UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

Detecção de Concept Drift em MLOps

Estudo Comparativo de Métricas de Monitoramento

Danielle Tavares da Silva

UFG
UNIVERSIDADE
FEDERAL DE GOIÁS

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

DANIELLE TAVARES DA SILVA

Detecção de Concept Drift em MLOps

Estudo Comparativo de Métricas de Monitoramento

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): DANIELLE TAVARES DA SILVA

Título do trabalho: Detecção de Concept Drift em MLOps

Estudo Comparativo de Métricas de Monitoramento

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Danielle Tavares Da Silva**, **Discente**, em 05/02/2026, às 11:11, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 13/03/2026, às 11:26, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

DANIELLE TAVARES DA SILVA

Detecção de Concept Drift em MLOps

Estudo Comparativo de Métricas de Monitoramento

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Orientador: Prof. Dr. Fernando Marques Federson

Goiânia

2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UFG.

SILVA, DANIELLE TAVARES DA
Detecção de Concept Drift em MLOps [manuscrito]: Estudo Comparativo de Métricas de Monitoramento / DANIELLE TAVARES DA SILVA. - 2025.
122 f.: 2025

Orientador: Prof. Dr. Fernando Marques Federson
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. Inteligência Artificial. 2. Mlops. 3. Concept Drift.

I. Federson, Fernando Marques , orient. II. Título.

CDU 004

DANIELLE TAVARES DA SILVA

Detecção de Concept Drift em MLOps
Estudo Comparativo de Métricas de Monitoramento

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 09 de dezembro de 2025.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Prof. Dr. Sávio Salvarino Teles de Oliveira
(INF-UFG)

DANIELLE TAVARES DA SILVA

Detecção de Concept Drift em MLOps

Estudo Comparativo de Métricas de Monitoramento

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **MLOps**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.

Palavras-chave: Inteligência artificial; MLOps; Concept drift.

ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **MLOps**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.

Keywords: Artificial intelligence; MLOps; Concept drift.

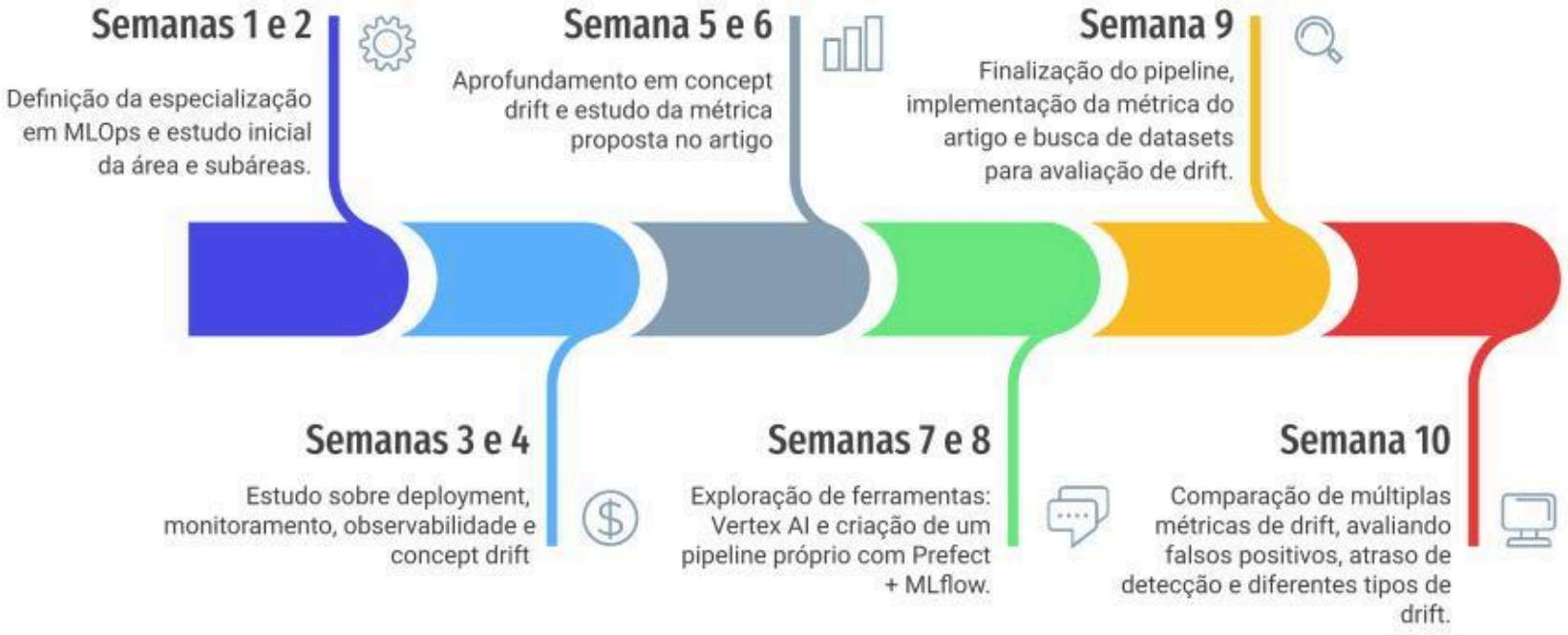
Goiânia

2025

Minha Jornada

Danielle Tavares da Silva

Especialista em: MLOps



MINHA JORNADA

Nome: **Danielle Tavares da Silva**

Especialidade: **MLOps**

Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

Minha Jornada

Nas **Semanas 1 e 2**, iniciei a busca pela área em que eu gostaria de me especializar dentro da Inteligência Artificial. Foquei em entender o que é o MLOps e em diferenciar suas principais etapas, como preparação de dados, treinamento, validação, deployment, monitoramento e re-treinamento contínuo. A cada leitura e material estudado, fui percebendo que meu interesse não estava apenas em “treinar modelos”, mas em todo o processo necessário para que eles funcionem bem no mundo real (em produção). Também organizei minhas primeiras referências, resumos e anotações, que foram sistematizados no **Apêndice 1**. Essa etapa inicial foi essencial para que eu começasse a enxergar MLOps não apenas como ciclo fechado, que uma vez implementado não precisa ser aprimorado, mas como um melhoramento contínuo.

Ao longo das **Semanas 3 e 4**, aprofundei meus estudos em MLOps, saindo de uma visão mais geral para olhar com mais atenção para deployment, monitoramento, observabilidade e concept drift, áreas que me despertaram mais atenção. Com isso pude

[1] Dez Semanas, entre setembro de 2025 e dezembro de 2025.

entender que colocar um modelo em produção é apenas o começo e que acompanhá-lo ao longo do tempo é um desafio tão importante quanto o treinamento em si. A partir da leitura de artigos, surveys e relatos de casos, comecei a perceber com mais clareza os desafios de levar modelos de machine learning para produção e mantê-los funcionando em sistemas reais, lidando com integrações, atualizações e possíveis mudanças nos dados ao longo do tempo. Nesse período, também iniciei a organização mais sistemática das referências: montei uma planilha com os principais materiais estudados, registrei as observações que considerei mais importantes e comecei, em conjunto com colegas, a estruturar um glossário com termos recorrentes da área. Todo esse conjunto de registros, leituras e anotações que deu suporte a essa fase da Jornada está reunido no **Apêndice 2**.

No período correspondente às **Semanas 5 e 6**, minha Jornada ganhou um foco mais definido em concept drift. A partir da leitura de materiais específicos sobre o tema, que depois organizei no **Apêndice 3**, aprofundei meu entendimento sobre o que é concept drift, seus diferentes tipos e por que ele é um problema central para modelos em produção. Nesse período, também avancei na construção do glossário e na seleção de referências que seriam importantes para as etapas seguintes, buscando conectar melhor teoria e prática. Em seguida, estudei com mais atenção propostas de métricas e métodos de detecção de drift e comecei a explorar, em experimentos iniciais, como esses métodos se comportam em cenários de mudança nos dados, o que reforçou a ideia de que o monitoramento contínuo é indispensável para manter modelos úteis ao longo do tempo.

Entre as **Semanas 7 e 8**, comecei uma fase bem mais prática focada em experimentar, na nuvem, o que eu vinha estudando em teoria sobre monitoramento e detecção de concept drift. Em um primeiro momento, utilizei soluções em nuvem que já ofereciam uma estrutura de monitoramento pronta, o que me ajudou a entender melhor o funcionamento dessas plataformas e os recursos que elas disponibilizam. No entanto, à medida que eu avançava, fui percebendo limitações na possibilidade de incluir métricas totalmente personalizadas para concept drift, o que me levou a buscar caminhos mais flexíveis. Parte dessas experiências iniciais e das escolhas que precisei fazer ao longo do processo está relatada no **Apêndice 4**. Foi nesse período que ficou claro para mim que eu

gostaria de trabalhar com um *pipeline* em que tivesse mais controle sobre o monitoramento e sobre as métricas utilizadas.

A **Semana 9** foi marcada por uma virada bem prática na construção do *pipeline* de detecção de drift. Nesse período, estudei com mais profundidade o Prefect e o MLflow, entendendo o primeiro como uma ferramenta de orquestração de fluxos de trabalho e o segundo como uma plataforma para acompanhar experimentos, métricas e versões de modelos, dentro de um cenário de MLOps mais completo. Segui um tutorial que mostra, passo a passo, como integrar essas duas ferramentas em um *pipeline* de detecção de drift e, a partir dele, consegui adaptar o fluxo para substituir a métrica original pela métrica PADD, que eu já vinha estudando nas semanas anteriores. Ao mesmo tempo, comecei a pesquisar conjuntos de dados adequados para testar diferentes abordagens de detecção de drift e encontrei o repositório driftDatasets, que reúne datasets reais e simulados, justamente para esse tipo de estudo. Todo esse conjunto de experimentos, adaptações no *pipeline* e seleção de dados que consolidou essa etapa da Minha Jornada está descrito no **Apêndice 5**.

A **Semana 10** foi dedicada a amarrar tudo o que eu vinha construindo ao longo da Jornada, focando na análise dos resultados das métricas de detecção de drift. A partir dos experimentos que descrevo com mais detalhes no **Apêndice 6**, avaliei como diferentes métricas se comportavam na identificação de mudanças nos dados, observando tanto a quantidade de falsos positivos quanto o atraso entre o momento em que o drift ocorria e o instante em que ele era detectado. Para isso, utilizei dados sintéticos do repositório original do PADD e também conjuntos do DriftDatasets, que trazem cenários com drift gradual e abrupto, permitindo uma comparação mais justa entre as abordagens. Ao final dessa etapa, pude perceber que algumas métricas se mostraram mais adequadas para mudanças graduais, enquanto outras responderam melhor a alterações súbitas, e que o comportamento de cada uma precisa ser interpretado dentro do contexto em que o modelo será utilizado, reforçando a importância de escolher com cuidado como o drift será monitorado em produção.

Claro, vamos encaixar os agradecimentos bonitinho nesse final:

Ao olhar para tudo o que construí ao longo destas **Semanas**, percebo que essa Jornada foi decisiva para consolidar meu interesse por MLOps e, especialmente, por monitoramento e concept drift. Passei por momentos de descoberta, dúvida, frustração e conquista, e cada etapa contribuiu para que eu desenvolvesse uma visão mais madura sobre o ciclo de vida de modelos em produção. Essa experiência não só ampliou meu conhecimento técnico, como também reforçou minha vontade de seguir aprendendo e contribuindo com essa área nos próximos projetos, pesquisas e desafios que virão.

Gostaria ainda de registrar meus agradecimentos aos meus pais, minha mãe Marlúcia Tavares de Abreu e meu pai Everaldo Antonio da Silva, que me criaram e formaram a pessoa que sou hoje, sempre fazendo de tudo para que eu tivesse acesso a todas as oportunidades. Agradeço também à minha família, que sempre esteve ao meu lado ao longo de toda a minha vida, em todos os momentos em que precisei. Estendo meus agradecimentos a todos os professores que fizeram parte do meu processo de formação; entrei no curso muito perdida, mas graças a vocês esse caminho foi o mais leve possível. Em especial, agradeço ao professor Anderson da Silva Soares e à professora Telma Woerle de Lima Soares, por todas as oportunidades que mudaram a minha vida; ao professor Fernando Marques Federson, por todos os ensinamentos ao longo de toda a Jornada, não só nessas dez **Semanas**, mas desde o início do curso; e ao professor Sávio Salvarino Teles de Oliveira, pelas oportunidades, por acreditar em mim e por sempre me ajudar quando precisei. Por fim, agradeço também aos meus colegas de turma, sem eles eu não chegaria até aqui. Muito obrigada pela parceria, pelo apoio, pelas risadas e por tudo o que aprendi ao lado de vocês ao longo do curso.

APÊNDICE 1

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 4 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Danielle Tavares da Silva

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante esta primeira Semana, foram realizadas as seguintes atividades:

- **Estudo inicial sobre MLOps**
  Foi estudado o conceito de Machine Learning Operations (MLOps) em sua relação direta com o DevOps. Ambos compartilham princípios como integração e entrega contínua, automação, colaboração e monitoramento, mas o MLOps amplia essas práticas ao ciclo de vida de modelos de machine learning, incluindo essas particularidades:
  - Coleta e organização de dados
  - Treinamento de modelos
  - Validação de modelos
  - Deployment
  - Monitoramento e Observabilidade
  - Re-treinamento contínuo

  Estudos Iniciais em MLOps

- **Pesquisa de materiais**
  Foram utilizadas as plataformas Consensus e Research Rabbit para localizar artigos e materiais de referência relacionados ao tema, resultando na criação de um banco de referências que auxiliará tanto na construção da base inicial de conhecimento quanto no suporte contínuo ao longo do processo.
  Referências_MLOPS

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Atividades para próxima semana:

- Estudar algumas das referências encontradas, de forma a consolidar a compreensão sobre os principais componentes do MLOps: coleta de dados, treinamento de modelos, validação, deployment, monitoramento e re-treinamento contínuo.
- Produzir um documento compilando as pesquisas, que servirá como base para as semanas seguintes.
- Realizar uma reflexão sobre possíveis recortes de aprofundamento para as próximas etapas

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Estudos Iniciais em MLOps

Como parte da primeira semana de atividades do processo de especialização em MLOps, iniciei os estudos a partir de materiais de referência que me permitissem construir uma visão ampla sobre o tema. Realizei a leitura do artigo mais antigo que encontrei sobre MLOps, de um artigo científico que apresenta uma definição consolidada, e, por fim, de um livro de caráter mais prático, utilizado também como referência principal no grupo de estudos de MLOps do qual faço parte.

Leituras feitas:

Sustainable MLOps: Trends and Challenges (Damian A. Tamburri, 2020)

O artigo mais antigo que encontrei sobre o tema foi o de Tamburri (2020), Sustainable MLOps: Trends and Challenges. O autor apresenta uma definição concisa de MLOps e relaciona o conceito à sustentabilidade de software, destacando que operações de IA precisam ser sustentáveis sob as perspectivas técnica, organizacional e social. Embora ferramentas como Apache AirFlow, Kubeflow e Google AutoML facilitem a adoção do MLOps, a complexidade crescente ameaça sua viabilidade a longo prazo. O estudo também aponta desafios educacionais e de mercado, como a escassez de profissionais capacitados e a dificuldade de manter operações contínuas em ambientes de dados intensivos. Por fim, o artigo sugere que a evolução do MLOps deve estar associada a princípios de explicabilidade, justiça, responsabilidade e sustentabilidade nos sistemas de IA.

Artigo Machine Learning Operations (MLOps): Overview, Definition, and Architecture (Kreuzberger, Kühl e Hirschl, 2022)

O artigo de Kreuzberger, Kühl e Hirschl (2022), Machine Learning Operations (MLOps): Overview, Definition, and Architecture, apresenta uma visão abrangente sobre o MLOps, discutindo sua importância, componentes principais e desafios. A partir de revisão de literatura, análise de ferramentas e entrevistas com especialistas, os autores propõem uma definição prática e teórica que entende o MLOps como a união das práticas de DevOps às necessidades do aprendizado de máquina, com foco em automação, reprodutibilidade e escalabilidade. Como contribuição, destacam uma arquitetura conceitual baseada em versionamento de dados e modelos, pipelines automatizados e monitoramento contínuo. O estudo também aponta entraves atuais, como a falta de padronização, a gestão de dados

sensíveis e lacunas de competências, além de sugerir caminhos para pesquisas futuras e melhores práticas na adoção do MLOps.

Livro Practical MLOps: Operationalizing Machine Learning (Noah Gift) – Capítulo 1

O Capítulo 1 do livro Practical MLOps: Operationalizing Machine Learning (Noah Gift, 2020), utilizado como referência no grupo de estudos em MLOps, traz uma introdução prática ao tema e estabelece um paralelo entre DevOps e MLOps. Assim como o DevOps transformou o desenvolvimento de software com automação e integração contínua, o MLOps busca aplicar esses princípios ao ciclo de vida de modelos de machine learning. O autor destaca a importância de automatizar não apenas o código, mas também dados, modelagem, treinamento e monitoramento, apresentando a Pirâmide de Necessidades do MLOps, que organiza a evolução do processo desde o DevOps até práticas completas de operacionalização de modelos. O capítulo também aborda conceitos fundamentais como CI/CD, infraestrutura como código, versionamento de modelos e monitoramento de data drift, além de destacar ferramentas como AWS SageMaker, GitHub Actions e Docker que viabilizam sua aplicação prática.

Link para a planilha: Referências_MLOPS

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 10 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Danielle Tavares da Silva

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Na Semana anterior o estudo foi mais introdutório e superficial. O foco desta segunda Semana foi aprofundar a compreensão dos principais componentes do MLOps.

Atividades Realizadas:

- **Leitura** do artigo *What is MLOps?* (Medium) → baixei a página e fiz anotações. What's MLOps?
- **Curso (Coursera / DeepLearning.AI)** → assisti os primeiros vídeos da playlist no YouTube, fiz algumas anotações pontuais e enriqueci alguns conceitos com ajuda de LLMs.
  Anotações - playlist

- **Leitura** *MLOps: Continuous delivery and automation pipelines in machine learning* (Google Cloud) → como aplicar MLOps com CI/CD e treinamento contínuo (CT) para automatizar todo o ciclo de vida de sistemas de Machine Learning, desde a criação até a operação em produção.

  Resumo

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima Semana:

- Aprofundar os estudos em **Deployment**, **Monitoramento e Observabilidade** e **Re-treinamento Contínuo**
  - Pesquisar referências e materiais sobre esses temas, estudando de forma mais detalhada os conceitos
  - Compilar os aprendizados em documentos organizados que servirão de base para as etapas seguintes.

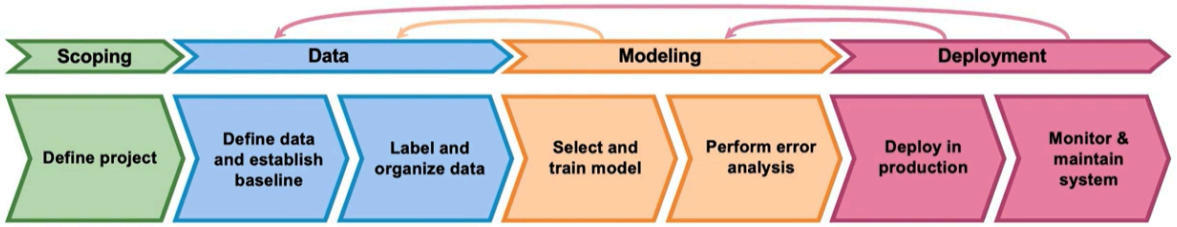
Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Anotações - playlist

Ciclo de vida de ML



1. **Scoping** → definir problema, métrica(s), recursos e timeline
2. **Dados** → definir, rotular/organizar, estabelecer *baseline*
3. **Modelagem** → selecionar/treinar, fazer **análise de erros**
4. **Deployment** → colocar em produção
    a. **Monitorar & Manter** → medir, alertar, aprender e **voltar** etapas quando necessário

Não é linear: você pode voltar de qualquer etapa para outra (principalmente de produção → dados/modelo).

0 - Scoping

Decisões

- Qual dor de negócio será resolvida?
- **Métricas-chave**:
    - *Acurácia/qualidade* (ex.: F1, AUC, WER, RMSE)
    - *Latência* (p95/p99) e *throughput* (req/s), além de SLA/SLO
    - Métrica de **negócio** (conversão, fraude evitada, tempo de atendimento etc.)
- Restrições/risco: privacidade, custo, explainability, fairness.
- Estimar **recursos** (dados, rótulos, GPU/CPU) e **timeline**.

1 - Dados (definir, organizar, baseline)

Decisões e checklists (generalize do caso de áudio para qualquer dado)

- **Definição do dado**: esquema, formatos, sampling rate/normalização
- **Rotulagem consistente**: guia de anotação, exemplos positivos/negativos, casos ambíguos.
- **Pré-/pós-processamento padrão**
- **Partições**: train/val/test por entidad
- Balanceamento, *stratification*, e análise de viés por fatias (grupo, região, dispositivo…).
- **Versionamento** de dados e *data lineage*; contratos de dados.

Baseline

- **Baseline de dados**: estatísticas, distribuição, (%) faltantes/outliers, cobertura por fatias.
- **Baseline de modelo** simples (regra/heurística/majoritária) para ancorar ganhos.

2 - Modelagem (selecionar/treinar + análise de erros)

Prática geral

- Pesquisa acadêmica: tende a **fixar dados** e variar **código/arquitetura/hiperparâmetros**.
- Produto: tende a **otimizar dados/rotulagem** primeiro (data-centric), depois refinar modelo.

Checklist

- Definir pipeline reproduzível (semente, ambiente, versões).
- **Rastreamento de experimentos** (código, dados, hiperparâmetros, métricas).
- **Análise de erros**: por tipo e por fatias (onde o modelo falha e por quê).
- Planos de melhoria priorizando **dados** (novos exemplos, limpeza, guias melhores), depois **modelo** (features/arquitetura/hparams).

3 - Deployment (colocar em produção)

Padrões de serviço

- **Batch** (janela programada)
- **Online/API** (baixa latência)
- **Streaming** (eventos contínuos)

Arquitetura (generalizando o diagrama)

- **Serviço de predição** (API ou job)
- **Pré-/pós-processamento** no cliente/edge ou no servidor (ex.: VAD em áudio; limpeza/normalização; OOD/outlier gate)
- Integração com **feature store**, **model registry**, observabilidade e **auth**.

Checklist

- Requisitos de **latência/throughput** e *autoscaling*
- Testes: unitários, integração, carga, *shadow* e **canary/blue-green**
- Plano de **rollback** (versões no registry)
- Segurança/privacidade (masking/PII, rate-limit)

4 - Monitoramento e Manutenção

O que medir

- **Operacional**: latência p95/p99, erro, uso de recursos
- **Dados**: esquema, nulos, deriva (PSI, KS, validação adversária), OOD
- **Modelo**: qualidade com rótulos (quando chegam) e **métricas proxy** quando rótulos atrasam
- **Negócio**: KPI real que o modelo influencia

Drifts

- **Data drift (covariate shift)**: mudam as entradas X
- **Concept drift**: muda a relação X→Y (o mundo mudou)
- **Label delay**: rótulos chegam muito depois (use *proxies* no curto prazo)

Ações

- Alertas acionáveis + dashboards por fatias
- **Gatilhos** de retreino (queda de métrica, drift, novos dados)
- Rotas de fallback (modelo anterior/heurística)

Desafios de deploy

- Data Drift e Concept Drift
- Software engineering issues
    - Real time or Batch
    - Cloud vs. Edge/Browser
    - Compute resourses (CPU/GPU/memory)
    - Latency, throughput (Queries per secund)
    - Logging
    - Security and privacy

Casos comuns de deployment

1. Novo produto/capacidade

Implantar um modelo de ML para criar **algo totalmente novo**, que não existia antes.
Exemplo: lançar um app que detecta doenças de pele a partir de fotos.

2. Automatizar/auxiliar em tarefa manual

Substituir ou apoiar uma tarefa repetitiva feita por humanos, tornando-a **mais rápida e escalável**.
Exemplo: um modelo que lê documentos e extrai informações automaticamente, reduzindo

3. Substituir sistema de ML anterior

Trocar um modelo antigo por um **mais atualizado, preciso ou eficiente**.
Exemplo: substituir um classificador de fraudes antigo por um novo modelo treinado com mais dados e técnicas modernas.

Resumo

MLOps: pipelines de entrega contínua e automação no aprendizado de máquina

Contexto geral

- O MLOps une práticas de **DevOps** ao **Machine Learning** para automatizar e monitorar **todo o ciclo de vida de ML**: integração, teste, implantação, monitoramento e reentreinamento.
- ML é mais do que apenas o código do modelo → envolve dados, automação, monitoramento, validação, recursos, infraestrutura e governança.
- Principais desafios: dados que mudam ao longo do tempo, modelos que degradam, integração entre ciência de dados e operações.

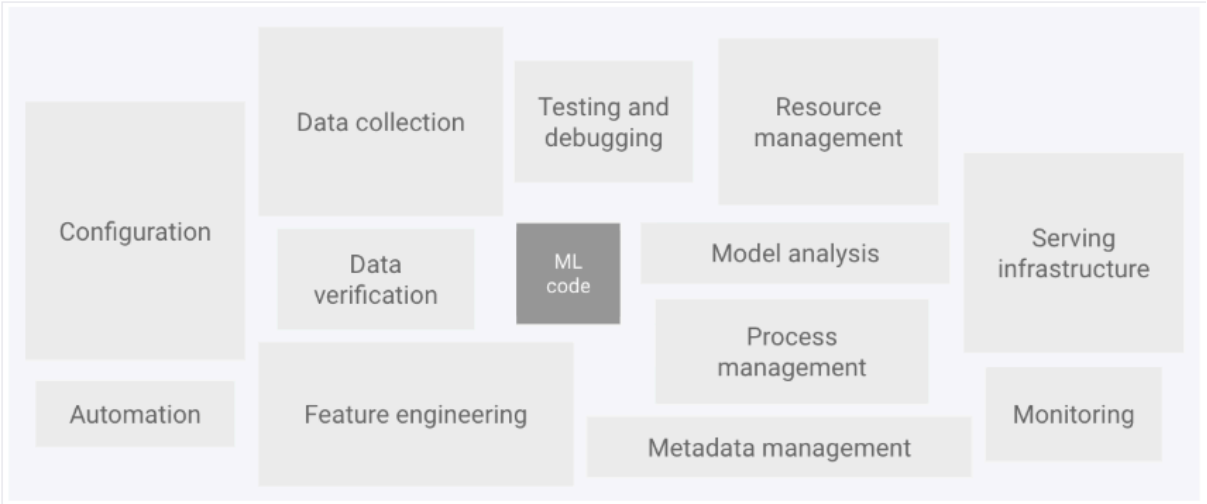


Figura 1: mostra que o código de ML é só uma pequena parte de todo o sistema

Diferença entre DevOps e MLOps

- DevOps → foca em **CI/CD** (código → build → deploy).

- MLOps → além de CI/CD, precisa também de **CT (Continuous Training)** para lidar com novos dados e re-treinamento automático.
- Diferenças-chave:
    - Testes incluem **dados, esquemas e modelos**.
    - Deployment é de **pipelines completos**, não apenas pacotes de software.
    - Monitoramento deve detectar **drift e degradação** do modelo.

Passos típicos de ciência de dados em ML

1. **Extração de dados**
2. **Análise exploratória (EDA)**
3. **Preparação de dados** (limpeza, divisão treino/validação/teste)
4. **Treinamento de modelo**
5. **Avaliação e validação do modelo**
6. **Implantação** (API REST, edge/mobile, batch)
7. **Monitoramento contínuo**

Nível 0 – Processo manual

- Orientado por **scripts e notebooks**.
- Cientistas de dados treinam o modelo → os engenheiros implantam manualmente.
- Iterações pouco frequentes, sem CI/CD, sem monitoramento ativo.
- Adequado para cenários simples, mas modelos tendem a falhar em produção.

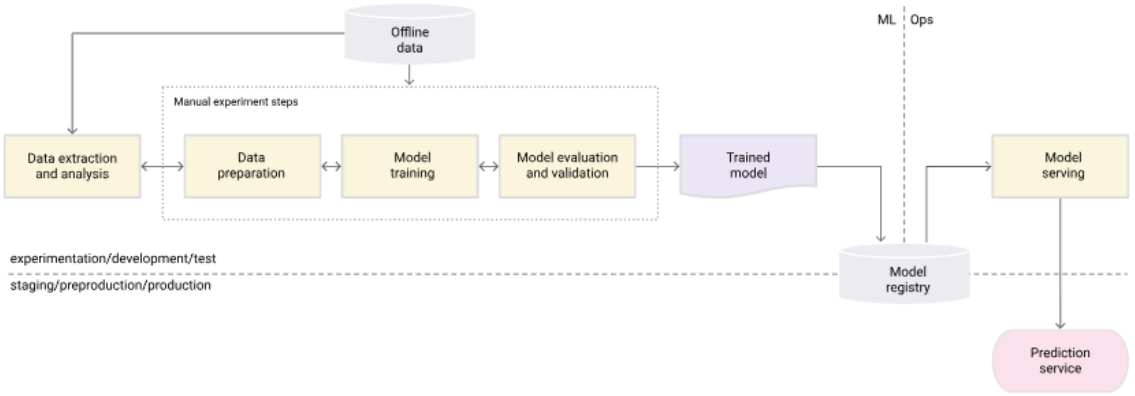


Figura 2: Passos manuais de ML para exibir o modelo como um serviço de previsão.

Nível 1 – Automação de pipeline de ML

- Introduz **pipelines automatizados** e **treinamento contínuo (CT)**.
- Características:
    - Experimentos rápidos e orquestrados.
    - Reuso de código modularizado em componentes.
    - Automação da implantação do **pipeline inteiro** (não só do modelo).
- Componentes extras:
    - **Validação de dados e modelos**.
    - **Feature Store** (armazenamento centralizado de features).
    - **Metadata Store** (para rastrear execuções, versões e métricas).
    - **Triggers** (gatilhos sob demanda, por cronograma, novos dados ou queda de desempenho).

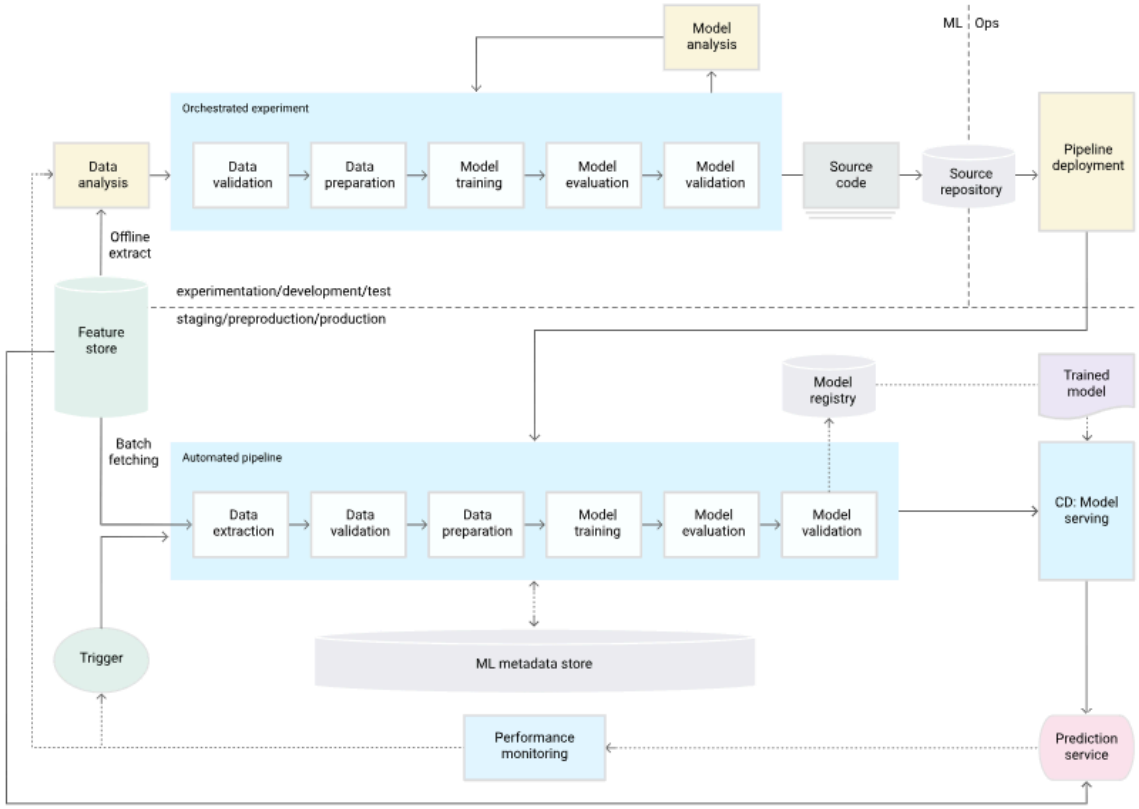


Figura 3: Automação de pipeline de ML para TC.

Nível 2 – Automação de CI/CD para ML

- Além da automação do pipeline, adiciona **CI/CD completo** para ML.
- Permite:
    1. Testes automáticos de código, dados e modelos.
    2. Criação e empacotamento automatizado (imagens, executáveis).
    3. Deploy contínuo com validações (compatibilidade, testes de carga, testes A/B).
- Fluxo típico:

1. Desenvolvimento e experimentação.
2. Integração contínua (build + testes).
3. Entrega contínua (deploy automatizado).
4. Acionadores automáticos para reentreinar modelos.
5. Monitoramento de métricas e drift → dispara novas iterações.

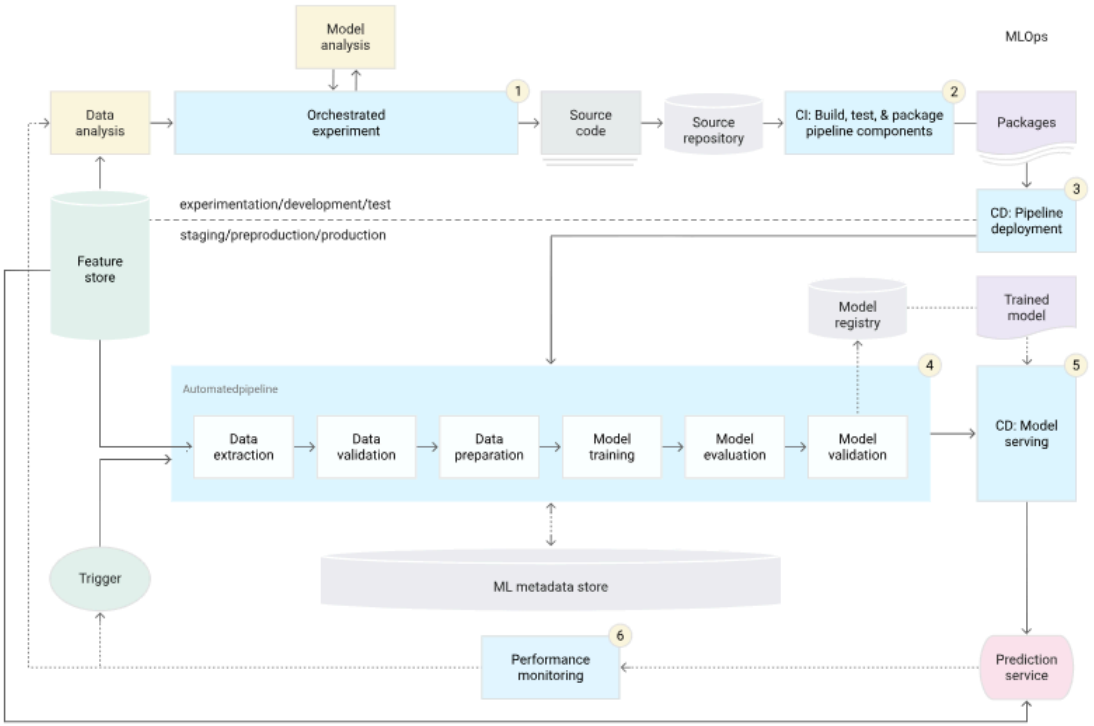


Figura 4: Pipeline de ML automatizado e CI/CD.

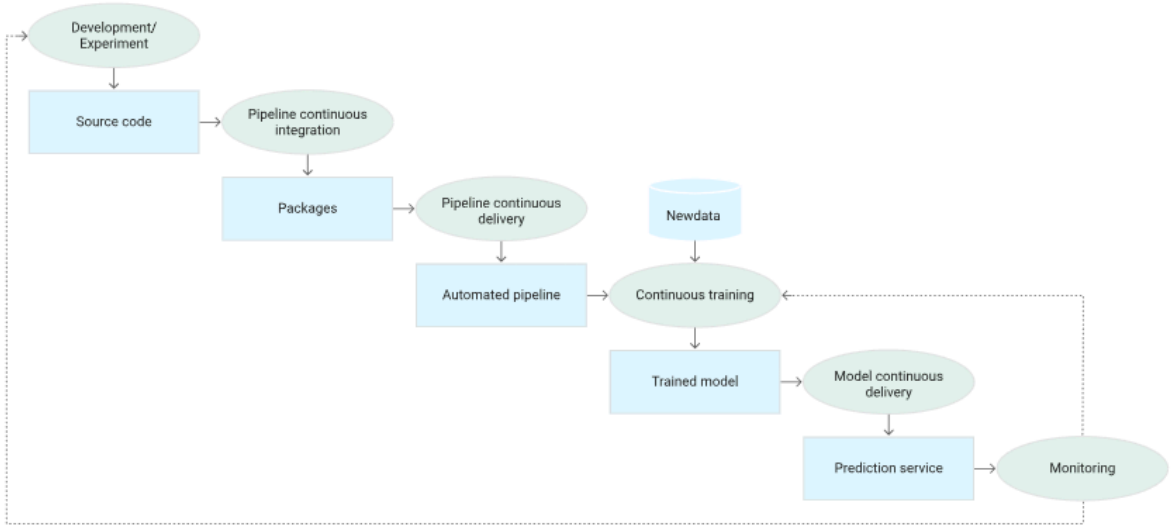


Figura 5: Estágios do pipeline de ML automatizado de CI/CD.

Conclusão

- MLOps não é apenas servir modelos como API, mas sim **implantar e automatizar pipelines completos de ML**.
- A maturidade vai de **nível 0 (manual) → nível 1 (pipeline automatizado) → nível 2 (CI/CD completo)**.
- O avanço de cada nível aumenta a **reprodutibilidade, escalabilidade e resiliência** dos sistemas de ML.

APÊNDICE 2

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 18 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Danielle Tavares da Silva

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Área: MLOps

Essa Semana eu realizei a leitura do survey "Challenges in Deploying Machine Learning: a Survey of Case Studies", e gerei algumas anotações presentes no documento:
Challenges in Deploying Machine Learning: a Survey of Case Studies

A partir dessa leitura, busquei referências, com apoio do ChatGPT no modo agente, relacionadas a **deployment, monitoramento e observabilidade, concept drift e re-treinamento**. Também acrescentei alguns artigos que já havia separado previamente.
Próximas leituras

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima Semana:

- Selecionar e realizar as leituras
- Montagem de um Glossário que será feito em colaboração com a Luísa e a Maria Eduarda

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Challenges in Deploying Machine Learning: a Survey of Case Studies

Este documento foi produzido com base em partes que eu mesma grifei e anotei, com auxílio da ferramenta ChatGPT para resumir

1 INTRODUCTION

O aprendizado de máquina (ML) evoluiu do meio acadêmico para aplicações práticas em empresas, com crescimento anual de cerca de 25%. Essa expansão traz desafios, já que soluções eficazes em pesquisa nem sempre funcionam em sistemas reais, onde surgem gargalos técnicos e questões como viés, custos e falta de expertise.

O estudo busca mapear os principais obstáculos no deployment de ML em produção, abordando o fluxo de implantação, problemas relatados em estudos de caso, aspectos transversais (ética, legislação, confiança e segurança) e possíveis soluções.

Enquanto pesquisas anteriores mostravam "onde" usar ML, hoje a preocupação é "quão difícil é implantar". Relatórios de mercado revelam prazos longos para colocar modelos em produção, altas taxas de fracasso e barreiras recorrentes. Na literatura acadêmica, porém, a cobertura do pipeline completo ainda é limitada, com foco em aspectos específicos ou setores isolados.

2 MACHINE LEARNING DEPLOYMENT WORKFLOW

O estudo adota a definição de workflow de ML proposta por Ashmore et al. Esse fluxo é dividido em quatro estágios principais:

- **Gestão de dados**: preparo dos dados;
- **Aprendizado do modelo**: seleção e treinamento;
- **Verificação do modelo**: checagem de requisitos e de desempenho;
- **Implantação**: integração à infraestrutura de software, manutenção e atualização.

Na prática, essas etapas não seguem uma sequência rígida, podem ocorrer em paralelo, com realimentações entre elas. O modelo é, portanto, uma abstração útil para organizar e classificar os desafios do deployment. Essa representação foi escolhida porque detalha mais passos que outras alternativas, permitindo uma análise mais granular dos problemas.

3 DATA MANAGEMENT

A seção destaca que os dados são parte essencial de qualquer solução de machine learning, tendo peso tão grande quanto os próprios algoritmos para a eficácia do sistema. A criação de datasets de qualidade é apresentada como a primeira etapa de um pipeline de produção em ML, mas também como uma das mais trabalhosas, consumindo tempo e energia além do previsto. O texto estrutura a discussão em quatro etapas principais da gestão de dados: **coleta, pré-processamento, aumento de dados e análise**.

3.1 Data collection

A coleta de dados é descrita como o processo de identificar e compreender quais dados estão disponíveis e como organizá-los em um armazenamento acessível. Esse passo inicial já pode ser bastante desafiador, especialmente em ambientes de produção de grande escala dentro de organizações. Localizar fontes de dados e entender sua estrutura se torna uma tarefa central que muitas vezes impede até o início do desenvolvimento de aplicações de ML. O exemplo do Twitter mostra como a aplicação do princípio de “responsabilidade única” em serviços independentes torna o sistema escalável e flexível, mas dificulta o rastreamento de quais serviços armazenam dados de uma mesma entidade e em qual formato.

3.2 Data preprocessing

O pré-processamento é caracterizado como a etapa de limpeza e padronização dos dados, envolvendo desde a definição de esquemas e preenchimento de valores ausentes até a simplificação e conversão para formatos mais adequados. O texto destaca um desafio menos conhecido, mas igualmente relevante: a dispersão de dados. Isso ocorre quando múltiplas fontes contêm informações relevantes, mas seguem esquemas, convenções e formatos distintos, exigindo um processo complexo de integração.

Um exemplo citado é o do sistema Firebird, criado pelo Departamento de Bombeiros de Atlanta para priorizar inspeções. O projeto precisou unir dados de 12 bases diferentes, que incluíam históricos de incêndios, licenças comerciais e registros de domicílios, em um único dataset. A junção foi especialmente difícil porque os edifícios eram identificados por informações geoespaciais, como endereços, que apareciam em formatos variados e até com pequenas diferenças de grafia.

3.3 Data augmentation

A etapa de **data augmentation**, em prática, é a falta de dados rotulados, causada pelo grande volume de informações, pela dificuldade de acesso a especialistas e pela ausência de dados variados. Em áreas como tráfego de rede, rotular pacotes em escala é inviável, seja coletando tráfego real ou simulando, já que ambos os métodos geram problemas de qualidade. Em domínios críticos, como saúde, depender de especialistas torna o processo caro e limitado, e alternativas com rótulos fracos podem comprometer a segurança. Além disso, o design da interface usada para coletar anotações é apontado como um fator essencial para garantir qualidade e confiabilidade nos rótulos.

3.4 Data analysis

A análise de dados é necessária para identificar vieses e mudanças inesperadas na distribuição. Um dos maiores desafios é a visualização para data profiling, que envolve tarefas de diagnóstico da qualidade dos dados, como detectar valores ausentes, inconsistências de tipos e validar suposições.

4 MODEL LEARNING

O aprendizado de modelos é a fase do workflow de implantação que mais recebe atenção na comunidade acadêmica. Apesar desse foco, ainda existem muitos desafios práticos ligados à seleção de modelos, ao treinamento e à escolha de hiperparâmetros.

4.1 Model selection

Na prática, a escolha de modelos costuma ser definida pelo nível de complexidade. Embora áreas como deep learning e reinforcement learning tenham grande destaque em pesquisas, muitas vezes opta-se por modelos mais simples, redes neurais simples, PCA, árvores de decisão e random forests. Esses modelos servem para validar conceitos rapidamente, montar pipelines funcionais e evitar atrasos provocados por arquiteturas muito elaboradas. Um exemplo é o caso da AirBnB, em que uma rede neural complexa foi simplificada para uma arquitetura com apenas uma camada oculta, o que viabilizou a implantação e forneceu bom desempenho.

Outra razão para a escolha de modelos simples está nos recursos computacionais. Ambientes restritos, como o da sonda espacial Europa Clipper, exigem baixo consumo de energia e memória, favorecendo técnicas como PCA ou limiares simples em vez de deep

learning. Situação parecida ocorre em redes celulares, onde restrições de energia e transmissão ainda inviabilizam modelos avançados.

Além disso, a interpretabilidade muitas vezes pesa mais que a performance. Árvores de decisão são amplamente utilizadas por sua lógica transparente, com aplicações frequentes em manufatura e em bancos, onde prever churn exige explicar quais fatores levam o cliente a desistir.

4.2 Training

O treinamento é o processo de alimentar o modelo com dados para aprender padrões. Um dos maiores problemas são os custos computacionais, especialmente em NLP. O custo cresce com o tamanho dos datasets, o número de parâmetros e a quantidade de operações, sendo esse último fator cada vez mais crítico devido ao aumento de modelos com bilhões de parâmetros.

Outra preocupação é o impacto ambiental. Pesquisas indicam que um ciclo completo de treino com neural architecture search pode emitir tanto CO_2 quanto quatro carros durante toda a sua vida útil. Isso levanta discussões sobre a necessidade de algoritmos e hardwares mais eficientes.

A privacidade dos dados também surge como desafio. Estudos mostraram ataques de membership inference capazes de identificar se um dado específico foi usado no treinamento, com precisão de até 94%. Isso exige técnicas que preservem a privacidade, como differential privacy (que insere ruído nos dados), homomorphic encryption (treinamento sobre dados criptografados, mas limitado) ou federated learning (treinamento distribuído nos dispositivos dos usuários). Todas essas abordagens, no entanto, envolvem um delicado equilíbrio entre privacidade e acurácia.

4.3 Hyper-parameter selection

Modelos de ML dependem não só de parâmetros aprendidos, mas também de hiperparâmetros, como a profundidade de uma árvore ou o número de camadas em uma rede neural. A otimização desses valores (HPO) normalmente exige múltiplos ciclos de treino, tornando-se cara e computacionalmente pesada. O problema cresce de forma exponencial à medida que mais hiperparâmetros entram no espaço de busca.

Técnicas como Hyperband e Bayesian Optimization foram criadas para reduzir o número de ciclos necessários, mas ainda não conseguem lidar bem com buscas de alta

dimensionalidade em grandes datasets. Outro obstáculo é a definição do espaço de busca: em muitos casos práticos não há conhecimento suficiente para estabelecer limites adequados, o que restringe a aplicação de técnicas avançadas.

A escolha de hiperparâmetros também precisa considerar o contexto de execução. Em dispositivos embarcados e móveis, por exemplo, restrições de energia e memória exigem abordagens de HPO adaptadas ao hardware, de modo a equilibrar precisão e eficiência.

5 MODEL VERIFICATION

A verificação é essencial no ciclo de desenvolvimento de software porque assegura a qualidade do produto e reduz custos de manutenção. Em machine learning, o modelo precisa generalizar para dados não vistos, lidar com casos extremos de forma razoável, ser robusto e cumprir requisitos funcionais. Essa etapa envolve três dimensões principais: **definição de requisitos**, **verificação formal** e **verificação baseada em testes**.

5.1 Requirement encoding

Definir requisitos claros para modelos de ML é fundamental antes das atividades de teste. Muitas vezes, ganhos em métricas de acurácia não se traduzem em valor de negócio, como mostrou a experiência da Booking.com ao implantar 150 modelos em produção: métricas de proxy (como cliques) não se convertiam no resultado desejado (como conversões). Por isso, é necessário estabelecer métricas adicionais, específicas de domínio, que podem ser inspiradas em KPIs (Key Performance Points) e outros indicadores de negócios, como conversões, número de cancelamentos ou tickets de atendimento.

Outro ponto é que métricas de desempenho precisam refletir as prioridades da audiência. Isso significa validar modelos em relação a aspectos como viés e justiça, ou ainda recursos específicos do domínio, como consumo de energia em espaçonaves. Assim, a verificação vai além da precisão, conectando o desempenho técnico a critérios práticos e sociais relevantes.

5.2 Formal verification

A verificação formal busca assegurar que o modelo atenda rigorosamente aos requisitos definidos. Em teoria, isso poderia incluir provas matemáticas de correção ou limites numéricos de erro, mas na prática raramente é aplicado. O que predomina são padrões de

qualidade estabelecidos por marcos regulatórios, que definem critérios e processos de conformidade.

O setor bancário é um exemplo marcante, especialmente após a crise financeira global, quando aumentou o nível de fiscalização sobre modelos usados em decisões críticas. Reguladores como a Prudential Regulation Authority (Reino Unido) e o Banco Central Europeu publicaram diretrizes que exigem frameworks de risco para modelos, obrigando os desenvolvedores a criar extensas baterias de testes. Nesse contexto, a verificação formal não é apenas técnica, mas também regulatória: trata-se de demonstrar que os modelos cumprem requisitos oficiais e podem ser auditados de acordo com normas específicas.

5.3 Test-based verification

A verificação baseada em testes garante que o modelo generalize bem para dados não vistos. Embora seja comum derivar um conjunto de validação a partir do dataset de treino, isso não é suficiente para garantir qualidade em produção. O ideal é realizar testes em ambientes reais, onde métricas de negócio possam ser acompanhadas, mas isso nem sempre é viável por razões de segurança, custo ou escala.

Por isso, simulações se tornaram recurso amplamente utilizado, como no desenvolvimento de veículos autônomos. Simulações permitem criar cenários raros, são mais baratas e rápidas, mas têm limitações: dependem de suposições feitas pelos desenvolvedores e podem divergir do mundo real. Pequenas diferenças entre simulação e realidade podem comprometer drasticamente o desempenho do sistema.

Além dos testes do modelo, é crucial validar continuamente os próprios dados usados no pipeline. Problemas podem surgir de bugs, loops de feedback ou mudanças em dependências externas. Como esses erros podem se propagar e se manifestar em diferentes pontos do pipeline, é fundamental implementar rotinas automáticas de validação de dados para evitar falhas difíceis de diagnosticar.

6 MODEL DEPLOYMENT

Sistemas de machine learning em produção são softwares complexos que precisam ser mantidos ao longo do tempo. Isso traz desafios comuns ao desenvolvimento tradicional e outros específicos de ML. Assim como o DevOps surgiu para sustentar sistemas em produção, há a necessidade de adaptar seus princípios ao contexto de ML. Entre os desafios particulares estão: ausência de telemetria padronizada, dificuldade em coletar rótulos para aprendizado

supervisionado em produção e falta de boas práticas consolidadas para lidar com modelos. Dentro desse cenário, a implantação envolve três etapas principais: **integração, monitoramento e atualização**.

6.1 Integration

A integração inclui tanto a construção da infraestrutura para rodar o modelo quanto a implementação do modelo em formato utilizável. Embora a parte de infraestrutura esteja mais ligada à engenharia de sistemas, o texto enfatiza aspectos de software e ML que se sobrepõem.

Um exemplo é o reuso de dados e modelos, prática comum em engenharia de software e cada vez mais adotada em ML. O caso do Pinterest mostra como três modelos distintos de embeddings de imagens foram unificados em um conjunto universal, o que reduziu retrabalho, simplificou pipelines e ainda melhorou a performance.

Outro ponto é a colaboração entre pesquisadores e engenheiros. Embora em teoria pesquisadores cuidem do modelo e engenheiros da infraestrutura, na prática suas áreas se sobrepõem em insumos, métricas e código. Experiências mostram que envolver pesquisadores em todo o ciclo, incluindo versionamento, revisão de código e manutenção da base, traz ganhos em velocidade e qualidade do produto, apesar das dificuldades iniciais.

6.2 Monitoring

O monitoramento é crítico para manter modelos em produção, mas ainda carece de consenso sobre **quais métricas de dados e modelos observar** e como disparar alertas. Monitorar derivações de dados de entrada, vieses de previsão e performance geral continua sendo um desafio aberto.

Um problema particular é o dos feedback loops: ao serem continuamente atualizados, modelos podem influenciar o próprio input, alterando seu comportamento de forma intencional ou não.

Outro aspecto central é a detecção de outliers. Modelos podem falhar ao generalizar fora da distribuição de treino ou gerar previsões excessivamente confiantes. Porém, treinar detectores é difícil pela escassez de rótulos de outliers, tornando o processo semi ou não supervisionado.

6.3 Updating

Após a implantação inicial, atualizar o modelo é essencial para refletir mudanças nos dados e no ambiente. Técnicas incluem re-treinamento agendado e aprendizado contínuo, mas a prática esbarra em limitações como o concept drift, mudanças na distribuição conjunta de entradas e saídas ao longo do tempo. Esse fenômeno pode ser abrupto (eventos externos) ou gradual (alterações acumuladas). Detectar concept drift é hoje uma das maiores preocupações das equipes de ML em produção, já que ele afeta diretamente a decisão de quando e como atualizar.

Além do desafio de detecção, há o como entregar o modelo atualizado. Em software tradicional, isso é resolvido com Continuous Delivery (CD). No caso de ML, o problema é mais complexo porque envolve três dimensões em constante mudança: código, modelo e dados. Trabalhos recentes tentam adaptar CD especificamente para ML, estruturando pipelines completos de atualização.

8.1 Tools and services

O mercado de ferramentas e serviços para ML cresce rapidamente, oferecendo soluções que atacam problemas pontuais do ciclo de deployment. Plataformas como **AWS SageMaker, Microsoft ML, Uber Michelangelo, TensorFlow TFX e MLflow** já entregam pipelines de ponta a ponta, cuidando de armazenamento de dados, re-treinamento e monitoramento, reduzindo a carga operacional.

Na área de **qualidade e verificação**, destacada na Seção 5, novas ferramentas ajudam a criar suítes de teste. Exemplos são o **Jenga** (robustez contra erros de dados), a metodologia **CheckList** (avaliação formal da qualidade em NLP) e o **Data Linter** (checagem de problemas em datasets).

No tema de **rotulagem de dados**, discutido na Seção 3.3, a área de **weak supervision** trouxe bibliotecas como **Snorkel, Snuba e cleanlab**, que já apresentam bons resultados industriais.

Para **seleção de modelos e ajuste de hiperparâmetros** (Seções 4.1 e 4.3), ferramentas de **AutoML** como Auto-Keras, Auto-sklearn e TPOT automatizam parte do processo. Porém, relatos práticos alertam que ainda não estão maduras para contextos de alto risco.

Quanto ao **concept drift** (Seção 6.3), bibliotecas como **Alibi Detect** e serviços como **Azure ML** e **AWS SageMaker** já oferecem detecção, enquanto abordagens como **domain adaptation, meta-learning e transfer learning** ajudam a mitigar efeitos do shift.

O uso de ferramentas, contudo, traz o risco de dependência excessiva. Como novas soluções surgem constantemente, escolher, aprender e manter ferramentas adicionais pode se tornar um fardo de manutenção.

Link para a planilha: Próximas leituras

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 24 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Danielle Tavares da Silva

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta Semana, foi realizada a leitura do artigo **"Machine Learning Operations: A Survey on MLOps Tool Support"**, que faz um levantamento das principais ferramentas usadas em MLOps. O estudo mostra como essas ferramentas se organizam para apoiar todo o ciclo de vida de machine learning, desde a preparação e versionamento de dados até o deploy e monitoramento dos modelos.
Machine Learning Operations: A Survey on MLOps Tool Support

Também foi lido o artigo **"Monitoring machine learning models: a categorization of challenges and methods"**, apresenta os principais desafios de verificação e validação de modelos em produção, como mudanças nos dados (dataset drift), perda de desempenho e problemas de integração com outros sistemas. Além disso, o artigo propõe uma taxonomia dos métodos de monitoramento, que classifica diferentes abordagens para acompanhar dados e modelos em produção, considerando métricas de desempenho, robustez, confiança, custos, interpretabilidade e aspectos éticos.
Monitoring machine learning models: a categorization of challenges and methods

Por fim, houve o acréscimo de novos termos no **glossário de MLOps**, desenvolvido em conjunto com Maria Eduarda e Luísa.
Glossário de termos na área de MLOPS

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima Semana, utilizando a planilha de referências levantada na Semana anterior, o foco será o aprofundamento nos estudos sobre **monitoramento de modelos e concept drift.**

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Machine Learning Operations: A Survey on MLOps Tool Support

O artigo aborda a importância do **MLOps** como extensão do DevOps para integrar o ciclo de vida de Machine Learning (dados, modelos e código) dentro de pipelines de CI/CD.
O objetivo é identificar e comparar ferramentas de MLOps, destacando como elas apoiam desde **data engineering, versionamento, automação de pipelines, deployment até monitoramento de modelos**.

Tabela de Ferramentas

Ferramenta	Para que serve
Kubeflow	Criado pelo Google, roda sobre Kubernetes. Facilita **deploy escalável de modelos**, automação de pipelines e monitoramento. Tem notebooks, pipelines e KFServing. Limitação: não possui ferramenta nativa de CI/CD, mas os pipelines ajudam.
MLFlow	Open-source, não depende de nuvem. Cobre ciclo de ML com **tracking de experimentos, packaging de projetos, modelos e model registry**. Suporta várias libs e linguagens. Permite deploy com AWS, Azure, Spark. Limitação: não possui notebooks embutidos, gestão de usuários limitada.
Iterative Enterprise (DVC + CML)	Inclui **DVC (Data Version Control)** para versionar datasets e modelos grandes e **CML (Continuous Machine Learning)** para CI/CD em ML via GitHub/GitLab. Gera relatórios automáticos em pull requests. DVC Studio facilita colaboração.
DataRobot	Plataforma comercial que centraliza **deploy, monitoramento e gestão de modelos em produção**, independente de como foram criados. Suporta várias linguagens e ambientes. Ponto negativo: requer licenciamento pago por instância.
Allegro.ai (ClearML)	Open-source da Allegro.ai. Inclui **ClearML**, que cobre experimentos, orquestração, DataOps, hyper-datasets, deploy e ambientes remotos. Tem pacotes Python, servidor para gestão de experimentos e módulos para notebooks remotos.

MLReef	Open-source baseado em Git. Oferece **ambiente colaborativo, reprodutível, com pipelines rápidos**. Focado em colaboração, compartilhamento e integração CI/CD. Disponibiliza CPU/GPU grátis em alguns casos.
Streamlit	Biblioteca Python para criar **dashboards e web apps** rápidos. Muito usado para visualização de resultados de ML, não cobre todo o ciclo de MLOps, mas facilita apresentação e interação.
Cloud Providers (Google, Azure, AWS)	- **Google Cloud**: Dataflow, AI Notebooks, Cloud Build, TFX e Kubeflow pipelines. - **Azure**: Azure ML (treino e deploy), Pipelines (CI/CD), Monitor, Kubernetes. - **AWS SageMaker**: cobre todo ciclo de ML (treino, deploy, monitoramento). Muito completo, mas pago.

Legenda:

- **DV** = Data Versioning
- **HT** = Hyperparameter Tuning
- **MEV** = Model/Experiment Versioning
- **PV** = Pipeline Versioning
- **CICD** = Integração e Deploy contínuos
- **MD** = Model Deployment
- **PM** = Performance Monitoring

Tabela Comparativa de Features

Ferramenta / Plataforma	DV	HT	MEV	PV	CICD	MD	PM
AWS SageMaker	✅	✅	✅	✅	✅	✅	✅
MLFlow	✅		✅	✅	✅	✅	
Kubeflow	✅		✅	✅	✅	✅	
DataRobot			✅	✅		✅	✅
Iterative (DVC + CML)	✅		✅	✅	✅		
ClearML (Allegro.ai)	✅		✅	✅	✅	✅	✅

MLReef	✅		✅	✅	✅	✅	✅
Streamlit				✅			

Monitoring machine learning models: a categorization of challenges and methods

Monitoring machine learning models: a categorization of challenges and methods (Schröder & Schulz, 2022).

- O uso de ML cresce rapidamente, mas muitos projetos não passam da fase de teste.
- Diferenças em relação a software tradicional: modelos são dinâmicos, black boxes, frágeis a mudanças no ambiente.
- Objetivo do artigo: investigar **desafios de verificação e validação (V&V)** em modelos de ML em operação e propor **taxonomia de métodos de monitoramento**.

2. Definições

- **MLM (Machine Learning Model):** modelo estatístico treinado com aprendizado supervisionado.
- **MLS (Machine Learning System):** conjunto de componentes, incluindo o MLM, pipelines de dados, monitoramento.
- **Verificação:** checar se requisitos técnicos foram atendidos.
- **Validação:** avaliar se o sistema satisfaz as necessidades reais dos usuários.
- **Monitoramento:** coleta, análise e interpretação de métricas em produção para detectar problemas e corrigi-los.
- Diferença entre **teste** (antes da produção) e **monitoramento** (durante a operação).

3. Desafios na operação de modelos de ML

1. **Alta dimensionalidade:** complexidade de dados e do espaço de estados do modelo, dificultando análises.
2. **Dataset shift:** mudanças nas distribuições entre treino e produçã
3. **Robustez:** vulnerabilidade a ruídos, ataques adversariais, inputs inválidos ou mudanças de configuração.

4. **Interdependência de sistemas:** dependências entre pipelines de dados, bibliotecas externas e entre modelos.

5. **Comunicação de resultados:** dificuldade de interpretar e explicar previsões; baixa transparência; problema da incerteza preditiva.

6. **Projeto e validação de testes:** dificuldade de definir oráculos de verdade, efeitos em cadeia em sistemas complexos e pouca validade de testes antes do deployment.

4. Monitoramento de modelos de ML

Propõe uma **taxonomia de métodos**, organizada em dois eixos: **dados vs. modelo** e **técnico vs. não técnico**.

4.1. Monitoramento de dados

- **Validade por observação:** checar completude, consistência e correção de registros individuais.
- **Validade entre lotes:** comparar distribuições de batches para detectar dataset shift.
- **Novidade:** identificar dados fora da distribuição conhecida.
- **Aspectos econômicos:** custo de coleta vs. utilidade preditiva das features.
- **Aspectos éticos:** evitar uso de atributos sensíveis (idade, gênero, etnia).

4.2. Monitoramento de modelos

Métricas divididas em seis aspectos:

1. **Performance:** acurácia, precisão, recall, MAE, RMSE; detectar quedas de desempenho.
2. **Robustez:** avaliar resistência a ruídos, erros de dados e ataques adversariais.
3. **Confiança:** medir a incerteza das previsões e a calibragem das probabilidades.

4. **Econômico:** consumo de recursos, ações do modelo, relação entre métricas técnicas e métricas de negócio.

5. **Interpretabilidade:** avaliar complexidade do modelo, explicabilidade

6. **Ética:** medir justiça (fairness) com base em critérios como group fairness, counterfactual fairness, individual fairness.

Glossário de termos na área de MLOps

Tabela resumo dos termos:

Termo	Definição
CI (Continuous Integration)	Processo de integração contínua de código e componentes de ML. Garante que alterações em scripts, pipelines ou configurações sejam testadas e validadas de forma automática, prevenindo falhas na colaboração entre equipes.
CD (Continuous Delivery/Deployment)	Prática de entregar e implantar modelos e pipelines de forma contínua e confiável em ambientes de produção. Permite atualizações rápidas e seguras, reduzindo o tempo
CT (Continuous Training)	Extensão das práticas de CI/CD para o aprendizado de máquina. Automatiza o reprocessamento e re-treinamento de modelos sempre que novos dados estão disponíveis ou quando ocorre drift, garantindo que o modelo permaneça atualizado e relevante.
CM (Continuous Monitoring)	Monitoramento constante de dados e modelos em produção, avaliando métricas técnicas e de negócio, além de detectar problemas como *drift* e vieses.
Orquestração	Gerenciamento e coordenação das várias tarefas e fluxos de trabalho envolvidos no ciclo de vida do aprendizado de máquina, desde a preparação de dados e treinamento de modelos até a implantação e monitoramento.
ML Observability	Conjunto de práticas, ferramentas e métricas voltadas para entender, monitorar e diagnosticar o comportamento de modelos de ML em produção.

Orquestração de Containers	É o processo de automatizar o gerenciamento, a implantação, a escala e a comunicação de aplicações empacotadas em containers.
Experiment Tracking	Processo de monitorar e registrar experimentos de ML, armazenando metadados como hiperparâmetros, métricas, modelos gerados e ambiente computacional. Garante reprodutibilidade, facilita comparações entre execuções e auxilia na escolha do melhor modelo.
Model Registry	Repositório central para versionar, organizar e gerenciar modelos de ML, permitindo controle de ciclo de vida, rastreabilidade e implantação estruturada em produção.
Data Lineage	Rastreamento da origem, transformação e uso dos dados ao longo do pipeline de ML. Permite entender de onde vêm os dados, como foram processados e onde são aplicados, garantindo transparência, auditoria e confiabilidade nos experimentos e modelos.
Feature Store	Repositório centralizado para armazenar, versionar e compartilhar *features* usadas em modelos de ML. Facilita a reutilização, mantém consistência entre treino e produção e acelera o desenvolvimento de novos modelos.
Explainability (XAI – Explainable Artificial Intelligence)	Conjunto de técnicas que tornam modelos de ML interpretáveis e transparentes, permitindo que especialistas entendam e confiem nas decisões dos algoritmos. Towards MLOps: A Framework and Maturity Model
GitOps	Extensão de DevOps que usa repositórios Git como "fonte de verdade" para infraestrutura e deployment, permitindo que alterações sejam rastreadas e aplicadas automaticamente.

Shadow Deployment	Estratégia de deployment em que um novo modelo roda em paralelo ao modelo atual, mas sem impactar usuários finais, servindo apenas para testes.
Canary Deployment	Implantação gradual de um novo modelo em uma fração controlada dos usuários, reduzindo riscos de falhas em larga escala. *Toward an Open Source MLOps Architecture.*
Technical Debt (Dívida Técnica)	Compromissos assumidos ao priorizar velocidade sobre qualidade na implementação, que podem aumentar custos de manutenção futura. Hidden Technical Debt in Machine Learning Systems
Maturity Model (Modelo de Maturidade)	Estrutura que descreve níveis de evolução na adoção de MLOps, desde fluxos manuais (nível inicial) até pipelines totalmente automatizados com monitoramento e re-treinamento. Towards MLOps: A Framework and Maturity Model, John, Olsson & Bosch, 2021
Compliance	Adequação de sistemas e pipelines a normas legais e regulatórias (ex.: GDPR, HIPAA), garantindo governança, privacidade e auditabilidade.
A/B Testing	Método experimental que compara duas ou mais versões de um modelo/sistema para avaliar impacto em métricas técnicas ou de negócio.
Blue-Green Deployment	Mantêm-se dois ambientes de produção idênticos: "Blue" (com a versão atual) e "Green" (com a nova versão). O tráfego é direcionado para o ambiente Blue. Quando a nova versão no Green é validada, o tráfego é todo redirecionado para ele. Se algo der errado, é fácil e rápido reverter para o ambiente Blue.
Rolling Deployment	A nova versão do modelo é gradualmente introduzida, substituindo as instâncias da versão antiga uma a uma, até que todas

	estejam atualizadas. Isso permite uma transição suave e sem tempo de inatividade (downtime).
Big Bang (ou Recreate)	A versão antiga é desligada e a nova é ligada de uma vez. É a abordagem mais simples, porém mais arriscada, pois qualquer problema na nova versão afeta todos os usuários.
Federated learning	É uma abordagem de aprendizado de máquina distribuído em que os dados permanecem nos dispositivos ou organizações que os geraram, sem necessidade de centralização; em vez disso, cada nó treina localmente um modelo e envia apenas atualizações de parâmetros (como gradientes ou pesos) para um servidor central, que agrega os resultados e atualiza o modelo global. Esse processo garante maior privacidade, já que os dados brutos não são compartilhados, melhora a eficiência ao reduzir a transferência de dados e permite personalização em contextos locais. No entanto, apresenta desafios como a heterogeneidade dos dados, altos custos de comunicação e riscos de segurança. É amplamente aplicado em áreas como teclados inteligentes de smartphones, saúde e sistemas financeiros.
Dataset Shift	Mudança na distribuição dos dados de entrada usados pelo modelo em produção em relação aos dados de treinamento. Isso faz com que o modelo enfrente padrões diferentes dos que aprendeu, podendo reduzir sua performance.
Concept Drift	Alteração na relação entre as variáveis de entrada (features) e o alvo (label) ao longo do tempo. Diferente do dataset drift, que é mudança só na distribuição dos dados, aqui muda o significado do que o modelo deve

	aprender. Mesmo que os dados de entrada não mudem muito, a regra que liga entrada e saída pode mudar, exigindo re-treino ou adaptação do modelo.
Ataques Adversariais	Técnicas usadas para enganar modelos de aprendizado de máquina, inserindo pequenas perturbações nos dados de entrada (muitas vezes imperceptíveis para humanos) que levam o modelo a tomar decisões erradas.
Interpretabilidade	Capacidade de compreender e explicar como um modelo de machine learning chega às suas previsões ou decisões. Permite identificar quais variáveis ou fatores influenciaram o resultado, sendo essencial para auditoria, transparência, confiança e uso ético de modelos.
Data Lake	É um repositório centralizado que armazena **grandes volumes de dados em seu formato bruto**, sem necessidade de estrutura pré-definida.
Data Warehouse	É um sistema usado para armazenar e analisar **dados já tratados, limpos e estruturados**, otimizados para consultas e relatórios.
Latência	Tempo entre pedido e resposta, medidos por percentis
AutoML	Automação de etapas críticas de ML (pré-processamento, seleção de features, tuning de hiperparâmetros, seleção de modelos, ensembling).

KaizenML	Foco na melhoria contínua de todo o ciclo de vida do ML.
Drift Detection	Processo de identificar se ocorreu uma mudança significativa na distribuição dos dados ao longo do tempo. Tem como objetivo detectar quando um modelo pode perder desempenho devido a alterações no ambiente ou nos dados.
Drift Localization	Processo de identificar em quais regiões do espaço de dados a mudança ocorreu. Busca determinar onde exatamente o drift acontece, permitindo ações mais direcionadas, como re-treinar o modelo apenas para segmentos afetados.
Drift Explanations	Processo de tornar compreensível o fenômeno do drift, explicando como e por que a mudança ocorreu. Envolve indicar quais variáveis foram mais impactadas, quais padrões se alteraram e como isso afeta o comportamento do modelo.
Exponential backoff	Ao falhar uma chamada, você espera e tenta de novo com intervalos que dobram (1s, 2s, 4s…), com limite e jitter.
Jitter	Pequena aleatoriedade
Circuit breaker	Um “disjuntor” que abre quando a taxa de erro/latência explode; enquanto aberto, bloqueia novas chamadas e retorna rápido (fallback).
Autoscaling	Escala automaticamente a quantidade de instâncias conforme carga para manter

	SLOs.
Throughput	Taxa sustentada de processamento, é importante para definir a capacidade de aguentar pico sem estourar a latência.
Políticas de rollback	Regras para voltar rápido à versão anterior quando o novo deploy degrada métricas.
SLO (Service Level Objective)	lvo/limite da qualidade (ex.: p95 < 200 ms em 99% do mês).
SLI (Service Level Indicator)	Métrica medida (p95, AUC, PSI, 5xx) usada para verificar SLO.
SLA (Service Level Agreement)	Compromisso contratual externo derivado de SLOs.
Burn rate (de erro)	Quão rápido você "queima" o orçamento do SLO; usado para alertas rápido/lento.
Multi-armed bandits	Roteamento adaptativo de tráfego conforme performance online.
Batching vs. Streaming	Processamento em lotes vs. em fluxo/tempo real para deploy/monitoria.

APÊNDICE 3

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 1 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Danielle Tavares da Silva

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta Semana, foi realizada a leitura do artigo *"One or Two Things We Know about Concept Drift: A Survey on Monitoring Evolving Environments"*, que apresenta uma revisão abrangente sobre o fenômeno de concept drift, abordando definições, tipos, metodologias de detecção, localização e explicação, bem como estratégias de segmentação. O estudo organiza os métodos em um esquema geral de quatro etapas: aquisição de dados, construção de descritores, cálculo de dissimilaridade e normalização.
One or Two Things We know about Concept Drift A Survey on Monitoring Evolving Environments

Também foram adicionados novos termos ao glossário de MLOps, desenvolvido em conjunto com Maria Eduarda e Luísa.
Glossário de termos na área de MLOPS

Além disso, houve a atualização da planilha de próximas leituras, incluindo artigos voltados a concept drift e ferramentas que oferecem suporte a esse desafio.
Próximas leituras

Por fim, iniciei a leitura do artigo *"From Concept Drift to Model Degradation: An Overview on Performance-Aware Drift Detectors"*.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Concluir a leitura do artigo *"From Concept Drift to Model Degradation: An Overview on Performance-Aware Drift Detectors"*;
- Ler outros artigos relacionados ao tema de concept drift;
- Explorar plataformas e ferramentas que oferecem suporte ao monitoramento e detecção de drift no ciclo de vida de ML.

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

One or Two Things We know about Concept Drift A Survey on Monitoring Evolving Environments

One or Two Things We know about Concept Drift A Survey on Monitoring Evolving Environments

1. Introdução ao Problema de Drift

- **Drift** é a mudança na distribuição dos dados ao longo do tempo, que compromete modelos de aprendizado de máquina.
- Detectar, localizar e explicar o drift é essencial para manter sistemas de ML confiáveis em cenários reais.
- O artigo divide o estudo em três grandes blocos:
    1. **Drift Detection**
    2. **Drift Localization**
    3. **Drift Explanations**

2. Drift Detection

Conceito

- Objetivo: identificar **quando** ocorre uma mudança significativa na distribuição dos dados.
- Pergunta central: “Houve drift ou não?”

Esquema Geral

1. **Aquisição de dados**: seleção de janelas (sliding, fixed, growing).
2. **Construção de descritores**: representação dos dados (ex.: estatísticas, embeddings, árvores).
3. **Cálculo da dissimilaridade**: comparação entre janelas para medir mudança.
4. **Normalização**: aplicar testes estatísticos (p-valor, bootstrap) para garantir robustez.

Observações

- Usa normalmente janelas de dados antes e depois do ponto suspeito.

- Métodos podem ser simples (médias, variâncias) ou mais sofisticados (modelos ML, embeddings).

- A saída é geralmente um score de dissimilaridade indicando presença de drift.

3. Drift Localization

Conceito

- Objetivo: identificar **onde** no espaço de dados o drift ocorreu.
- Mais detalhado que a detecção, pois segmenta regiões ou pontos específicos.

Esquema Geral

1. **Aquisição de dados**: seleção das amostras que serão analisadas.
2. **Construção de descritores**: gerar representações que permitam verificar regiões/pontos (ex.: árvores, k-NN).
3. **Cálculo da dissimilaridade**: medir drift em **regiões** do espaço de dados, não apenas globalmente.
4. **Normalização**: ajustar scores para obter conclusões válidas (ex.: bootstrap).

Abordagens Exemplares

- **kdq-Tree**: usa estrutura de árvores kd-tree para dividir espaço e medir diferenças entre regiões.
- **LDD-DIS**: baseado em vizinhos próximos, calcula o grau de drift local (Local Drift Degree).
- **Model-based**: treina modelos de classificação e analisa a saída para localizar drift.
- **Drift Segmentation**: divide o espaço em segmentos homogêneos e avalia o drift em cada segmento.

Resultados práticos

- Localização é mais difícil do que detecção e demanda maior quantidade de dados.

- Diferenças de alinhamento de eixos, intensidade do drift, número de amostras e dimensionalidade afetam fortemente os métodos.

4. Drift Explanations

Conceito

- Objetivo: explicar o drift de forma **compreensível para humanos**.
- Pergunta central: "Por que ocorreu drift e de que forma?"

Características

- Explicações são mais complexas e dependem do contexto.
- Necessitam detalhar quais variáveis mudaram, como mudaram e como essas mudanças se relacionam.

Esquema Geral

1. **Aquisição de dados**: seleção de amostras para explicação.
2. **Construção de descritores**: criar representações adequadas (árvores, histogramas, embeddings).
3. **Geração de explicações**: cálculo da intensidade da mudança, correlações ou visualizações.

Exemplos de Métodos

- **Feature-Based Drift Explanations**: medem intensidade de mudança em variáveis específicas.
- **ConceptExplorer**: ferramenta de visualização para séries temporais.
- **Parallel Histograms**: mostram distribuições antes/depois lado a lado.
- **Model-Based Explanations**: usam o modelo treinado como descritor e aplicam técnicas de XAI (ex.: LIME, SHAP, contrafactuais).

Técnicas de Explicação

- **Modelos interpretáveis**: árvores e regressões lineares (simples de entender, mas limitados).
- **Redução de dimensionalidade discriminativa**: projeta dados para visualização e entendimento global.
- **Importância de features global**: avalia quais variáveis mais contribuíram para o drift.
- **Importância local**: análise em instâncias específicas (ex.: LIME, saliency maps).
- **Contrafactuais**: mostram pares de exemplos para ilustrar como o drift alterou pontos semelhantes.

5. Considerações Experimentais

- Experimentos mostraram que:
    - **Model-based** é geralmente mais eficaz, mas sensível ao alinhamento dos dados.
    - **kdq-Tree** e **LDD-DIS** têm limitações, mas funcionam em casos específicos.
    - Quanto maior a intensidade do drift, mais fácil detectá-lo e explicá-lo.
    - Alta dimensionalidade torna o problema muito difícil sem técnicas de seleção de features.

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 9 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Danielle Tavares da Silva

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

MLOps - Tema: Concept Drift

Nesta Semana, li o artigo *"Unsupervised Concept Drift Detection based on Parallel Activations of Neural Network"*, que apresenta o método **PADD (Parallel Activations Drift Detector)** para detecção **não supervisionada** de concept drift.
O PADD utiliza as ativações de uma rede neural não treinada e compara suas distribuições entre lotes consecutivos por meio de testes estatísticos. Diferenças significativas indicam a ocorrência de drift, dispensando o uso de rótulos.

Reproduzi os experimentos disponíveis no GitHub (https://github.com/w4k2/padd) e estudei por cima o código para entender a implementação dos testes e da lógica de detecção.

Experimento 1
Encontrar bons hiperparâmetros do PADD (*alpha*, *th*) em um cenário controlado, observando quais combinações geram um número de alarmes compatível com os drifts — equilibrando sensibilidade e falsos positivos.
• **alpha**: nível de significância dos t-tests (p < alpha ⇒ "drift"). Menor → mais conservador; maior → mais sensível.
• **th**: fração mínima de testes "significantes" para declarar drift.

Experimento 2
Comparar o desempenho do PADD com outros detectores, como MD3, OCDD, CDDD (não supervisionados) e ADWIN/DDM/EDDM (supervisionados), em múltiplos cenários variando número de *features*, tipo e frequência de drift.

Avalia quantos drifts são detectados, com que rapidez e quantos falsos positivos ocorrem.

Unsupervised Concept Drift Detection based on Parallel Activations of Neural Network (PADD)

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima semana, pretendo aprofundar o estudo sobre como integrar métricas e métodos de detecção de drift dentro de uma arquitetura de MLOps.
O objetivo é entender como esses detectores (como o PADD, ADWIN ou DDM) podem ser incorporados de forma prática em pipelines de produção, monitorando continuamente os modelos e os dados em tempo real.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Unsupervised Concept Drift Detection based on Parallel Activations of Neural Network (PADD)

Unsupervised Concept Drift Detection based on Parallel Activations of Neural Network (PADD)

Contexto e Motivação

- **Problema central:**
  Detectar **concept drift** (mudança nos padrões de dados ao longo do tempo) em **data streams**.

- **Desafio:**
  A maioria dos detectores **assume acesso imediato a rótulos (labels)**, o que é **irrealista** em cenários reais por:

  - Alto custo de rotulação
  - Atraso na obtenção de rótulos
  - Escassez de dados rotulados

- **Proposta:**
  Criar um **método não supervisionado**, sem uso de rótulos, que **detecte drifts** com base nas **ativações internas de uma rede neural não treinada**.

Conceito de Concept Drift

Dividido em três eixos principais:

Eixo	Tipos	Descrição
Impacto na classificação	*Real* e *Virtual*	Real altera fronteira de decisão; Virtual muda distribuição sem afetar rótulos
Dinâmica da mudança	*Sudden*, *Gradual*, *Incremental*	Mudança abrupta ou transição contínua
Recorrência	*Recorrente*	Conceitos que reaparecem ciclicamente (ex: sazonalidade)

Trabalhos Relacionados

Estratégias

- **Continuous rebuild:** modelo atualizado continuamente (exige labels).
- **Triggered rebuild:** atualização apenas quando o drift é detectado.

Tipos de detectores

1. **Supervisionados:** DDM, EDDM, ADWIN, Paired Learners, DDD (ensembles).
2. **Não supervisionados (sem labels):**
    - Baseados em **distribuição dos dados**:
        - NN-DVI (Nearest Neighbor Density), GC3 (Grid-based Clustering), CDDD (Centroid Distance), FAAD (Anomaly Detection).
    - Baseados em **modelo/classificador**:
        - MD3 (Margin Density), CDBD (Confidence Distribution Batch).

Proposta — *Parallel Activations Drift Detector (PADD)*

Ideia principal

Usar **as ativações de uma rede neural aleatoriamente inicializada (não treinada)** para capturar mudanças na distribuição dos dados.

- **Sem labels**
- **Sem treino**
- **Baseado em estatística (teste t de Student)**

Intuição

- A rede neural atua como um **mapeamento não linear fixo**.
- As **distribuições das ativações** refletem a estrutura dos dados.
- Quando o conceito muda, as **atividades médias e variações** mudam.

- Compara ativações entre lotes consecutivos via **testes estatísticos repetidos**.

Algoritmo (resumo do pseudocódigo)

Entrada: Fluxo de dados dividido em lotes (DS_1, DS_2, …, DSk)

1. Obter ativações do lote atual: c ← NN(DSk)
2. Comparar ativações do lote atual com histórico C:
    - Executar **r** repetições do teste t em cada **e** saída da rede.
    - Se fração de testes significativos > **θ (threshold) → Drift detectado**
    - Limpar buffer histórico C
3. Armazenar c no histórico.

Parâmetros principais:

- α — nível de significância do teste
- θ — fração mínima de testes que indicam diferença
- r — número de repetições do teste
- s — tamanho da amostra
- e — número de saídas da rede

Experimentos

Configuração

- **Dados sintéticos** (stream-learn library)
- 250 lotes × 200 amostras cada
- 240 streams com diferentes combinações:
    - 3, 5, 10, 15 drifts
    - 30, 60, 90 features
    - drifts súbitos e graduais

Medidas de avaliação

Medida	Interpretação
D1	Distância média entre cada detecção e o drift real (penaliza falsos positivos)

D2	Distância média de cada drift real até a detecção mais próxima (penaliza falsos negativos)
R	Razão ajustada entre número de drifts reais e número de detecções (equilíbrio geral)

Ajuste de Hiperparâmetros

- Testou-se 15 valores de α (0.03–0.2) e 10 de θ (0.1–0.3).
- Configuração ótima:
    - **Drifts graduais:** α = 0.13, θ = 0.26
    - **Drifts súbitos:** α = 0.07, θ = 0.19

Comparação com Métodos de Referência

Métodos comparados:

Categoria	Método	Tipo
Não supervisionado	MD3, OCDD, CDDD, **PADD (proposto)**	Sem labels
Supervisionado	ADWIN, DDM, EDDM	Requerem labels

Resultados

- **PADD** mostrou desempenho **estável e competitivo**, especialmente:
    - Melhor em **D1 (precisão na localização dos drifts)**
    - Segundo melhor em **D2 e R**, empatando com ADWIN e OCDD.
- **CDDD** teve bons resultados, mas **falhou em detectar drifts frequentes**.
- **DDM e EDDM** geraram **muitos falsos positivos**.

Conclusão dos testes estatísticos (Nemenyi + Wilcoxon):
PADD foi o melhor método geral entre os avaliados.

APÊNDICE 4

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **16 de out. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]**:

Danielle Tavares da Silva

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Área: **MLOps** - Tema: **Detecção de Concept Drift**

Nesta Semana, utilizei como base o artigo **"Mastering Data Drift Detection with Google Vertex AI: A Step-by-Step Guide for Beginners"** , que apresenta um tutorial prático de como configurar um pipeline completo de **monitoramento de drift de dados** no **Vertex AI**. O artigo demonstra, passo a passo, como treinar um modelo de Machine Learning, implantá-lo no Vertex AI, habilitar o Model Monitoring.

Seguindo o tutorial, desenvolvi um modelo de detecção de fraudes com dados que eu gerei com auxílio do ChatGPT, realizei o treinamento localmente e o deploy no Vertex AI, e avancei até o final do Step 2, que corresponde à ativação do monitoramento.

Tive algumas dificuldades técnicas durante o processo, o que me impediu de concluir todas as etapas nesta Semana. Acredito que cometi algum erro de configuração no meio do caminho, o que acabou dificultando o avanço até a parte final do tutorial.

Material de apoio - Semana 7

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para a próxima Semana, o objetivo é refazer o que havia feito para corrigir possíveis falhas e, em seguida, concluir o Step 3 do tutorial, que envolve testar a detecção de drifts na prática, validando o comportamento do modelo diante de alterações nos dados de entrada. Além disso, pretendo implementar em produção o método de detecção de concept drift estudado na Semana anterior (PADD).

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 22 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Danielle Tavares da Silva

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta Semana, eu concluí o teste realizado no Vertex AI com base no material: Mastering Data Drift Detection with Google Vertex AI: A Step-by-Step Guide for Beginners | by Henri TO, PhD.
Complementei o material de apoio:
Material de apoio - Semana 7 e 8

Meu objetivo era implementar em produção a métrica estudada anteriormente, o PADD (Parallel Activations Drift Detector), que identifica mudanças conceituais (concept drift) ao comparar as ativações internas de uma rede neural entre diferentes períodos, sem necessidade de rótulos. No entanto, o Vertex AI não permite o uso de métricas personalizadas de monitoramento, o que impossibilitou a aplicação direta da PADD nesse ambiente.

Por isso, busquei alternativas que oferecessem maior flexibilidade e encontrei o tutorial Step-by-Step Guide to Drift Detection in Machine Learning Pipelines with Prefect and MLFlow | by Dr. Ernesto Lee, que utiliza ferramentas que permitem personalizar as métricas de detecção. Iniciei a replicação desse pipeline, mas ainda não consegui concluir completamente a implementação.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima Semana, vou estudar com mais profundidade as ferramentas envolvidas (Prefect e MLFlow) e documentar esse processo, de modo a compreender melhor suas integrações e permitir a implementação efetiva da métrica PADD, além de interpretar adequadamente os resultados obtidos.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Material de apoio - Semana 7 e 8

Material de apoio - Semana 7 e 8

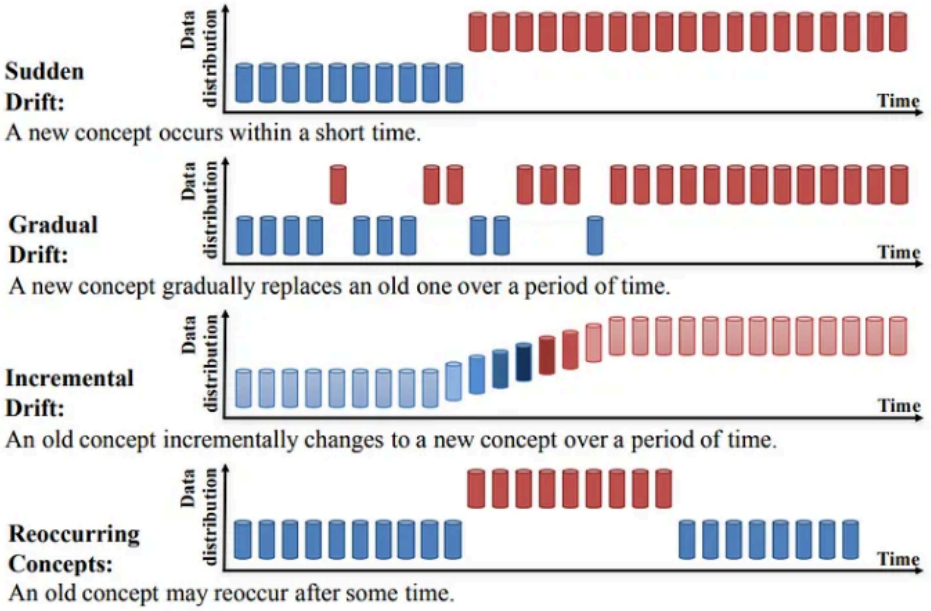
Documentando processos de colocar em produção um modelo AutoML no Google Vertex AI, seguindo [Mastering Data Drift Detection with Google Vertex AI: A Step-by-Step Guide for Beginners]

Google Vertex AI Model Monitoring é um recurso que nos ajuda a acompanhar e manter o desempenho de modelos de *machine learning* (ML) em produção. Ele monitora continuamente as predições para detectar

Feature Skew(Desvio de Atributo): ocorre quando há **inconsistências entre as features usadas durante o treinamento do modelo e as disponíveis durante a implantação (serving)**.

Isso pode acontecer devido a diferenças no modo como as features são processadas, transformadas ou até mesmo ausentes durante a inferência.

Tipos de Concept Drift:



Anomalies: são **desvios inesperados nos padrões de dados** que não estão alinhados com o

comportamento normalmente observado. As anomalias podem representar **eventos raros ou valores atípicos** que o modelo não foi treinado para lidar, como transações fraudulentas, falhas em equipamentos ou mudanças súbitas no comportamento de um sistema.

Etapas gerais para implementar o monitoramento de modelos no Google Vertex AI

1. Implantar o modelo no Vertex AI

- Hospedamos o modelo treinado no Vertex AI Prediction para inferência em tempo real ou em lote. Isso garante que o modelo esteja acessível para servir previsões.

2. Ativar o monitoramento do modelo

- Definimos um *baseline* de dados (geralmente seus dados de treinamento) para servir como referência para comparações futuras.
- Estabelecemos limites para detecção de *data drift* ou anomalias, especificando a faixa aceitável para os dados de entrada ou resultados das previsões.
- Configuramos a frequência de monitoramento (por exemplo, a cada hora, diariamente ou semanalmente) para determinar com que frequência o sistema verifica mudanças ou problemas.

3. Monitorar e responder a alertas

- O Vertex AI coleta e avalia continuamente os dados de entrada em tempo real, comparando-os com o *baseline* para detectar variações.
- Alertas são acionados quando o *data drift* ou as anomalias ultrapassam os limites definidos, permitindo intervenções e ajustes oportunos no modelo.

Passo a passo

Step 1: Deploy Your Model to an Endpoint

Antes de configurar o monitoramento de *data drift*, o modelo precisa estar implantado em um **endpoint** ativo.

O tutorial oferece duas opções para isso:

1. **Treinar um modelo do zero** seguindo o artigo *"Build a Simple Fraud Detection System on GCP (Part 2/3)"*, que ensina a criar um modelo simples diretamente no Google Cloud.

2. **Importar um modelo já treinado** para o **Vertex AI Model Registry**, associando-o a um contêiner de predição. Esse contêiner pode ser:

    - um dos **prebuilt containers** fornecidos pela Google (como o de XGBoost, Scikit-Learn etc.), ou

    - um contêiner personalizado hospedado no **Artifact Registry**.

Escolhi usar como base o tutorial *"Build a Simple Fraud Detection System on GCP (Part 2/3)"*, mas treinar localmente e importar o modelo já treinado para o Vertex AI Model Registry

Estrutura do Projeto

```
treino/
│
├── data/
│   └── transactions.csv              ← Dados de entrada
│
├── beam_feature_pipeline_local.py    ← Pipeline Apache Beam (gera features)
├── prepare_features.py               ← Consolida features e gera Parquet
├── train.py                          ← Treina modelo XGBoost
├── serve.py                          ← API local (FastAPI) para testar predições
├── make_data.py                      ← Gera dados sintéticos
│
└── out/
    ├── declined_transactions_features-00000-of-00001.jsonl
    ├── geo_changes_features-00000-of-00001.jsonl
    ├── outlier_features-00000-of-00001.jsonl
    ├── featured_transactions.parquet
    └── fraud_model.joblib            ← Modelo treinado final
```

Descrição dos Arquivos e Códigos

make_data.py

Gera dados sintéticos para teste local.

Esse script cria 500 transações com colunas básicas (transaction_id, amount, status, is_fraud) para treinar o modelo.

```
import pandas as pd, numpy as np
from datetime import datetime, timedelta

rng = np.random.default_rng(7)
rows = []
start = datetime(2025,1,1,12,0,0)
ips = ["8.8.8.8","1.1.1.1","201.48.0.1","189.6.75.1"]
cards = [f"card_{i:04d}" for i in range(1,51)]

for i in range(500):
    t = start + timedelta(minutes=int(i*3 + rng.integers(0,3)))
    amount = float(np.round(rng.uniform(5, 800), 2))
    status = rng.choice(["approved","declined"], p=[0.85,0.15])
    # rótulo: fraudes mais prováveis qdo alto valor + decline/variação
    is_fraud = int((amount > 450 and status=="declined") or (amount>700 and rng.random()<0.4))
    rows.append({
        "transaction_id": f"tx_{10000+i}",
        "payment_details": rng.choice(cards),
        "ip_address": rng.choice(ips),
        "timestamp": t.isoformat(),
        "transaction_status": status,
        "amount": amount,
        "is_fraud": is_fraud
    })

df = pd.DataFrame(rows)
df.to_csv("data/transactions.csv", index=False)
print(df.shape, df['is_fraud'].value_counts())
```

beam_feature_pipeline_local.py

Cria features com Apache Beam.

O pipeline lê o CSV, calcula:

- número de recusas (declined_count)
- mudanças de IP (geo_changes_count)
- valores atípicos (outlier_amount e z_score)

E salva cada conjunto de features em arquivos .jsonl na pasta out/.

```
import apache_beam as beam

from apache_beam.options.pipeline_options import PipelineOptions, StandardOptions

from apache_beam.transforms.window import FixedWindows

import csv, io, os, json, datetime

import numpy as np

import geoip2.database



HEADERS = [

   "transaction_id",

   "payment_details",

   "ip_address",

   "timestamp",

   "transaction_status",

   "amount",

   "is_fraud"

]
```

```
INPUT_CSV = "./data/transactions.csv"

GEOIP_DB = "./data/GeoLite2-City.mmdb"

OUT_DIR = "./out"



os.makedirs(OUT_DIR, exist_ok=True)

def parse_csv_line(line):

    import csv

    import io

    parts = next(csv.reader([line]))

    # se a linha tiver menos colunas que o esperado, completa com vazio

    if len(parts) < len(HEADERS):

        parts = parts + [""] * (len(HEADERS) - len(parts))

    row = dict(zip(HEADERS, parts))

    row["amount"] = float(row["amount"])

    row["timestamp"] = datetime.datetime.fromisoformat(row["timestamp"])

    # is_fraud pode existir ou não — tratamos como opcional

    row["is_fraud"] = int(row.get("is_fraud", 0) or 0)

    return row



class AddGeolocation(beam.DoFn):

    def setup(self):
```

```
        self.reader = geoip2.database.Reader(GEOIP_DB)

    def process(self, x):

        try:

            loc = self.reader.city(x["ip_address"])

            x["country"] = loc.country.name

            x["city"] = loc.city.name

        except Exception:

            x["country"] = "Unknown"

            x["city"] = "Unknown"

        yield x



class TrackGeoChanges(beam.DoFn):

    def process(self, kv):

        payment_details, txs = kv

        txs = sorted(txs, key=lambda t: t["timestamp"])

        geos = [(t["country"], t["city"]) for t in txs]

        changes = sum(1 for i in range(1, len(geos)) if geos[i] != geos[i-1])

        yield {"payment_details": payment_details, "geo_changes_count": changes}



class ComputeDeclined(beam.DoFn):

    def process(self, kv):
```

```
        payment_details, txs = kv
        declined = sum(1 for t in txs if t["transaction_status"].lower() == "declined")
        window_end = max(t["timestamp"] for t in txs)
        yield {
            "payment_details": payment_details,
            "declined_count": declined,
            "window_end": window_end.isoformat()
        }


class DetectOutliers(beam.DoFn):
    def process(self, kv):
        payment_details, amounts = kv
        mean = float(np.mean(amounts))
        std = float(np.std(amounts))
        for a in amounts:
            z = (a - mean) / std if std > 0 else 0.0
            if abs(z) > 2.0:
                yield {"payment_details": payment_details, "outlier_amount": float(a), "z_score": float(z)}


def to_jsonl(path, pcoll):
    _ = (pcoll
```

```
        | f"to_json_{path}" >> beam.Map(json.dumps)
        | f"write_{path}" >> beam.io.WriteToText(
                file_path_prefix=os.path.join(OUT_DIR, path),
                file_name_suffix=".jsonl",
                num_shards=1))


def run():
    options = PipelineOptions()
    options.view_as(StandardOptions).streaming = False

    with beam.Pipeline(options=options) as p:
        # Read CSV as text and parse with DictReader once per line
        lines = (p
                | "read all (skip header)" >> beam.io.ReadFromText(INPUT_CSV, skip_header_lines=1))

        dict_rows = lines | "parse line" >> beam.Map(parse_csv_line)

        with_ts = dict_rows | "add geo" >> beam.ParDo(AddGeolocation())

        # Declined in janelas fixas de 10 min
        declined = (
```

```
    with_ts
    | "key by payment" >> beam.Map(lambda x: (x["payment_details"], x))
    | "window 10m" >> beam.WindowInto(FixedWindows(600))
    | "group" >> beam.GroupByKey()
    | "declined" >> beam.ParDo(ComputeDeclined())
)


# Mudanças de geolocalização
geo_changes = (
    with_ts
    | "key by payment (geo)" >> beam.Map(lambda x: (x["payment_details"], x))
    | "group (geo)" >> beam.GroupByKey()
    | "geo changes" >> beam.ParDo(TrackGeoChanges())
)


# Outliers por valor
outliers = (
    with_ts
    | "key for outliers" >> beam.Map(lambda x: (x["payment_details"], x["amount"]))
    | "group amounts" >> beam.GroupByKey()
    | "detect outliers" >> beam.ParDo(DetectOutliers())
```

```
    )



    to_jsonl("declined_transactions_features", declined)

    to_jsonl("geo_changes_features", geo_changes)

    to_jsonl("outlier_features", outliers)



if __name__ == "__main__":

  run()
```

prepare_features.py

Une os arquivos JSONL e gera um único dataset .parquet.

```
import pandas as pd, json, os, glob

OUT_DIR = "./out"
INPUT_CSV = "./data/transactions.csv"

def read_jsonl(path):
  rows = []
  with open(path, "r", encoding="utf-8") as f:
    for line in f:
      line = line.strip()
      if line:
        rows.append(json.loads(line))
  return pd.DataFrame(rows)

def main():
  base = pd.read_csv(INPUT_CSV, parse_dates=["timestamp"])
```

```
    d_path = glob.glob(os.path.join(OUT_DIR, "declined_transactions_features-*.jsonl"))[0]
    g_path = glob.glob(os.path.join(OUT_DIR, "geo_changes_features-*.jsonl"))[0]
    o_path = glob.glob(os.path.join(OUT_DIR, "outlier_features-*.jsonl"))[0]

    d = read_jsonl(d_path)
    g = read_jsonl(g_path)
    o = read_jsonl(o_path)

    # garante colunas mesmo se algum JSONL estiver vazio
    if d.empty: d = pd.DataFrame(columns=["payment_details","declined_count"])
    if g.empty: g = pd.DataFrame(columns=["payment_details","geo_changes_count"])
    if o.empty: o = pd.DataFrame(columns=["payment_details","outlier_amount","z_score"])

    df = (base
          .merge(d, on="payment_details", how="left")
          .merge(g, on="payment_details", how="left")
          .merge(o, on="payment_details", how="left"))

    for col in ["declined_count","geo_changes_count","outlier_amount","z_score"]:
        if col not in df.columns:
            df[col] = 0.0
        df[col] = df[col].fillna(0.0)

    df.to_parquet("./out/featured_transactions.parquet", index=False)
    print("✅ Salvo em ./out/featured_transactions.parquet")
    print(df.head())

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

train.py

Treina um modelo XGBoost para classificar fraudes.

- lê featured_transactions.parquet
- divide em treino/teste
- ajusta o peso da classe positiva (scale_pos_weight)

- treina com logloss
- salva fraud_model.joblib

```
import pandas as pd, numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from xgboost import XGBClassifier
import joblib

DF_PATH = "./out/featured_transactions.parquet"
LABEL_COL = "is_fraud"

def main():
    df = pd.read_parquet(DF_PATH)

    # garante que todas as colunas de features existam, mesmo se não foram criadas
    feature_cols = ["amount", "declined_count", "geo_changes_count", "outlier_amount", "z_score"]
    for col in feature_cols:
        if col not in df.columns:
            df[col] = 0.0

    X = df[feature_cols].fillna(0.0).astype(float)

    if LABEL_COL not in df.columns:
        raise RuntimeError(f"Coluna de rótulo '{LABEL_COL}' não encontrada.")
    y = df[LABEL_COL].astype(int)
    neg, pos = (y==0).sum(), (y==1).sum()
    scale_pos_weight = (neg / pos) if pos > 0 else 1.0
    print(f"pos={pos}, neg={neg}, scale_pos_weight={scale_pos_weight:.2f}")

    Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.25, random_state=42, stratify=y)

    n_estimators = 200
    model = XGBClassifier(
        n_estimators=n_estimators,
        max_depth=4,
```

```
        learning_rate=0.1,
        subsample=0.8,
        colsample_bytree=0.8,
        eval_metric="logloss",
        n_jobs=4,
        random_state=42,
        scale_pos_weight=scale_pos_weight
    )

    # imprime progresso a cada 10 iterações
    model.fit(
        Xtr, ytr,
        eval_set=[(Xte, yte)],
        verbose=10  # mostra logloss a cada 10 árvores
    )

    preds = model.predict(Xte)
    print("\n", classification_report(yte, preds, digits=4))

    joblib.dump(model, "./out/fraud_model.joblib")
    print("✅ Modelo salvo em ./out/fraud_model.joblib")

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

serve.py

API local para testar o modelo (FastAPI).

```
from fastapi import FastAPI
import joblib, numpy as np
from pydantic import BaseModel

THRESHOLD = 0.35  # ajuste conforme suas métricas

app = FastAPI()
```

```
model = joblib.load("./out/fraud_model.joblib")

@app.get("/")
def root():
    return {"status": "ok", "message": "use POST /predict ou /docs"}

class Item(BaseModel):
    amount: float
    declined_count: float = 0.0
    geo_changes_count: float = 0.0
    outlier_amount: float = 0.0
    z_score: float = 0.0

@app.post("/predict")
def predict(item: Item):
    x = np.array([[item.amount, item.declined_count, item.geo_changes_count, item.outlier_amount,
item.z_score]])
    p = float(model.predict_proba(x)[0, 1]) if hasattr(model, "predict_proba") else float(model.predict(x)[0])
    decision = "review" if p >= THRESHOLD else "approve"
    return {"fraud_probability": p, "threshold": THRESHOLD, "decision": decision}
```

Executar:

uvicorn serve:app --reload

Importação do Modelo Local para o Vertex AI Model Registry

1. Abertura do Cloud Shell e configuração inicial do projeto

Comandos:

- *PROJECT_ID="gen-lang-client-0213204827"*
- *REGION="us-central1"*
- *gcloud config set project [PROJECT_ID]*
- *gcloud config set ai/region [REGION]*
- *BUCKET_NAME="fraud-model-dani-$PROJECT_ID"*

2. Criação do bucket no Cloud Storage

- *gsutil mb -l $REGION gs://$BUCKET_NAME*

Cria um **bucket** no **Google Cloud Storage (GCS)** com o nome:
gs://fraud-model-dani-gen-lang-client-0213204827

O bucket será usado para armazenar os arquivos do modelo que serão importados no Vertex AI.

3. Upload do modelo treinado para o bucket

- *gsutil cp fraud_model.joblib gs://$BUCKET_NAME/model/model.joblib*

O arquivo fraud_model.joblib, gerado localmente após o treinamento, foi enviado para o diretório:
gs://fraud-model-dani-gen-lang-client-0213204827/model/model.joblib

Essa é a estrutura esperada pelo Vertex AI para importar modelos preexistentes.
O arquivo deve se chamar **model.joblib** e estar dentro de uma pasta (neste caso, model/).

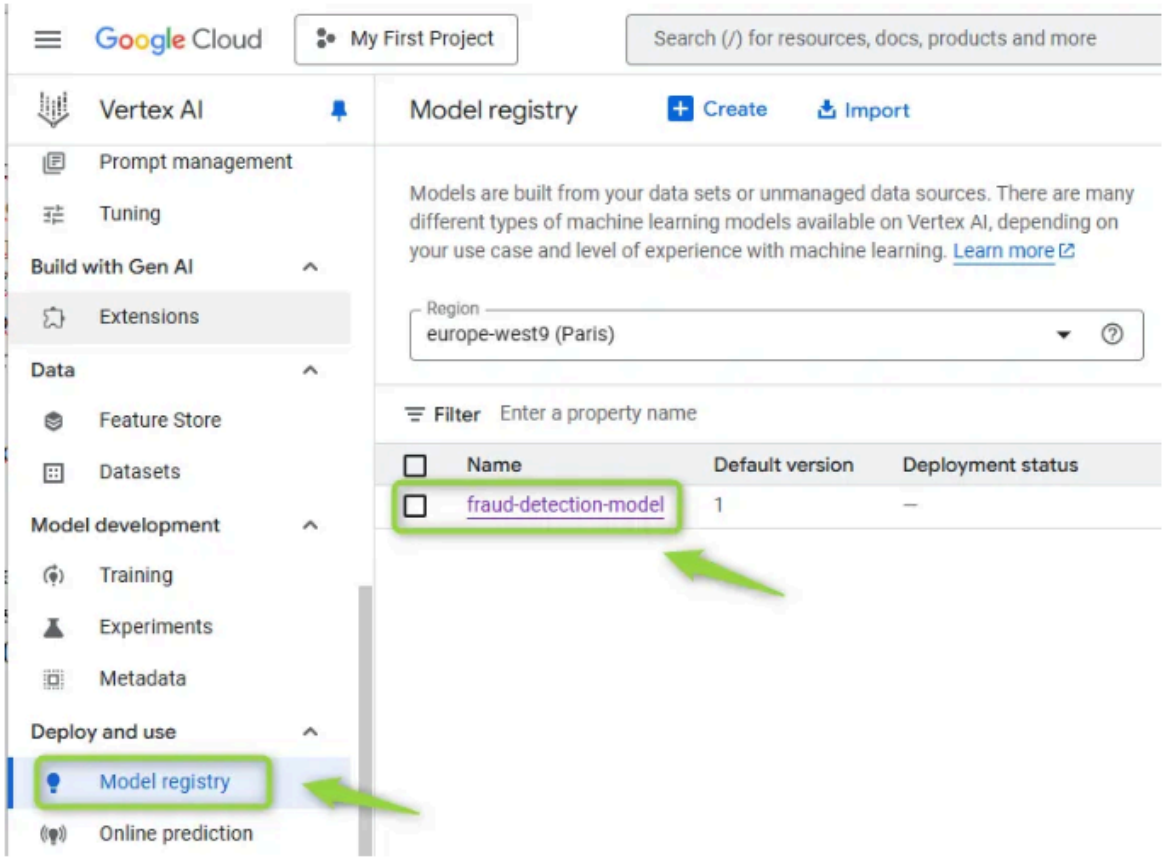
4. Importação do modelo para o Vertex AI Model Registry

- gcloud ai models upload \
  --display-name="fraud-xgb" \
  --artifact-uri="gs://$BUCKET_NAME/model" \
  --container-image-uri="us-docker.pkg.dev/vertex-ai/prediction/xgboost-cpu.2-1:latest"

Descrição dos parâmetros:

- --display-name="fraud-xgb": nome atribuído ao modelo dentro do Vertex AI.
- --artifact-uri: caminho do artefato no GCS (pasta onde está o model.joblib).
- --container-image-uri: define o **container prebuilt** que será usado para servir predições. Neste caso, foi utilizado o container oficial do **XGBoost 2.1** (xgboost-cpu.2-1)

Deploy do modelo treinado



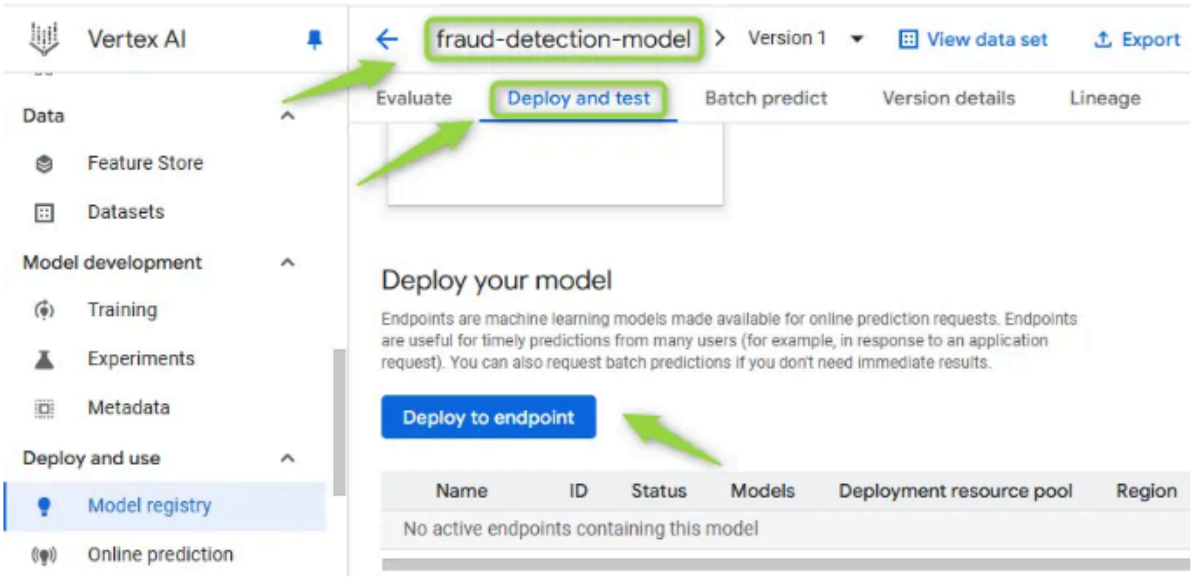
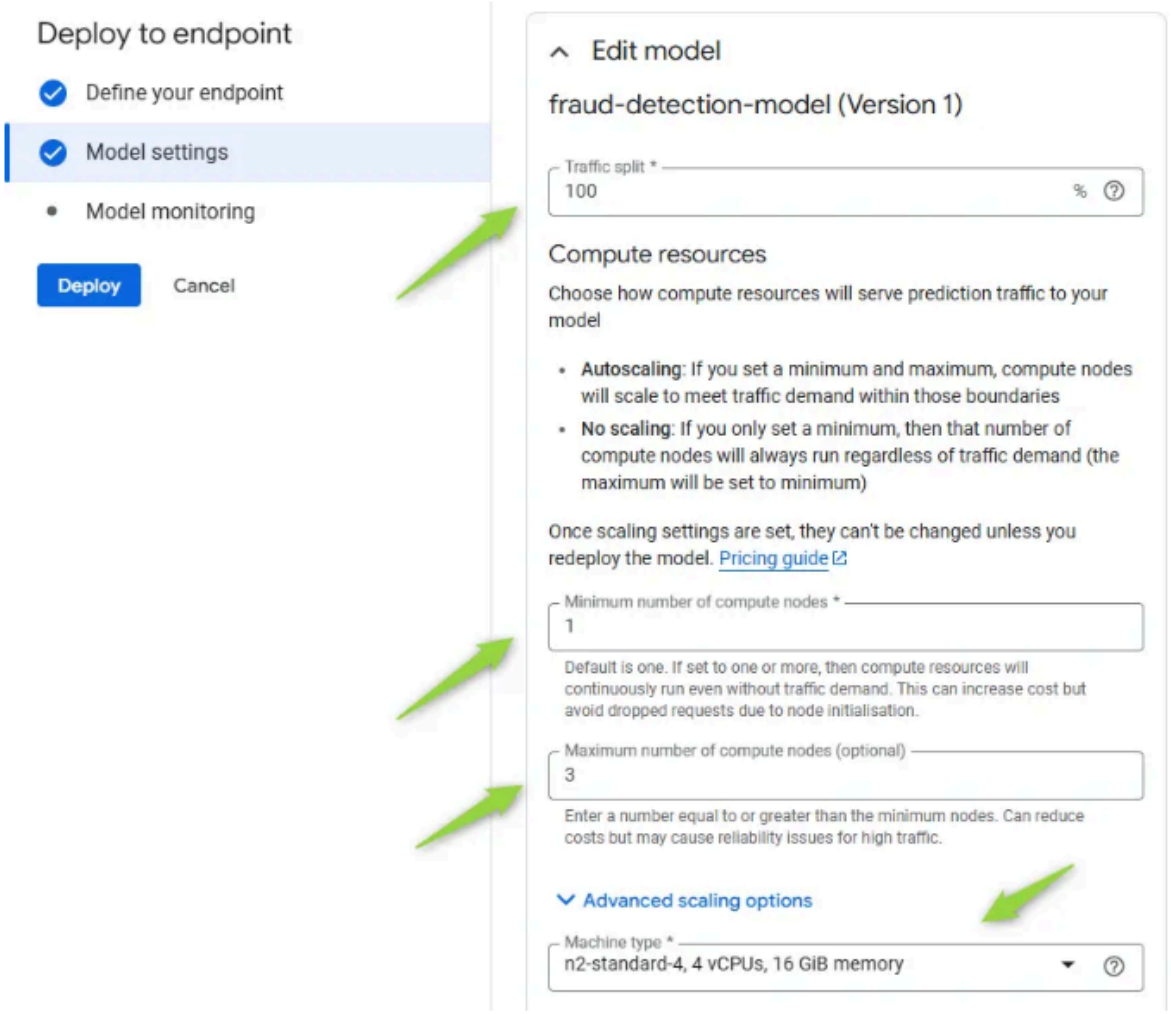


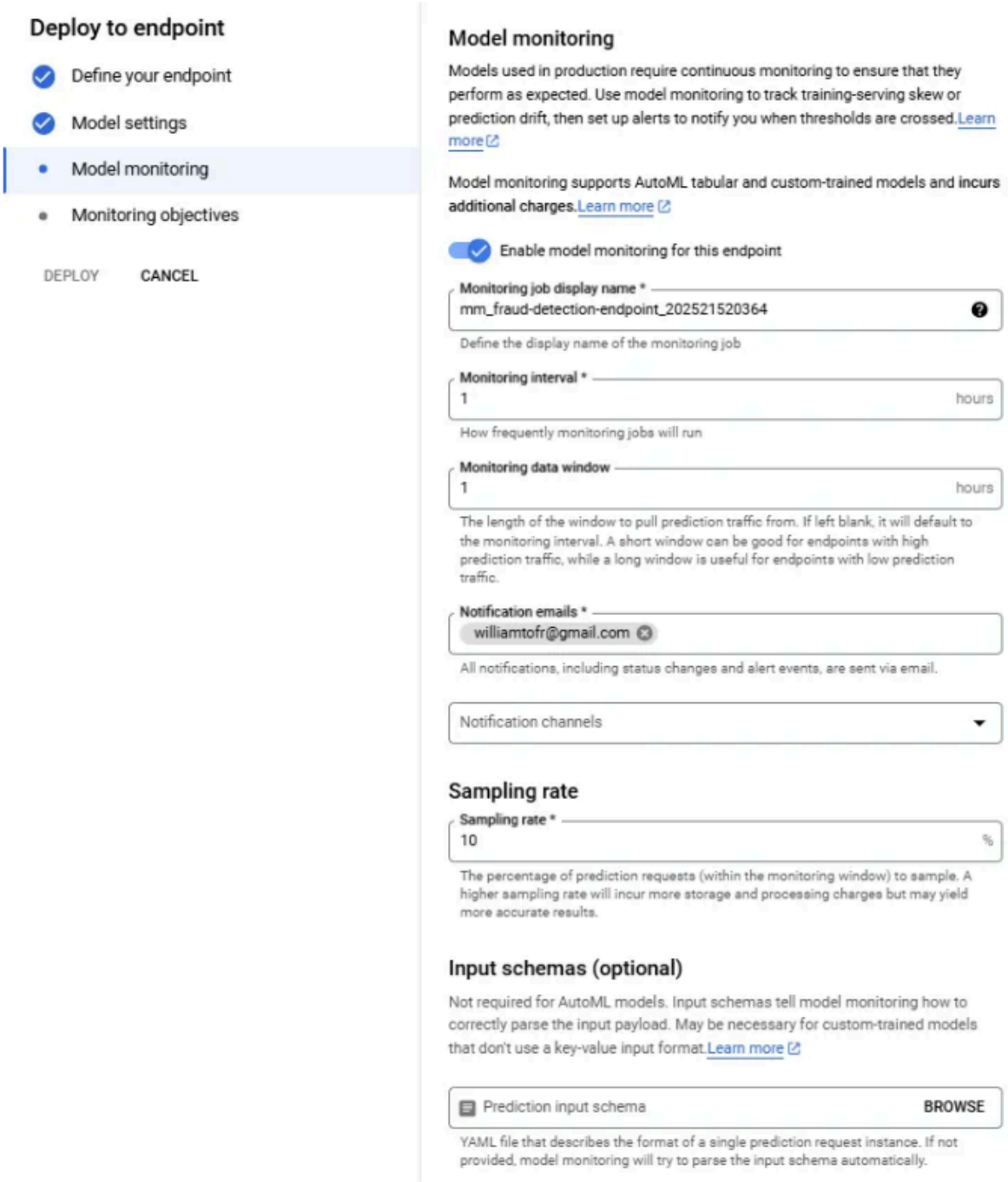
Figure 3: Start deploying your model to endpoint.

- Definir um endpoint



Step 2: Set Up Model Monitoring for Data Drift

- Configurar o monitoramento:



- Escolher os objetivos de monitoramento conforme o tipo de drift que deseja detectar.

a) Prediction Drift Detection

- Monitora **mudanças nas previsões do modelo** ao longo do tempo.
- Útil para identificar variações inesperadas nas saídas do modelo, como classificações de fraude.

  Threshold recomendado: 0.3

  Exemplo:
  O modelo foi treinado com transaction_amount entre **$10 e $1.000**.
  Durante o monitoramento, aparecem valores **acima de $50.000** → possível **data drift**.

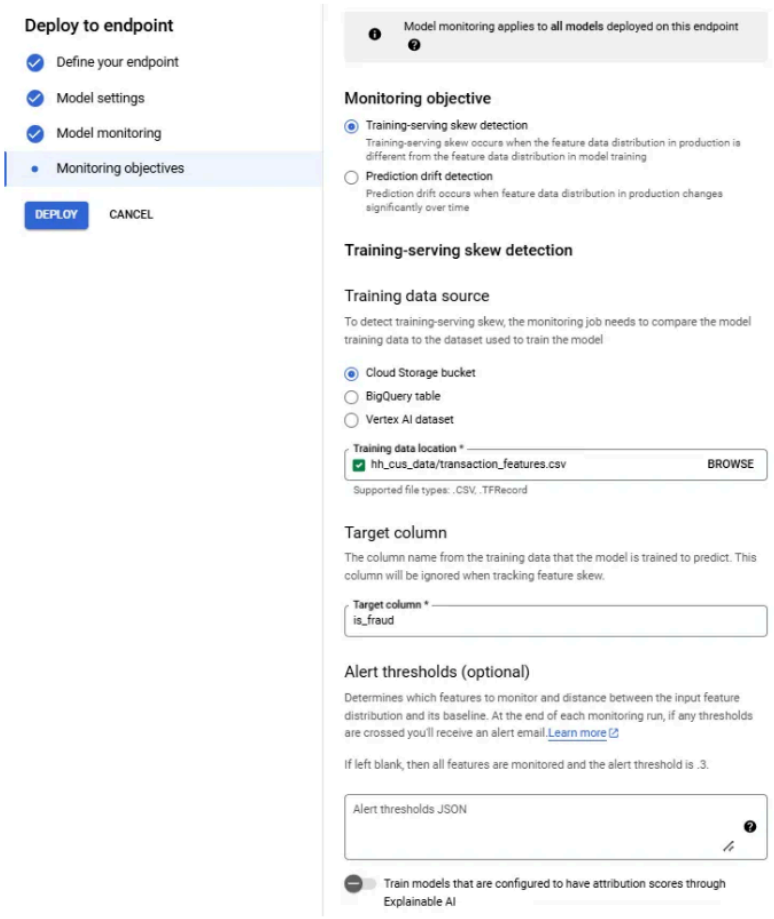
b) Training-Serving Skew Detection

- Compara os **dados de treinamento** com os **dados reais de produção**.

- Detecta discrepâncias entre o que o modelo aprendeu e o que está sendo recebido.

Threshold recomendado: 0.3

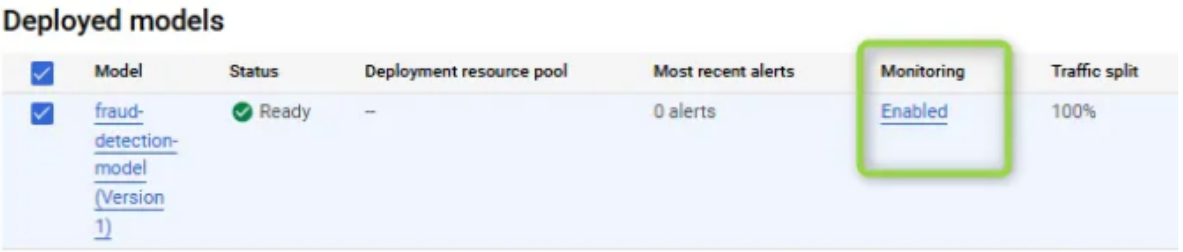
Exemplo:
Durante o treino, o atributo geo_changes_count variava entre **0 e 3** (clientes raramente mudavam de localização).
Agora, surgem valores **acima de 10** → indica **data drift**, pois esse padrão não existia nos dados de treinamento.



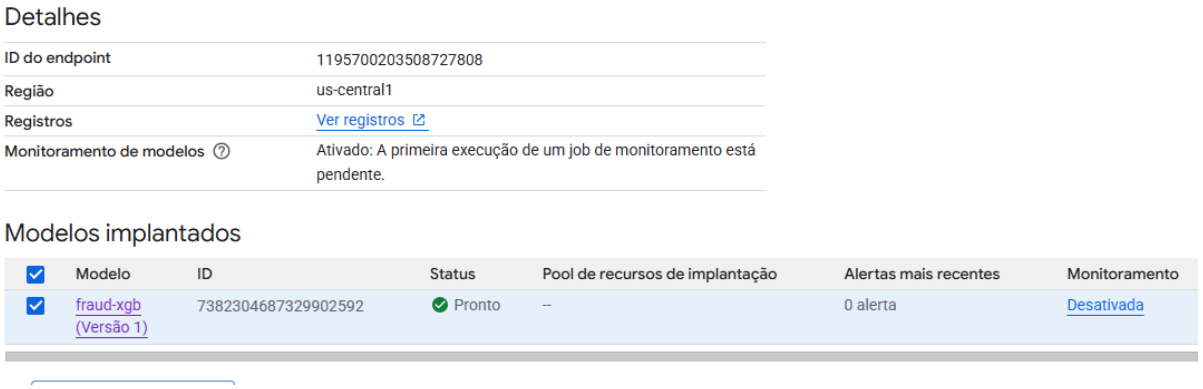
Após definir os objetivos de monitoramento:

1. Clique em **Deploy**.
2. Aguarde alguns minutos para que o monitoramento seja ativado e o job de verificação seja iniciado.

Deveria ficar assim:



Está assim no momento:



- **Tive erro de formatação durante o teste**

Tentativa 2 - Semana 8

Tentando fazer algo mais simples, eu usei o dataset Íris para treinar um modelo simples de regressão logística. Segui novamente o tutorial, e agora sim deu certo.

- Testei com dados com uma distribuição parecida com que ele viu no treinamento
- Quando testei com dados com uma distribuição muito diferente, ele detectou o drift após um 1h (tempo escolhido para rodar a inspeção)
- Recebi esse email informando:

Vertex AI <noreply-vertexai@google.com> ter., 21 de out., 00:55 (há 1 dia)
para mim

Hello Vertex AI Customer,

You are receiving this mail because you are subscribing to the Vertex AI Model Monitoring service.
This mail is just to inform you that there are some anomalies detected in your deployed models and may need your attention.

Basic Information:

Endpoint Name: projects/70680939442/locations/us-central1/endpoints/2125003033258491904
Monitoring Job: projects/70680939442/locations/us-central1/modelDeploymentMonitoringJobs/2617520969043935232
Statistics and Anomalies Root Path(Google Cloud Storage): gs://cloud-ai-platform-689c5309-c02f-4aec-8901-af70bdabb036/model_monitoring/job-2617520969043935232
BigQuery Command: SELECT * FROM `bq://gen-lang-client-0213204827.model_deployment_monitoring_2125003033258491904.serving_predict`

Training Prediction Skew Anomalies (Raw Feature):

Anomalies Report Path(Google Cloud Storage): gs://cloud-ai-platform-689c5309-c02f-4aec-8901-af70bdabb036/model_monitoring/job-2617520969043935232/serving/2025-10-20T22:00/stats_and_anomalies/7688127249526030336/anomalies/training_prediction_skew_anomalies

For more information about the alert, please visit the model monitoring alert page.

Deployed model id: 7688127249526030336

Feature name	Anomaly short description	Anomaly long description
'petal length (cm)'	High approximate Jensen-Shannon divergence between training and serving	The approximate Jensen-Shannon divergence between training and serving is 1 (up to six significant digits), above the threshold 0.3.
'petal width (cm)'	High approximate Jensen-Shannon divergence between training and serving	The approximate Jensen-Shannon divergence between training and serving is 1 (up to six significant digits), above the threshold 0.3.
'sepal length	High approximate Jensen-Shannon divergence between	The approximate Jensen-Shannon divergence between training and serving is 0.946239 (up to six

APÊNDICE 5

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 6 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Danielle Tavares da Silva

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa Semana, estudei sobre o **Prefect** e o **MLflow.**

O **Prefect** é uma ferramenta de orquestração de fluxos de trabalho, usada para automatizar e monitorar tarefas de dados, enquanto o **MLflow** é uma plataforma de gerenciamento de experimentos de machine learning, que permite rastrear métricas, modelos e versões de código.

Segui o tutorial "Step-by-step guide to drift detection in ML pipelines with Prefect and MLflow".

Ao final, eu pude trocar a métrica utilizada no tutorial pela métrica PADD (Parallel Activations Drift Detector), que foi proposta no paper Unsupervised Concept Drift Detection based on Parallel Activations of Neural Network, é uma abordagem não supervisionada para detecção de *concept drift* que utiliza as ativações de uma rede neural não treinada para identificar mudanças nos dados, sem depender de rótulos.

Semana 9

Por fim, pesquisei datasets para testar diferentes métricas de drift e encontrei o repositório driftDatasets, que reúne conjuntos de dados reais e simulados voltados à avaliação e comparação de métodos de detecção de drift, abrangendo diversos tipos de mudanças e contextos experimentais.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Na próxima Semana, o objetivo é testar diferentes métricas de detecção de drift e observar se há impacto na detecção dependendo da métrica utilizada. A ideia é comparar o desempenho de alguns métodos (como KS, PSI, ADWIN, MMD e PADD) utilizando os datasets do repositório driftDatasets em diferentes tipos de drift, avaliando a sensibilidade, tempo de detecção e estabilidade de cada um.

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Semana 9

Drift Detection in Machine Learning Pipelines with Prefect and MLFlow

Eu estou seguindo esse tutorial: [Step-by-Step Guide to Drift Detection in Machine Learning Pipelines with Prefect and MLFlow | by Dr. Ernesto Lee]

Primeiramente eu estudei um pouco do que era o Prefect e MLflow:

Prefect

É uma ferramenta de orquestração de pipelines de dados.
Ele serve para automatizar, agendar e monitorar tarefas de machine learning ou data engineering.

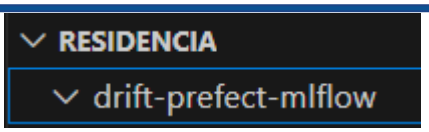
- **Tasks** - São as etapas individuais dentro do fluxo do Prefect. Cada @task é uma função isolada (como um bloco do pipeline).
- **Flows** - São o conjunto organizado de tasks. Um @flow define como as tasks se conectam, é o pipeline em si.

MLflow

É uma ferramenta para gerenciar experimentos de machine learning.
Ele registra tudo o que acontece durante o treinamento e monitoramento de modelos.

Ele armazena:

- Hiperparâmetros usados
- Métricas (acurácia, perda, drift, etc.)
- Modelos treinados (artefatos)
- Logs e versões de cada execução



Passo 1: Setar o Ambiente

- Criar pasta do projeto

```
PS C:\Users\Danielle\Desktop\Residencia> mkdir drift-prefect-mlflow


    Diretório: C:\Users\Danielle\Desktop\Residencia


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        05/11/2025     19:07                drift-prefect-mlflow


PS C:\Users\Danielle\Desktop\Residencia> cd .\drift-prefect-mlflow\
PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow>
```

- Criar e ativar o ambiente virtual

```
PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> python -m venv .venv
PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> .\.venv\Scripts\Activate.ps1
(.venv) PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow>
```

- Instalar as dependências

```
(.venv) PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> pip install "prefect>=3.0" "mlflow==3.2.*" pandas scikit-learn xgboost
pyarrow scipy fastparquet
Collecting prefect>=3.0
  Downloading prefect-3.5.0-py3-none-any.whl (6.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.2/6.2 MB 3.1 MB/s eta 0:00:00
Collecting mlflow==3.2.*
  Downloading mlflow-3.2.0-py3-none-any.whl (25.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.9/25.8 MB 3.6 MB/s eta 0:00:06
```

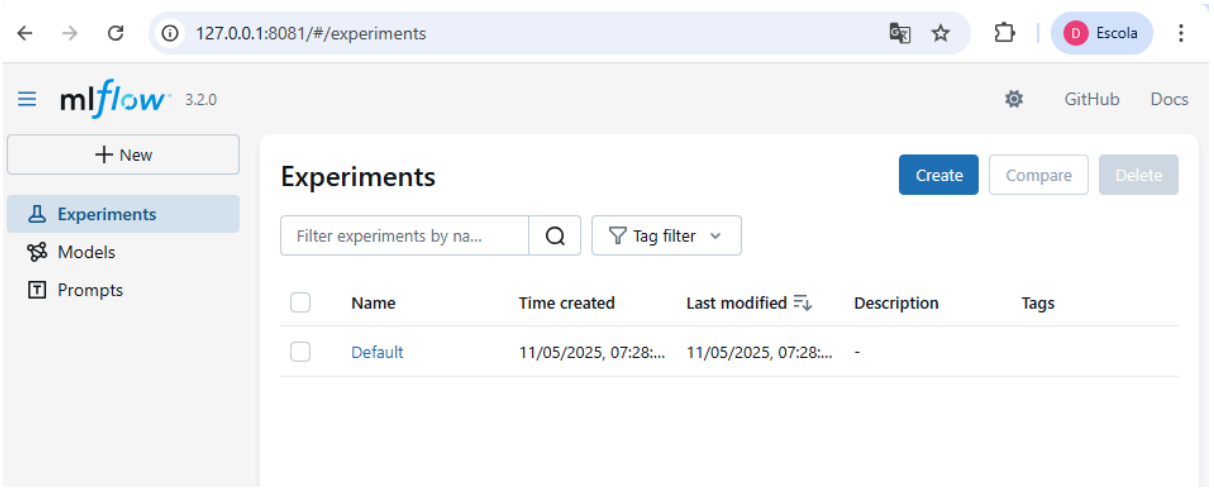
- Iniciar a UI do MLflow (em outro terminal)

```
PS C:\Users\Danielle\Desktop\Residencia> cd .\drift-prefect-mlflow\
PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> .\.venv\Scripts\Activate.ps1
(.venv) PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> mlflow ui --backend-store-uri sqlite:///mlflow.db --default-artifact-
root ./mlruns --port 8081
2025/11/05 19:28:11 INFO mlflow.store.db.utils: Creating initial MLflow database tables...
2025/11/05 19:28:11 INFO mlflow.store.db.utils: Updating database tables
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 451aebb31d03, add metric step
INFO  [alembic.runtime.migration] Running upgrade 451aebb31d03 -> 90e64c465722, migrate user column to tags
INFO  [alembic.runtime.migration] Running upgrade 90e64c465722 -> 181f10493468, allow nulls for metric values
INFO  [alembic.runtime.migration] Running upgrade 181f10493468 -> df50e92ffc5e, Add Experiment Tags Table
INFO  [alembic.runtime.migration] Running upgrade df50e92ffc5e -> 7ac759974ad8, Update run tags with larger limit
INFO  [alembic.runtime.migration] Running upgrade 7ac759974ad8 -> 89d4b8295536, create latest metrics table
```

(esse terminal deve ficar aberto)

se clicarmos aqui:

abre essa página:



- Baixar os scripts do tutorial - eles disponibilizam no github:

orchestrate.py
duration_prediction_original.ipynb

Passo 2 a 4: Enhancing orchestrate.py for Drift Detection

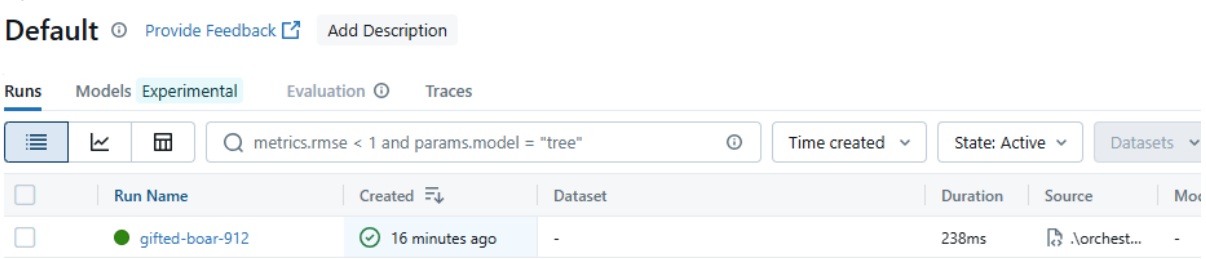
- Se os scripts foram baixados corretamente já foi implementado o que pede.

- O que o orchestrate_monitoring2.py faz:
    - O flow **baixa os dados** de duas janelas temporais (jan/2024 e dez/2023);
    - Extrai features e durations;
    - Roda o **teste KS (Kolmogorov–Smirnov)** entre as distribuições de duração;
    - Registra métricas de **drift (ks_statistic e p_value)** no MLflow;
    - Se o drift for detectado, treina um novo modelo e salva os artefatos;
    - Caso contrário, apenas registra que **não há drift suficiente**.

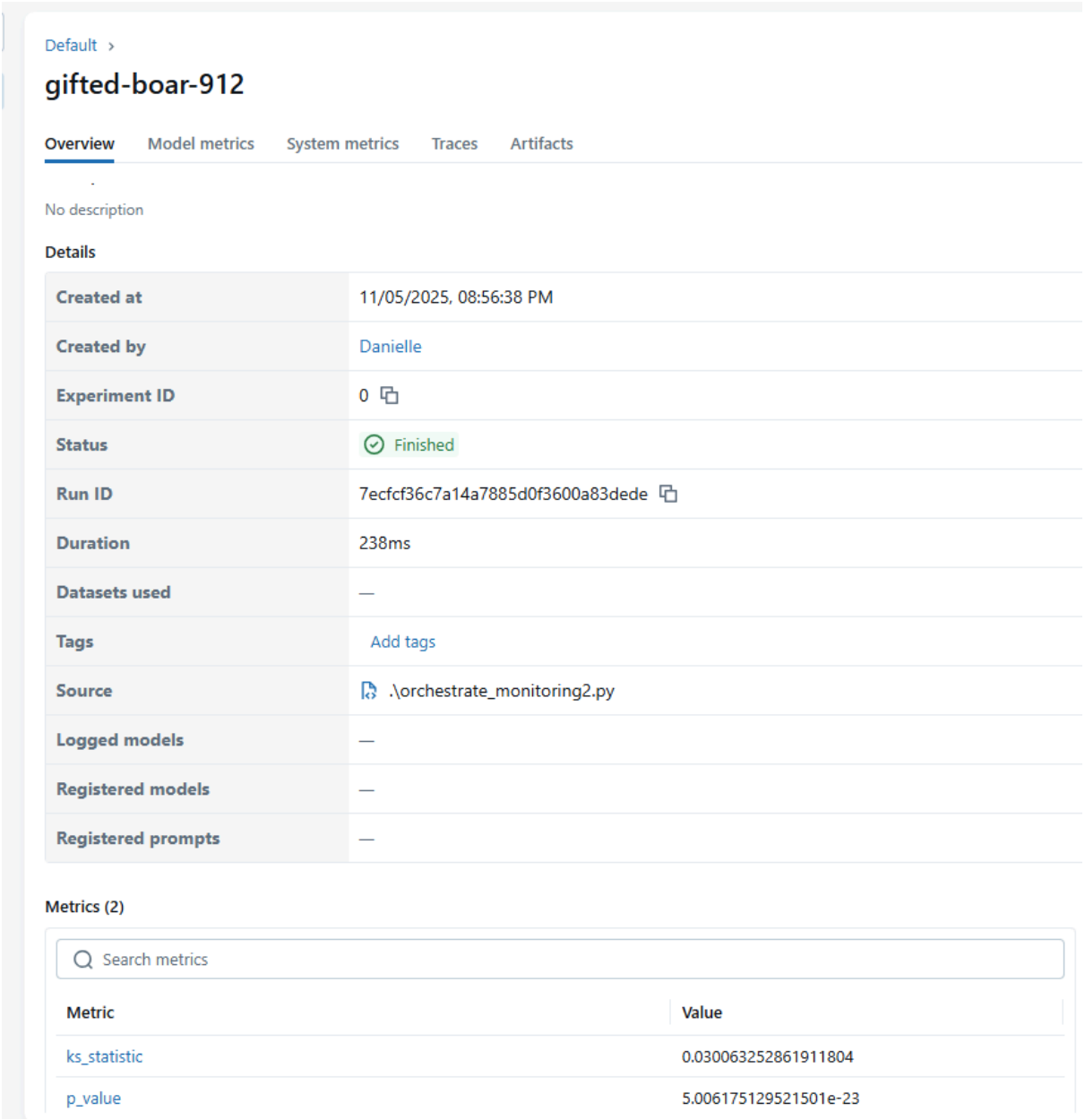
- Para testar:

```
(.venv) PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> python .\orchestrate_monitoring2.py
20:18:58.531 | INFO    | prefect - Starting temporary server on http://127.0.0.1:8869
See https://docs.prefect.io/v3/concepts/server#how-to-guides for more information on running a dedicated Prefect server.
20:19:06.239 | INFO    | Flow run 'eager-stingray' - Beginning flow run 'eager-stingray' for flow 'main-flow'
2025/11/05 20:19:06 WARNING mlflow.utils.autologging_utils: MLflow sklearn autologging is known to be compatible with 1.3.1 <= scikit-learn
 <= 1.7.1, but the installed version is 1.7.2. If you encounter errors during autologging, try upgrading / downgrading scikit-learn to a co
mpatible version, or try upgrading MLflow.
2025/11/05 20:19:08 INFO mlflow.tracking.fluent: Autologging successfully enabled for sklearn.
2025/11/05 20:19:08 WARNING mlflow.utils.autologging_utils: MLflow xgboost autologging is known to be compatible with 2.0.0 <= xgboost <= 3
.0.2, but the installed version is 3.1.1. If you encounter errors during autologging, try upgrading / downgrading xgboost to a compatible v
ersion, or try upgrading MLflow.
2025/11/05 20:19:08 INFO mlflow.tracking.fluent: Autologging successfully enabled for xgboost.
20:19:12.395 | INFO    | Task run 'read_data-2c2' - Finished in state Completed()
20:19:13.614 | INFO    | Task run 'read_data-107' - Finished in state Completed()
20:19:14.274 | INFO    | Task run 'add_features-b41' - Finished in state Completed()
Drift KS statistic: 0.030063252861911804, p-value: 5.006175129521501e-23
Model needs retraining: False
20:19:15.102 | INFO    | Task run 'detect_drift-2f2' - Finished in state Completed()
No retraining required at this time.
20:19:15.143 | INFO    | Flow run 'eager-stingray' - Finished in state Completed()
20:19:15.161 | INFO    | prefect - Stopping temporary server on http://127.0.0.1:8869
View run silent-asp-233 at: http://127.0.0.1:8081/#/experiments/0/runs/abb6d900b55c4999ad0f3bda9340b4ae
View experiment at: http://127.0.0.1:8081/#/experiments/0
```

- O que aconteceu:
    - Prefect orquestrou as tasks (read → features → drift detection).
    - MLflow fez o autolog (registrou métricas, parâmetros, artefatos).
    - O drift foi calculado, mas ficou abaixo do limiar de 0.05, então o modelo não precisou ser re-treinado.
    - A execução foi registrada na interface em: http://127.0.0.1:8081/#/experiments/0



Ele atribui um nome aleatório, se clicarmos nele:



No código

- **Se ks_statistic > 0.05**, o código considera que houve **drift**.
  → Ele **re-treina** o modelo.

- **Se ks_statistic <= 0.05**, o código entende que os dados **não mudaram o suficiente**,
  → Então **não re-treina**

Como podemos perceber o **ks_statistic** está abaixo desse valor, por isso não é re-treinado o modelo.

Prefect

No **Prefect 3**, a UI roda localmente através de um **servidor temporário**.

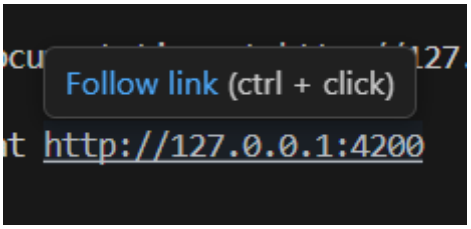
Starting temporary server on http://127.0.0.1:8869

- Existe uma UI rodando, mas ela vive apenas enquanto o flow está ativo.
- Para que ele fique ativo sempre, abra outro terminal (mantenha os outros abertos)

Rode:

```
PS C:\Users\Danielle\Desktop\Residencia> cd .\drift-prefect-mlflow\
PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> .\.venv\Scripts\Activate.ps1
(.venv) PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> prefect server start
Switched to profile 'local'


 ___ ___ ___ ___ ___ ___ _____
| _ \ _ \ __| __| __/ __|_   _|
|  _/   / _|| _|| _| (__  | |
|_| |_|_\___|_| |___\___| |_|

Configure Prefect to communicate with the server with:

    prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api

View the API reference documentation at http://127.0.0.1:4200/docs

Check out the dashboard at http://127.0.0.1:4200
```

Basta clicar para ter acesso a interface:



Passo 5: Agendar para rodar a pipeline

- Você responde algumas perguntas, eu escolhi rodar a pipeline a cada 30 minutos

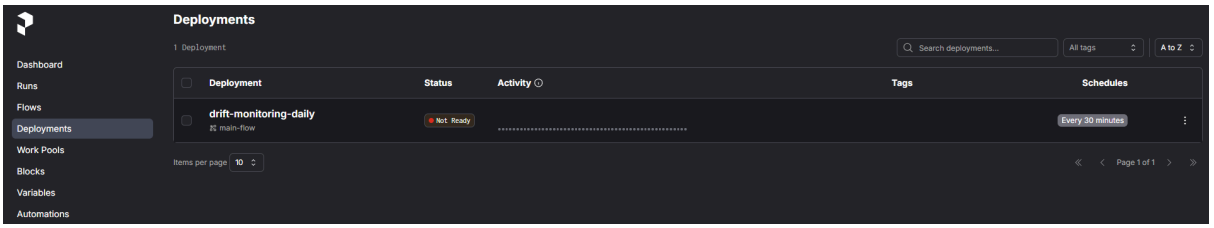
primeiro setar a URL e o worker

```
(.venv) PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> $env:PREFECT_API_URL = "http://127.0.0.1:4200/api"
>> prefect worker start -p "local-pool"
```

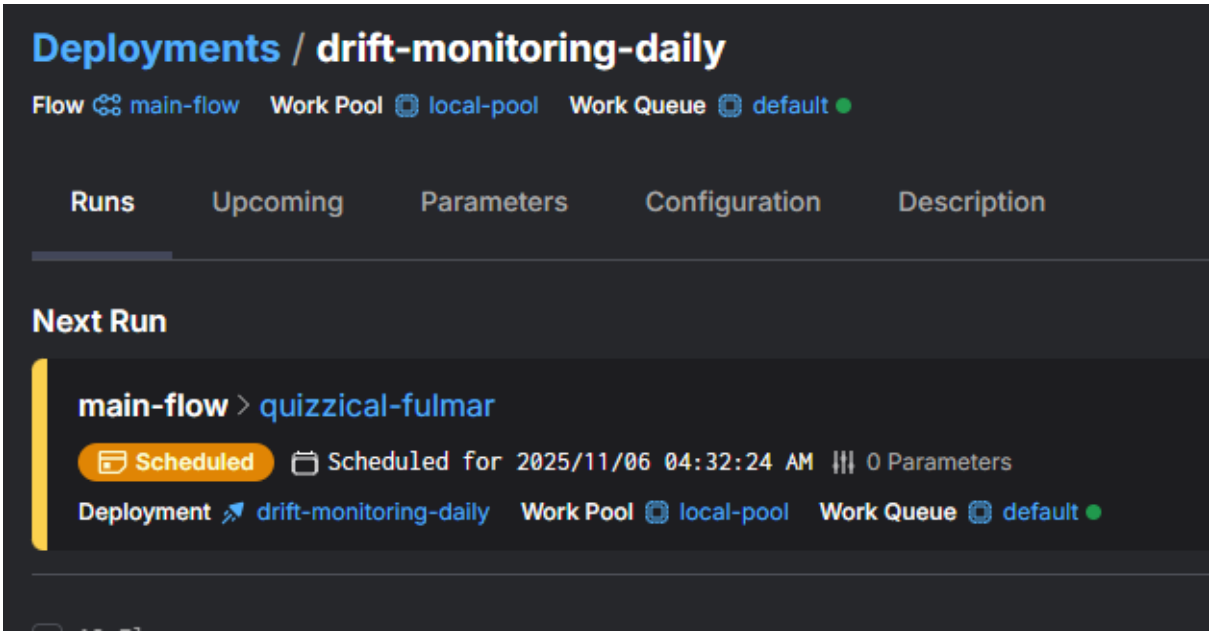
```
(.venv) PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> prefect deploy orchestrate_monitoring2.py:main_flow -n "drift-monitoring-daily"
Unable to read the specified config file. Reason: [Errno 2] No such file or directory: 'prefect.yaml'. Skipping.
22:30:57.533 | INFO    | prefect - Starting temporary server on http://127.0.0.1:8920
See https://docs.prefect.io/v3/concepts/server#how-to-guides for more information on running a dedicated Prefect server.
? Which work pool would you like to deploy this flow to? [Use arrows to move; enter to select]

|  | Work Pool Name | Infrastructure Type | Description |
|---|---|---|---|
| > | local-pool | process | None |

? Your Prefect workers will need access to this flow's code in order to run it. Would you like your workers to pull your flow code from a remote storage location when running this flow? [y/n] (y): n
Your Prefect workers will attempt to load your flow from:
C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow\orchestrate_monitoring2.py. To see more options for managing your flow's code, run:

        $ prefect init

? Would you like to configure schedules for this deployment? [y/n] (y): y
? What type of schedule would you like to use? [Use arrows to move; enter to select]

|  | Schedule Type | Description |
|---|---|---|
| > | Interval | Allows you to set flow runs to be executed at fixed time intervals. |
|  | Cron | Allows you to define recurring flow runs based on a specified pattern using cron syntax. |
|  | RRule | Allows you to define recurring flow runs using RFC 2445 recurrence rules. |

? Seconds between scheduled runs (3600): 1800
? Would you like to activate this schedule? [y/n] (y): y
? Would you like to add another schedule? [y/n] (n): n
Deployment 'main-flow/drift-monitoring-daily' successfully created with id '18ea7590-fe12-4cce-906b-992a90aa5579'.
? Would you like to save configuration for this deployment for faster deployments in the future? [y/n]: y
Deployment configuration saved to prefect.yaml! You can now deploy using this deployment configuration with:
```

```
Deployment 'main-flow/drift-monitoring-daily' successfully created with id '18ea7590-fe12-4cce-906b-992a90aa5579'.
? Would you like to save configuration for this deployment for faster deployments in the future? [y/n]: y
Deployment configuration saved to prefect.yaml! You can now deploy using this deployment configuration with:

        $ prefect deploy -n drift-monitoring-daily

You can also make changes to this deployment configuration by making changes to the YAML file.

To execute flow runs from these deployments, start a worker in a separate terminal that pulls work from the None work pool:

        $ prefect worker start --pool None

To schedule a run for this deployment, use the following command:

        $ prefect deployment run 'main-flow/drift-monitoring-daily'

22:32:40.333 | INFO    | prefect - Stopping temporary server on http://127.0.0.1:8920
(.venv) PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow>
```

Na interface fica em “Deployments”

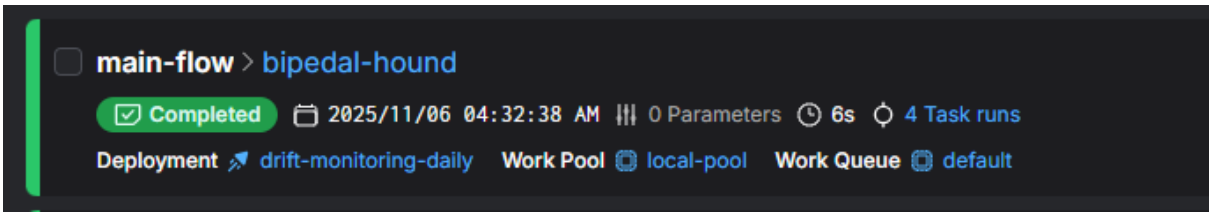


Clicando você consegue acompanhar



quando chega a hora ele executa:

```
04:32:25.288 | INFO    | prefect.flow_runs.worker - Worker 'ProcessWorker 7fa36fb5-1e7e-4512-9a4f-167f8374b418' submitting flow run '019a58
0c-477a-7a1c-91c6-0abd6d58e3a9'
04:32:25.783 | INFO    | prefect.flow_runs.runner - Opening process...
04:32:25.855 | INFO    | prefect.flow_runs.worker - Completed submission of flow run '019a580c-477a-7a1c-91c6-0abd6d58e3a9'
04:32:31.869 | INFO    | Flow run 'bipedal-hound' -  > Running set_working_directory step...
04:32:38.839 | INFO    | Flow run 'bipedal-hound' - Beginning flow run 'bipedal-hound' for flow 'main-flow'
04:32:38.842 | INFO    | Flow run 'bipedal-hound' - View at http://127.0.0.1:4200/runs/flow-run/019a580c-477a-7a1c-91c6-0abd6d58e3a9
2025/11/06 04:32:39 WARNING mlflow.utils.autologging_utils: MLflow sklearn autologging is known to be compatible with 1.3.1 <= scikit-learn
 <= 1.7.1, but the installed version is 1.7.2. If you encounter errors during autologging, try upgrading / downgrading scikit-learn to a co
mpatible version, or try upgrading MLflow.
2025/11/06 04:32:40 INFO mlflow.tracking.fluent: Autologging successfully enabled for sklearn.
2025/11/06 04:32:40 WARNING mlflow.utils.autologging_utils: MLflow xgboost autologging is known to be compatible with 2.0.0 <= xgboost <= 3
.0.2, but the installed version is 3.1.1. If you encounter errors during autologging, try upgrading / downgrading xgboost to a compatible v
ersion, or try upgrading MLflow.
2025/11/06 04:32:40 INFO mlflow.tracking.fluent: Autologging successfully enabled for xgboost.
04:32:40.817 | INFO    | Task run 'read_data-c28' - Finished in state Completed()
04:32:41.187 | INFO    | Task run 'read_data-05e' - Finished in state Completed()
04:32:42.391 | INFO    | Task run 'add_features-728' - Finished in state Completed()
04:32:44.600 | INFO    | Task run 'detect_drift-92f' - Finished in state Completed()
04:32:44.723 | INFO    | Flow run 'bipedal-hound' - Finished in state Completed()
Drift KS statistic: 0.030063252861911804, p-value: 5.006175129521501e-23
Model needs retraining: False
No retraining required at this time.
04:32:46.496 | INFO    | prefect.flow_runs.runner - Process for flow run 'bipedal-hound' exited cleanly.
```



Implementando métrica PADD

Para mudar a métrica de detecção de drift, eu apenas adicionei o script já disponível no github https://github.com/w4k2/padd

padd_methods.py:

```
import numpy as np
from scipy.stats import ttest_ind

def relu(x):
    return x * (x > 0)

class PADD:
    def __init__(self, alpha=0.05, ensemble_size=30, n_replications=35,
stat_proba=75, neck_width=10, th=0.17):
        self.alpha = alpha
        self.ensemble_size = ensemble_size
        self.n_replications = n_replications
        self.stat_proba = stat_proba
        self.neck_width = neck_width
        self.th = th
        self.past_probas = [[] for _ in range(self.ensemble_size)]
        self._is_drift = None

    def process(self, X: np.ndarray):
        if self._is_drift is None:
```

```
            self._is_drift = False
            self.n_features = X.shape[1]
            self.stack = [
                np.random.normal(0, 0.1, (self.n_features + 1,
self.neck_width)),
                np.random.normal(0, 0.1, (self.neck_width,
self.ensemble_size)),
            ]
        self.current_probas = self._predict_proba(X)

        if len(self.past_probas[0]) > 0:
            indications = np.zeros((self.ensemble_size,
self.n_replications))
            for member_id, (_past, current) in
enumerate(zip(self.past_probas, self.current_probas.T)):
                past = np.concatenate(_past)
                for rid in range(self.n_replications):
                    a = np.random.choice(past, self.stat_proba)
                    b = np.random.choice(current, self.stat_proba)
                    _, pval = ttest_ind(a, b)
                    indications[member_id, rid] = pval < self.alpha

            th = self.th * self.ensemble_size * self.n_replications
            if np.sum(indications) > th:
                self._is_drift = True
                self.past_probas = [[] for _ in
range(self.ensemble_size)]
            else:
                self._is_drift = False

        for member_id, probas in enumerate(self.current_probas.T):
            self.past_probas[member_id].append(probas)

    def _predict_proba(self, X: np.ndarray):
```

```
        val = np.concatenate((np.copy(X), np.ones((X.shape[0], 1))),
axis=1)
        for layer in self.stack:
            val = relu(val @ layer)
        predict_proba = np.exp(val - np.max(val, axis=1)[:, None])
        predict_proba = predict_proba / np.sum(predict_proba,
axis=1)[:, None]
        return predict_proba
```

Em drift-prefect-mlflow\orchestrate_monitoring2.py eu adicionei essa task:

```
@task
def detect_drift_padd(X_train, X_val, padd_params: dict | None = None)
-> bool:
    if padd_params is None:
        padd_params = dict(alpha=0.05, ensemble_size=30,
n_replications=35, stat_proba=75, neck_width=10, th=0.17)
    Xtr = X_train.toarray().astype(np.float32) if hasattr(X_train,
"toarray") else np.asarray(X_train, dtype=np.float32)
    Xva = X_val.toarray().astype(np.float32)   if hasattr(X_val,
"toarray") else np.asarray(X_val, dtype=np.float32)
    padd = PADD(**padd_params)
    padd.process(Xtr)
    padd.process(Xva)
    is_drift = bool(padd._is_drift)
    mlflow.log_params({f"padd_{k}": v for k, v in padd_params.items()})
    mlflow.log_metric("padd_is_drift", int(is_drift))

    return is_drift
```

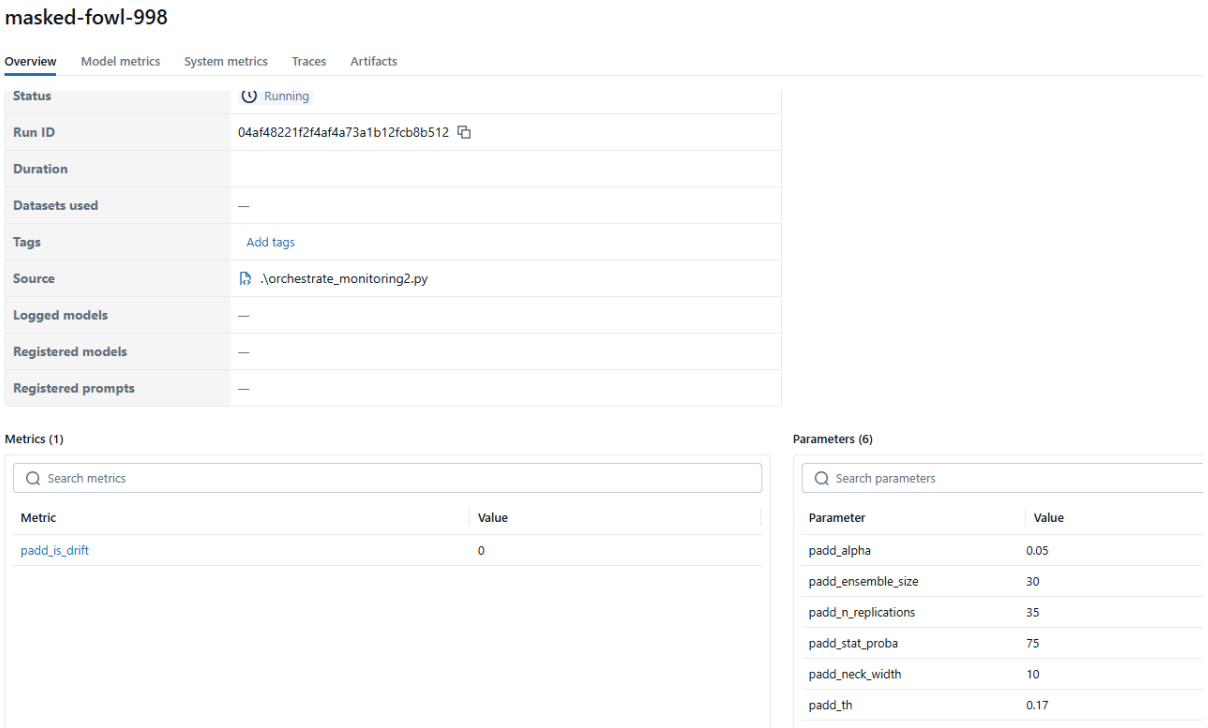
E no main_flow chamei ela:

```
needs_retraining = detect_drift_padd.submit(X_train, X_val).result()
```

Com isso foi só chamar como eu fazia antes

```
(.venv) PS C:\Users\Danielle\Desktop\Residencia\drift-prefect-mlflow> python .\orchestrate_monitoring2.py
05:41:48.043 | INFO    | Flow run 'competent-mole' - Beginning flow run 'competent-mole' for flow 'main-flow'
05:41:48.049 | INFO    | Flow run 'competent-mole' - View at http://127.0.0.1:4200/runs/flow-run/bd61d707-822e-4624-aafb-224eec5ee949
2025/11/06 05:41:48 WARNING mlflow.utils.autologging_utils: MLflow sklearn autologging is known to be compatible with 1.3.1 <= scikit-learn
 <= 1.7.1, but the installed version is 1.7.2. If you encounter errors during autologging, try upgrading / downgrading scikit-learn to a co
mpatible version, or try upgrading MLflow.
2025/11/06 05:41:48 INFO mlflow.tracking.fluent: Autologging successfully enabled for sklearn.
2025/11/06 05:41:48 WARNING mlflow.utils.autologging_utils: MLflow xgboost autologging is known to be compatible with 2.0.0 <= xgboost <= 3
.0.2, but the installed version is 3.1.1. If you encounter errors during autologging, try upgrading / downgrading xgboost to a compatible v
ersion, or try upgrading MLflow.
2025/11/06 05:41:48 INFO mlflow.tracking.fluent: Autologging successfully enabled for xgboost.
2025/11/06 05:41:48 WARNING mlflow.utils.autologging_utils: MLflow sklearn autologging is known to be compatible with 1.3.1 <= scikit-learn
 <= 1.7.1, but the installed version is 1.7.2. If you encounter errors during autologging, try upgrading / downgrading scikit-learn to a co
mpatible version, or try upgrading MLflow.
2025/11/06 05:41:48 INFO mlflow.tracking.fluent: Autologging successfully enabled for sklearn.
2025/11/06 05:41:48 WARNING mlflow.utils.autologging_utils: MLflow xgboost autologging is known to be compatible with 2.0.0 <= xgboost <= 3
.0.2, but the installed version is 3.1.1. If you encounter errors during autologging, try upgrading / downgrading xgboost to a compatible v
 <= 1.7.1, but the installed version is 1.7.2. If you encounter errors during autologging, try upgrading / downgrading scikit-learn to a co
mpatible version, or try upgrading MLflow.
2025/11/06 05:41:48 INFO mlflow.tracking.fluent: Autologging successfully enabled for sklearn.
2025/11/06 05:41:48 WARNING mlflow.utils.autologging_utils: MLflow xgboost autologging is known to be compatible with 2.0.0 <= xgboost <= 3
.0.2, but the installed version is 3.1.1. If you encounter errors during autologging, try upgrading / downgrading xgboost to a compatible v
2025/11/06 05:41:48 INFO mlflow.tracking.fluent: Autologging successfully enabled for sklearn.
2025/11/06 05:41:48 WARNING mlflow.utils.autologging_utils: MLflow xgboost autologging is known to be compatible with 2.0.0 <= xgboost <= 3
.0.2, but the installed version is 3.1.1. If you encounter errors during autologging, try upgrading / downgrading xgboost to a compatible v
.0.2, but the installed version is 3.1.1. If you encounter errors during autologging, try upgrading / downgrading xgboost to a compatible v
ersion, or try upgrading MLflow.
2025/11/06 05:41:48 INFO mlflow.tracking.fluent: Autologging successfully enabled for xgboost.
05:41:48.908 | INFO    | Task run 'read_data-466' - Finished in state Completed()
05:41:49.175 | INFO    | Task run 'read_data-9ba' - Finished in state Completed()
05:41:49.791 | INFO    | Task run 'add_features-e85' - Finished in state Completed()
05:42:11.555 | INFO    | Task run 'detect_drift_padd-628' - Finished in state Completed()
No retraining required at this time.
05:42:11.946 | INFO    | Flow run 'competent-mole' - Finished in state Completed()
```

No MLflow:



APÊNDICE 6

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 12 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Danielle Tavares da Silva

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta semana, analisei se diferentes **métricas de detecção de drift** influenciam na identificação das mudanças nos dados, tanto em relação à ocorrência de **falsos positivos**, quanto ao **atraso** na detecção.

Primeiro, testei as métricas utilizando **dados sintéticos do repositório original do PADD**. Em seguida, utilizei **dados sintéticos do repositório DriftDatasets** (*Rotating Hyperplane* e *Moving Squares*), pois eles possuem ground truth de drift, o que me permitiu calcular o atraso exato entre o momento da mudança e a detecção.

Nos experimentos, comparei as métricas **KS (Kolmogorov–Smirnov Test), PSI (Population Stability Index), ADWIN (Adaptive Windowing), PADD (Probabilistic Adaptive Drift Detector), MD3 (Margin Density Drift Detector) e EDDM (Early Drift Detection Method)**, avaliando seu comportamento em dois tipos de drift:

- **Gradual**, quando as mudanças nos dados acontecem de forma lenta e contínua.
- **Abrupto**, quando há uma transição súbita e rápida na distribuição.

A partir dos resultados, percebi que o **MD3** e o **ADWIN** apresentaram o melhor desempenho em situações de **drift gradual**, pois conseguiram detectar as mudanças com pouco atraso e sem gerar muitos falsos positivos. Já o **PSI** e o **EDDM** tiveram um bom desempenho em **drifts abruptos**, conseguindo detectar rapidamente, mas sendo mais sensíveis e apresentando muitos falsos positivos. O **PADD** mostrou boa capacidade de antecipar as mudanças, porém com bastante ruído, enquanto o **KS** não apresentou bons resultados.

Por fim, consolidei esses resultados no documento:
Experimentos de Detecção de Drift

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Organizar todas as entregas para a transição para o TCC

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Documentação – Experimentos de Detecção de Drift

O objetivo deste trabalho foi comparar diferentes métricas e métodos de detecção de drift em fluxos de dados (data streams), avaliando como cada detector reage a mudanças graduais e abruptas na distribuição dos dados.

Inicialmente, a ideia foi inspirada no repositório original do PADD (Probabilistic Adaptive Drift Detector), mas ao longo do processo evoluímos a abordagem para usar também datasets do repositório DriftDatasets, que contém fluxos de dados sintéticos com drifts conhecidos e realistas, como *Rotating Hyperplane* e *Moving Squares*.

O que é drift

Em aprendizado de máquina, **drift** ocorre quando a distribuição dos dados muda ao longo do tempo, o que faz com que modelos previamente treinados passem a ter pior desempenho.
Essas mudanças podem ser:

- **Abrupto:** mudança súbita.
- **Gradual:** transição lenta/suave.
- **Recorrente:** volta após um tempo.
- **Incremental:** mudança contínua.

Objetivo da detecção de drift

Os detectores buscam identificar quando ocorre uma mudança significativa na distribuição para que o modelo possa ser reentreinado ou atualizado.

Métricas utilizadas

Durante os experimentos, testei e comparei diferentes métricas de detecção:

- **KS (Kolmogorov–Smirnov Test):** compara duas distribuições amostrais para ver se são diferentes.
- **PSI (Population Stability Index):** mede o quanto a distribuição atual se desviou da distribuição de referência.
- **ADWIN (Adaptive Windowing):** usa janelas deslizantes que se expandem ou contraem conforme o erro muda.
- **PADD (Probabilistic Adaptive Drift Detector):** detector proposto no repositório original, baseado em ensembles probabilísticos.
- **MD3 (Margin Density Drift Detector):** mede mudanças na densidade dos dados próximos à fronteira de decisão do modelo.
- **EDDM (Early Drift Detection Method):** detector supervisionado baseado em distâncias entre erros consecutivos do modelo.

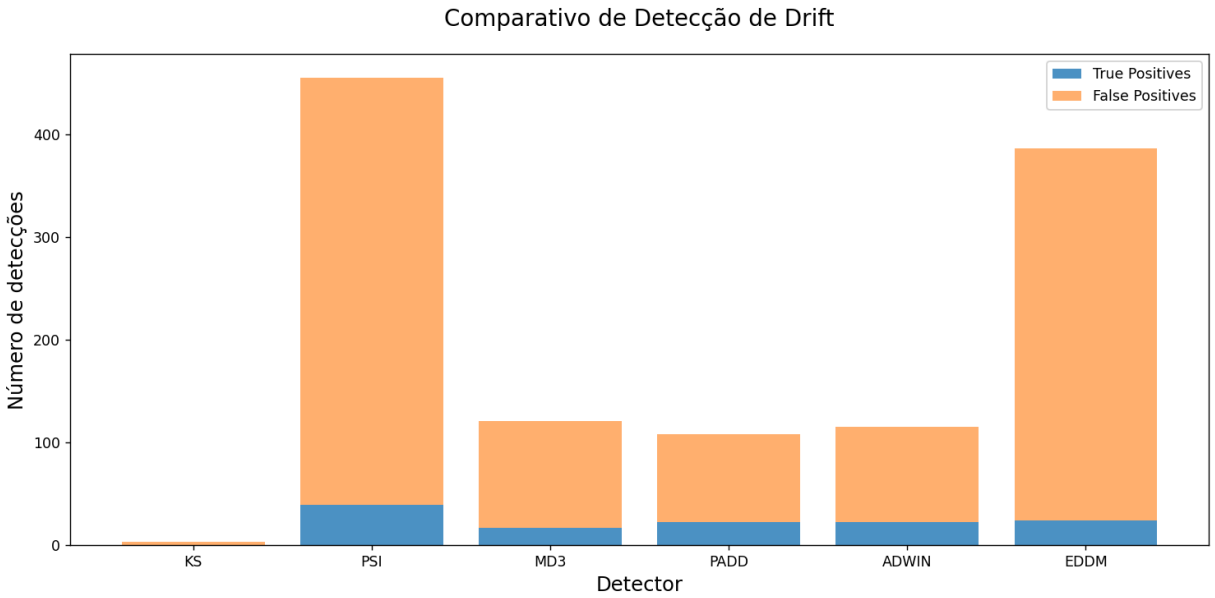
Primeira fase – Utilizado o Repositório PADD

Começamos com o código do repositório **PADD (https://github.com/w4k2/padd)**. Esse código já trazia uma estrutura pronta para comparar detectores em streams sintéticas com múltiplos drifts, usando funções de geração de dados internas

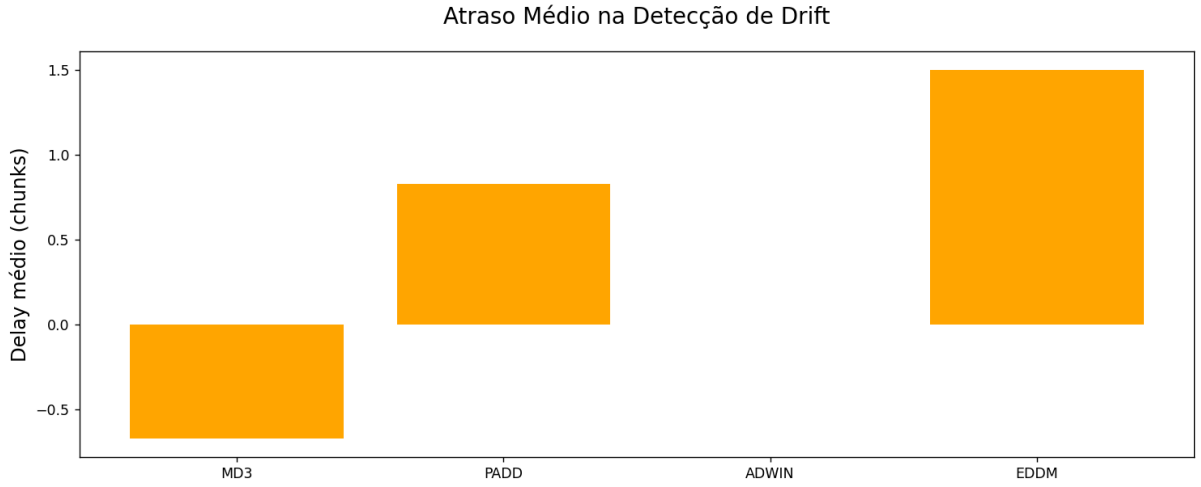
Resultados iniciais

Resultados do drift abrupto

- PSI e EDDM: detectam cedo (atraso baixo/negativo), porém com muitos falsos positivos.
- PADD e MD3: equilíbrio entre detecção e ruído; costumam acertar perto do ponto de mudança com FP moderado.
- ADWIN: conservador, normalmente confirma após a mudança (atraso pequeno/positivo), FP contido.
- KS: quase não ativa; só reage quando a diferença estatística é muito marcada.



Abrupto: barras TP/FP

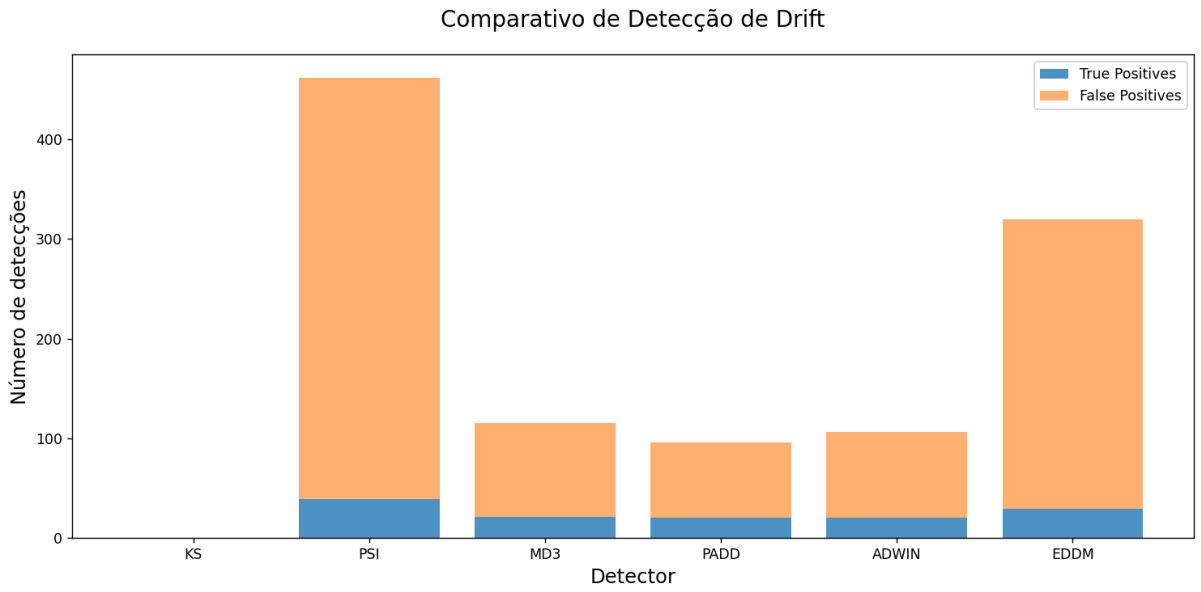


Abrupto: delay médio

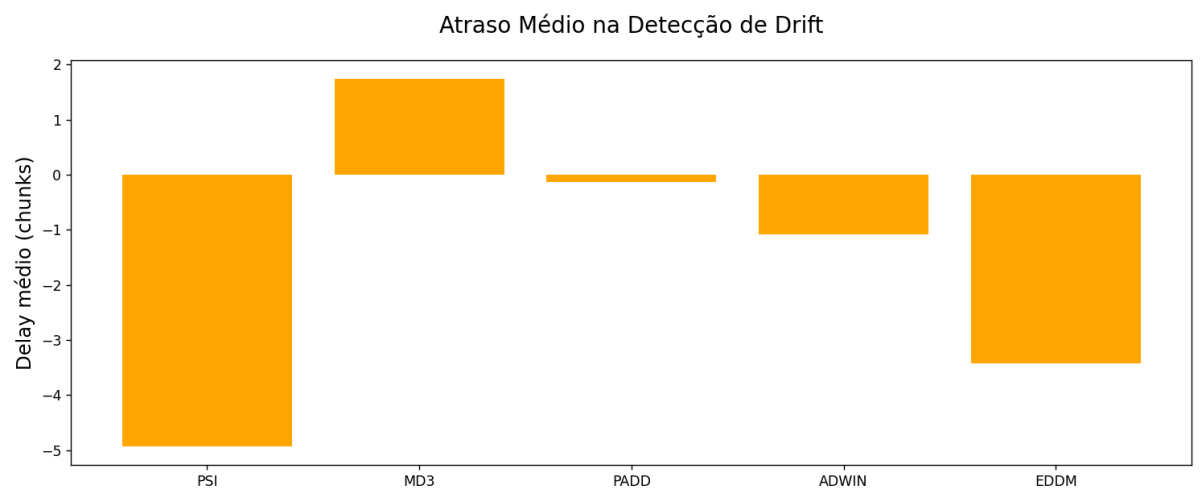
Resultados do drift gradual

- PSI e EDDM: seguem muito sensíveis, sinalizam cedo, mas aumentam FP ao longo da rampa.
- MD3: consistente; acompanha a transição com atraso pequeno/médio e FP controlado.
- PADD: estável; atraso pequeno a moderado, FP menor que PSI/EDDM.

- ADWIN: bom para confirmar drift gradual (tende a detectar depois, com menos ruído).
- KS: quase não aciona em drift gradual; a diferença entre janelas muda devagar.



Gradual: barras TP/FP



Gradual: delay médio

Segunda fase – Repositório DriftDatasets

Após testar com dados sintéticos gerados manualmente, passamos a usar datasets prontos do repositório **DriftDatasets (https://github.com/vlosing/driftDatasets)**.
Esses datasets já foram amplamente usados em trabalhos acadêmicos para avaliar detecção de drift.

Datasets utilizados

1. **Rotating Hyperplane:**
    - Simula drifts **graduais**, onde o hiperplano de separação entre classes rotaciona ao longo do tempo.
    - É útil para ver como detectores reagem a mudanças lentas e contínuas.
2. **Moving Squares:**
    - Simula drifts **abruptos**, com mudanças súbitas na posição dos quadrados.
    - Ideal para testar como os detectores respondem a transições rápidas na distribuição.

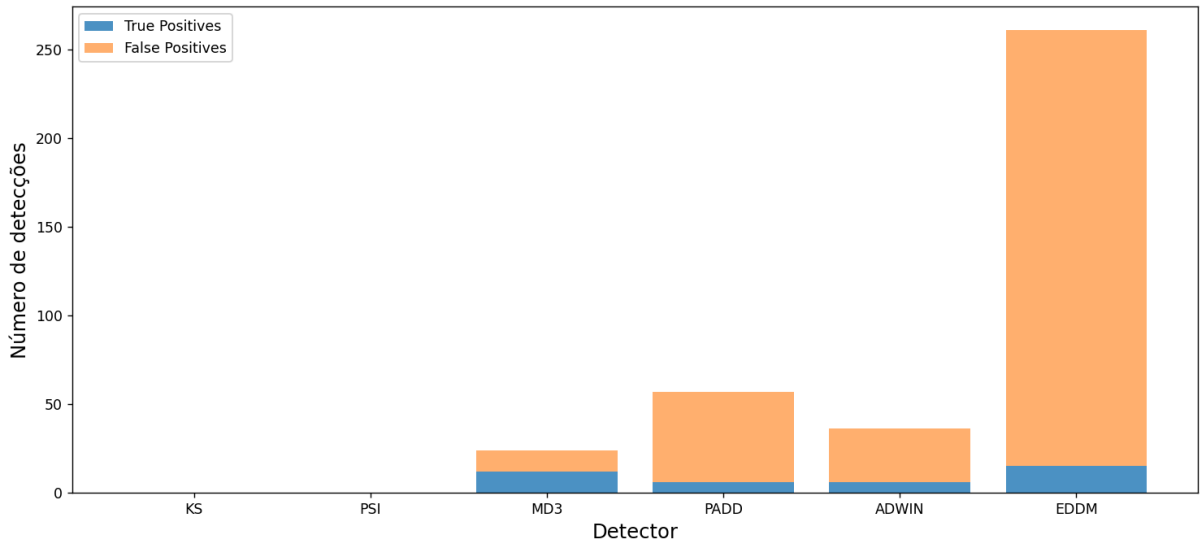
Resultados com DriftDatasets

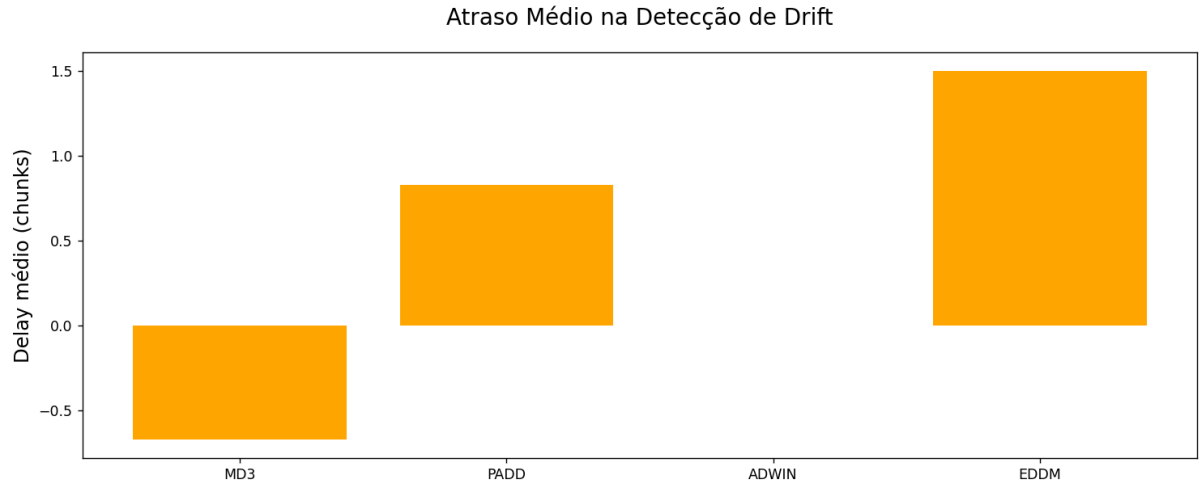
Rotating Hyperplane (drift gradual)

Esse experimento mostrou que:

- MD3 é o melhor trade-off (acerta cedo e com pouco FP).
- ADWIN serve bem como confirmação (atraso ~0 e FP médio).

- PADD ficou mais ruidoso que ADWIN neste dataset (FP bem maior e atraso positivo).
- KS/PSI não capturaram esse tipo de drift.
- EDDM aciona demais (muitos FP), apesar do TP maior.

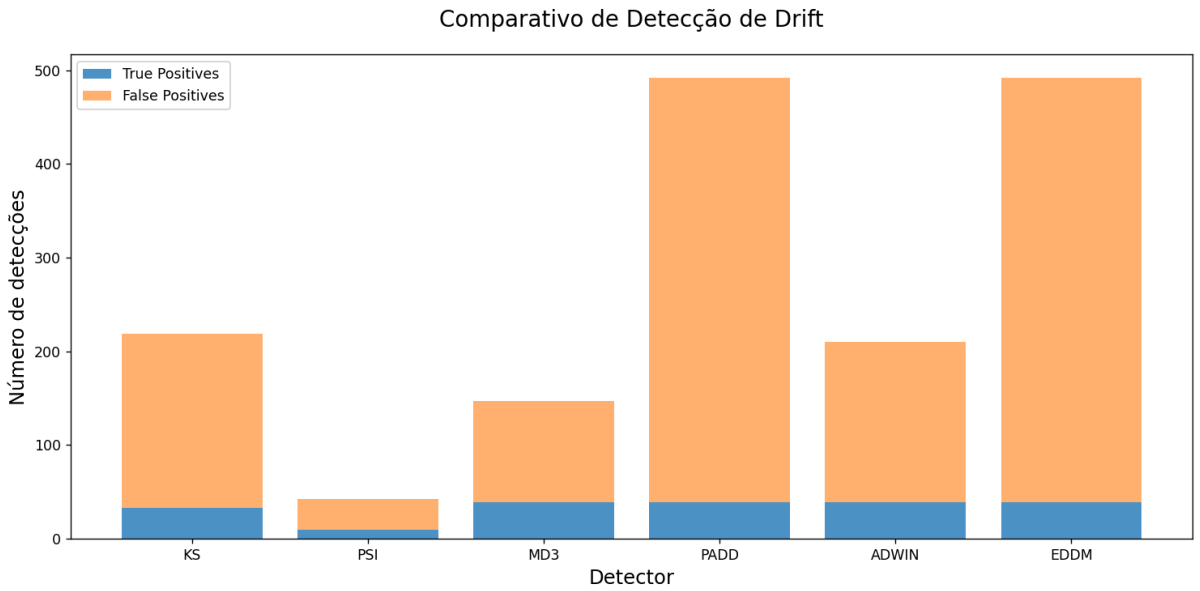


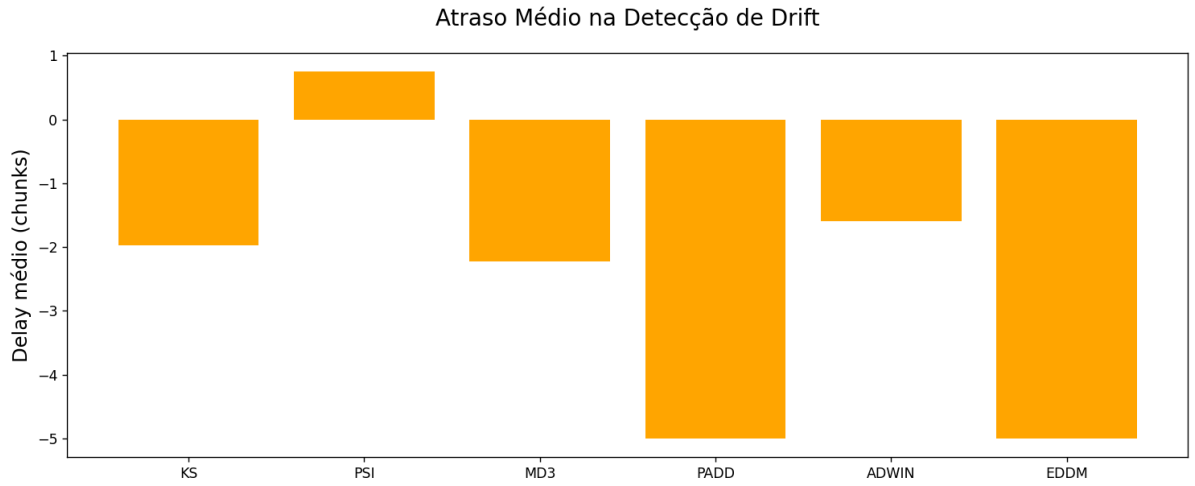


Moving Squares (drift abrupto)

Com esse dataset abrupto:

- MD3: melhor equilíbrio, antecipa e mantém FP moderado.
- ADWIN: bom confirmador, perto do ponto, FP intermediário.
- PSI: mais limpo, poucos FP, detectou logo depois.
- PADD: muito precoce, porém muito FP;
- KS: detecta, mas gera FP alto;
- EDDM: hipersensível, TP alto, FP altíssimo.


Comparativo de Detecção de Drift
True Positives
False Positives
Número de detecções
0
100
200
300
400
500
KS
PSI
MD3
PADD
ADWIN
EDDM
Detector


Atraso Médio na Detecção de Drift
Delay médio (chunks)
1
0
−1
−2
−3
−4
−5
KS
PSI
MD3
PADD
ADWIN
EDDM