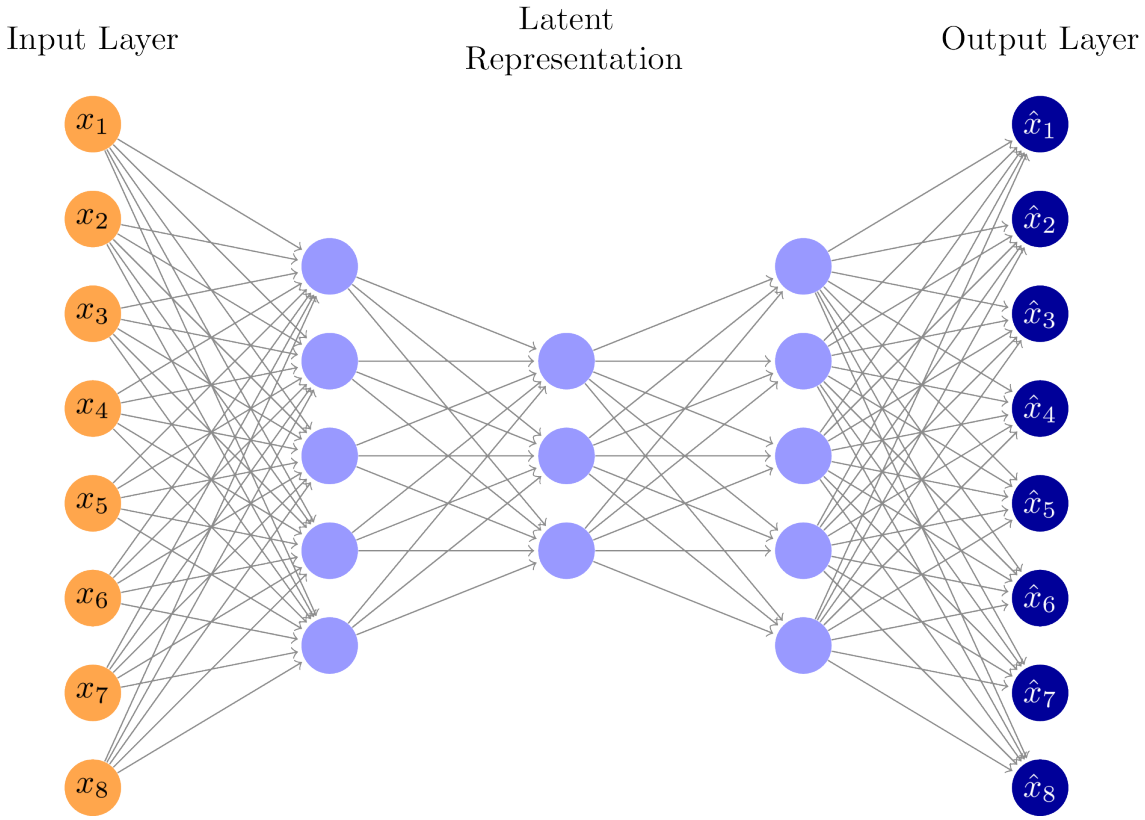
2.3 Detecção de anomalias

Em ML, em vez serem criadas regras em uma linguagem de programação, como ocorre no desenvolvimento de *software* tradicional, existe uma máquina mapeando as respostas para os dados de entrada, desta forma, a máquina descobre as regras.

Se obtiver todos os dados de entrada corretamente rotulados, basta treinar o modelo, para detectar variações.

A Figura 5, ilustra o princípio de funcionamento de um esquema de *auto-encoder*.

Figura 5 – *Auto-Encoder*



Fonte:Riebesell (2022)

Basicamente pode ser dividido em duas redes neurais, em camadas de codificação e decodificação dos dados, da esquerda para a direita respectivamente.

Os dados de entrada, possuem um alto nível de dimensionalidade, o processo de codificação, deve aprender uma maneira de representar os dados em uma dimensão muito menor, fluindo da esquerda para a direita, para compactar os dados na camada de entrada.

Este processo deve ser encerrado assim que encontrado o menor nível de representação latente das informações originais.

Após isso, a rede deve aprender como reconstruir os dados a partir da menor representação latente encontrada, no processo de decodificação, na mesma dimensionalidade dos dados originais, perdendo o mínimo de informação.

O objetivo é construir um modelo de aprendizado de máquina, que aprenda a representação dos dados de entrada, para utilizar então este modelo, em dados futuros que não se parecem com os dados de entrada do treinamento, realizando assim a detecção de anomalias.

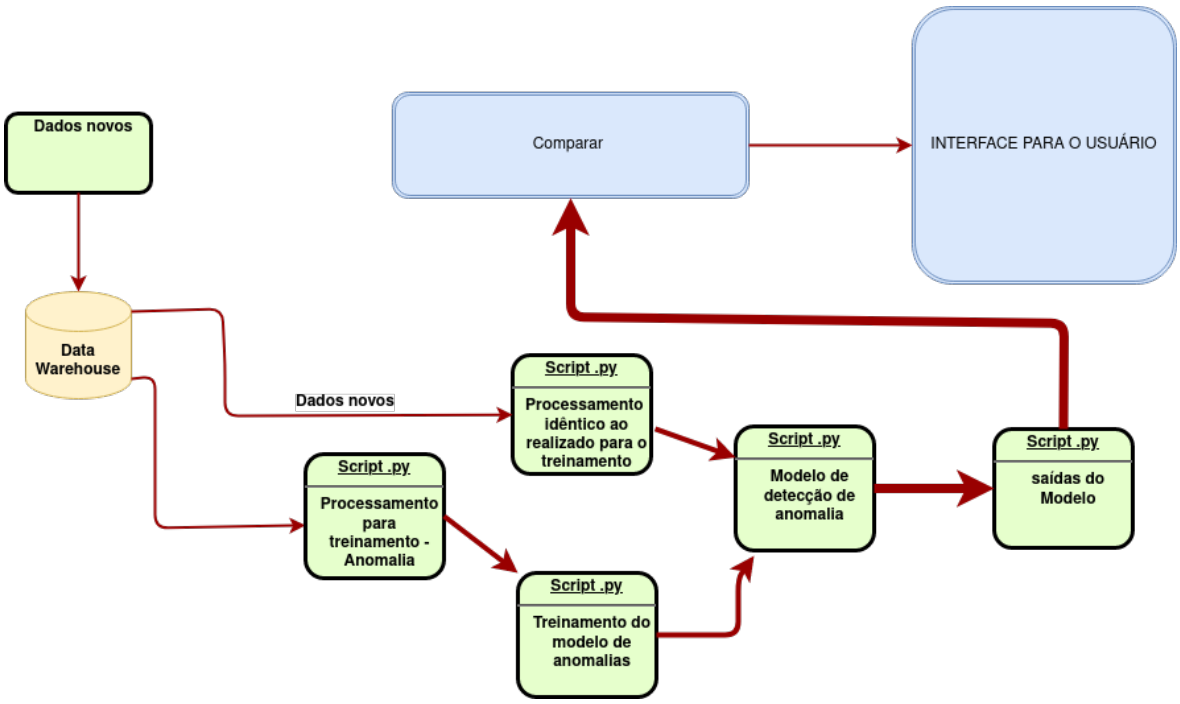
3 PROPOSTA

Neste capítulo esta sendo descrita a infraestrutura e a base de dados utilizada, bem como a proposta do sistema desenvolvido. Também o cronograma adotado para o desenvolvimento deste trabalho .

3.1 Descrição de infraestrutura e base de dados

A Figura 6, representa a arquitetura do sistema de ML deste trabalho. Um conjunto de dados de Eletrocardiograma (ECG), deve ser armazenado na base de dados nomeada de *Data Warehouse*, esta base deverá armazenar todo e qualquer dado de entrada, ao longo do tempo, considerando que um ECG é uma série temporal. Deve ser garantido que a rede neural aprenda uma representação latente do conjunto, e então o reconstrua a partir dela, para enfim medir o erro.

Figura 6 – *Arquitetura do sistema de ML*



Fonte: O próprio autor.

Os *scripts* responsáveis por realizar o processo de normalização de dados, limpeza e aprendizado de máquina, seguem em *pipelines* até uma base de dados contendo os resultados. Estes resultados deverão ser ofertados para análise através de uma interface gráfica.

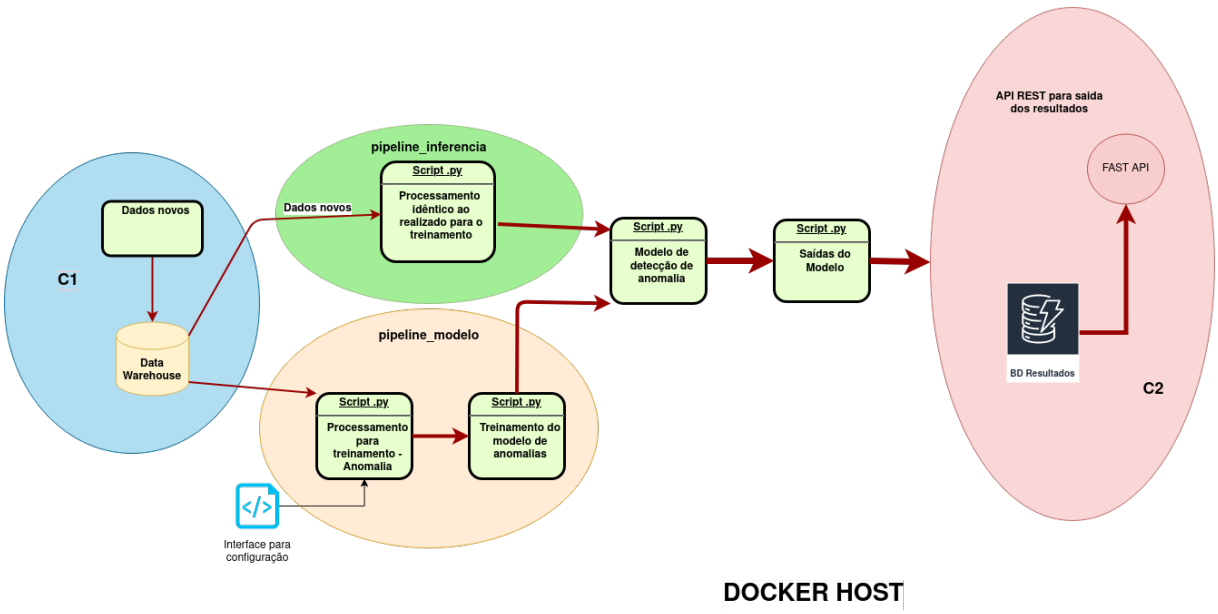
3.2 Desenvolvimento e implantação

O projeto desenvolvido visa realizar a detecção de anomalias em dados que representam ECG. Existem dois *pipelines* compostos de *scripts*, responsáveis por nortear as informações. O *pipeline* inferior, chamado de *pipeline* de treinamento, é executado primeiramente, e possui objetivo de processar os dados para treinamento e gerar o modelo de detecção de anomalias.

Após o modelo ser desenvolvido, o *pipeline* superior, chamado de *pipeline* de inferência, é executado, e dados novos são encaminhados para possíveis detecções de anomalia, conforme explicado na seção 2.3. Este processo utiliza a plataforma *TensorFolow Extended*, cujos benefícios estão destacados na seção 2.2.2.

As informações resultantes da análise são armazenadas em outra base de dados, esta possui uma API de interface para interação com os dados. Conforme ilustra a Figura 7, todo o *pipeline* possui seus componentes dockerizados, desfrutando dos benefícios abordados na seção 2.2.1.

Figura 7 – *Infraestrutura do sistema de detecção de anomalias*



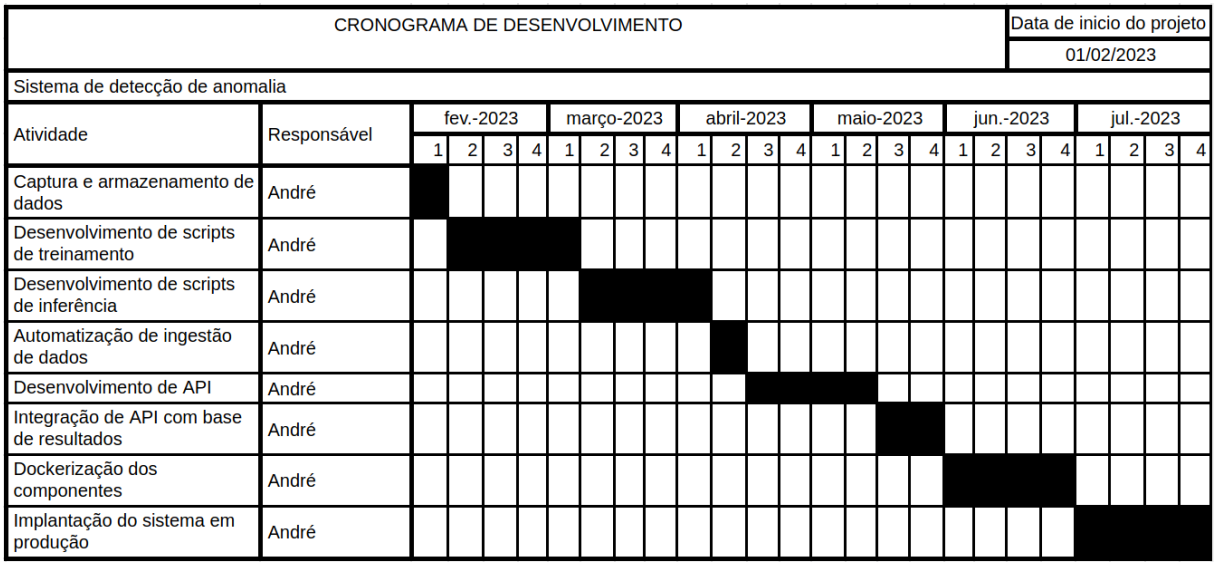
Fonte: O próprio autor.

Como ilustra também a Figura 7, a API de interface da base de dados de resultados está dockerizada de forma isolada, se comunicando com os *pipelines* de ML. O *Data Warehouse*, também está em um container separado, no início do processo, consumido para a ingestão de dados do sistema, tanto para treinamento como para inferência com dados novos.

Após todo o desenvolvimento, o projeto deve implantado em produção, seguindo os conceitos de MLOps abordados neste trabalho.

3.3 Cronograma

Figura 8 – *Cronograma de atividades*



Fonte: O próprio autor.

REFERÊNCIAS

ASHMORE, R.; CALINESCU, R.; PATERSON, C. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 5, may 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3453444>. 6, 10

BANSAL, P.; OUDA, A. Study on integration of fastapi and machine learning for continuous authentication of behavioral biometrics. In: *2022 International Symposium on Networks, Computers and Communications (ISNCC)*. [S.l.: s.n.], 2022. p. 1–6. 21

BAYLOR, D. et al. Tfx: A tensorflow-based production-scale machine learning platform. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2017. (KDD '17), p. 1387–1395. ISBN 9781450348874. Disponível em: <https://doi-org.ez130.periodicos.capes.gov.br/10.1145/3097983.3098021>. 21

COMBE, T.; MARTIN, A.; PIETRO, R. D. To docker or not to docker: A security perspective. *IEEE Cloud Computing*, v. 3, n. 5, p. 54–62, 2016. 20

GARG, S. et al. On continuous integration / continuous delivery for automated deployment of machine learning models using mlops. In: *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. [S.l.: s.n.], 2021. p. 25–28. 6, 8, 20

GOOGLE. *TensorFlow Extended*. 2022. Urlhttps://www.tensorflow.org/learnimplement-mlops. 21

NG, A. *Machine Learning Engineering for Production (MLOps)*. 2022. Urlhttps://www.deeplearning.ai/courses/machine-learning-engineering-for-production-mlops/. 7, 8, 11, 18

NG, A. *Machine Learning Engineering for Production (MLOps)*. 2022. Urlhttps://www.deeplearning.ai/resources/. 10, 13, 15

RAMíREZ, S. *Fast API*. 2022. Urlhttps://fastapi.tiangolo.com/. 21

RIEBESELL, J. *Fast API*. 2022. Urlhttps://tikz.net/autoencoder/. 22

SCULLEY, D. et al. Hidden technical debt in machine learning systems. In: CORTES, C. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015. v. 28. Disponível em: <https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf>. 8, 9