Relatório Técnico: Projeto Vendas do Produto Alfa

Objetivo:

O projeto proposto tem como objetivo central construir uma solução de previsão de demanda para o Produto Alfa, item de maior relevância comercial dentro da empresa Logística Eficiente S.A.

Desafio:

O desafio enfrentado pelo time de operações consiste em manter o equilíbrio entre dois polos críticos da gestão de estoque: um estoque excessivo, que eleva significativamente os custos de armazenamento, enquanto, por outro lado, lida-se com a falta de produtos, resultando em perda direta de vendas e em insatisfação dos clientes. A fim de apoiar decisões estratégicas e operacionais, a meta é prever com confiabilidade as vendas diárias do Produto Alfa para um período de **14 dias.**

A entrega recebida, materializada no repositório **Vendas-Produto-Alfa**, constitui uma estrutura que organiza os elementos de um pipeline completo de ciência de dados, embora diversos módulos ainda estejam representados como placeholders, visto que estamos lidando com dados puramente fictícios.

Estrutura:

O repositório inclui uma estrutura de diretórios, com separação entre notebooks, scripts, código-fonte em src/, além de arquivos de apoio como requirements.txt, Dockerfile e diagramas visuais (.drawio) que descrevem a arquitetura e os fluxos planejados.

Essa organização demonstra a intenção de modularizar o processo em etapas distintas, sendo elas: ingestão de dados, engenharia de atributos, treinamento, predição, deploy e monitoramento. Além disso,também deixa evidente que se trata de uma base conceitual e não de uma implementação finalizada.

O arquivo de dados reais não foi incluído, o que é coerente com as instruções do desafio, mas reforça que os módulos existentes precisariam ser preenchidos com implementações concretas para permitir a execução real.

Arquitetura em nuvem:

Para evoluir essa base até um ambiente de produção em nuvem, é necessário estruturar a arquitetura com componentes modernos e consolidados de MLOps. A proposta delineada considera os princípios de modularidade, observabilidade, reprodutibilidade e segurança. A camada de armazenamento pode ser implementada em serviços como Amazon S3, Google

Cloud Storage ou Azure Blob Storage, organizando os dados em zonas lógicas como raw, processed, models e forecasts. A orquestração dos fluxos de ingestão, pré-processamento, treinamento e geração de previsões pode ser realizada com Airflow ou Prefect, enquanto o processamento é delegado a containers em serviços como ECS, EKS ou Cloud Run. O treinamento de modelos deve ser orquestrado em ambientes especializados como SageMaker ou Vertex AI, que permitem tanto execução isolada quanto otimização de hiperparâmetros em escala. O versionamento de modelos e experimentos é melhor tratado com MLflow ou registries nativos das nuvens, garantindo governança e rastreabilidade. Para disponibilizar as previsões, um serviço FastAPI containerizado pode ser colocado em produção em plataformas serverless como Cloud Run, ECS Fargate ou mesmo em clusters Kubernetes, oferecendo tanto endpoints online de baixa latência quanto previsões em lote agendadas.

Pipeline dos dados:

O pipeline de dados tem início com a ingestão de arquivos CSV diários, exportados do ERP ou recebidos de fluxos de eventos. Esses arquivos são armazenados em buckets organizados por data e passam por verificações de qualidade, que incluem a checagem do schema, a validação dos tipos de dados e a detecção de possíveis duplicatas, com alertas automáticos em caso de inconsistências.

Na etapa seguinte, ocorre o pré-processamento: as datas são padronizadas, as variáveis de feriado e promoção são convertidas em indicadores binários, atributos temporais como finais de semana e início ou fim de mês são gerados, e valores atípicos são tratados. Em seguida, aplica-se a engenharia de atributos, que introduz defasagens de diferentes horizontes, médias móveis, desvios padrão e interações entre variáveis relevantes, como promoções e finais de semana.

Para capturar sazonalidades semanais e anuais, utiliza-se encoding adequado das datas.

Por fim, são construídos os conjuntos de treino e validação, geralmente reservando as duas últimas semanas para teste e adotando estratégias de validação do tipo walk-forward, que simulam o comportamento da operação em tempo real.

Modelagem:

A estratégia proposta envolve a combinação de um modelo estatístico como Prophet, voltado para capturar a sazonalidade e efeitos de feriados de forma natural, com algoritmos de machine learning como LightGBM, que aproveitam atributos de lag e interações. A abordagem híbrida, em que previsões estatísticas servem como features adicionais para o modelo de machine learning, pode trazer ganhos de acurácia. Como métricas principais, recomenda-se priorizar o MAE (erro absoluto médio), por estar diretamente ligado ao impacto de excesso ou falta de estoque, além de utilizar RMSE, MAPE e métricas de cobertura quando intervalos de confiança forem incluídos.

O treinamento na nuvem deve ser acionado por jobs agendados, executando o código de treino, registrando artefatos como modelos, métricas e metadados, e promovendo versões aprovadas para o ambiente de staging antes de irem para produção. O deploy pode ocorrer tanto em modo online, por meio de APIs que respondem requisições de previsão, quanto em modo batch, gerando arquivos com previsões que alimentam diretamente os sistemas de ERP e BI da empresa. O monitoramento precisa abranger métricas de negócio, como o erro preditivo em janelas móveis, métricas operacionais como latência e throughput dos serviços, além de monitoramento de drift de dados e de conceito, utilizando ferramentas como Evidently. Alertas devem ser disparados quando desvios significativos ocorrerem, permitindo ações rápidas como re-treinamento emergencial.

CI/CD:

Um aspecto central do projeto é a adoção de práticas de CI/CD e infraestrutura como código. Isso inclui pipelines de integração contínua no GitHub Actions para realizar testes, build e deploy automatizados, além do uso de Terraform para provisionar recursos de nuvem de forma replicável e segura. Segregação de ambientes de desenvolvimento, staging e produção garante maior controle, enquanto o uso de secrets managers e políticas de menor privilégio fortalecem a segurança operacional.

Considerações:

Para que o repositório atual se torne efetivamente operável, é essencial preencher o requirements.txt com dependências fixas, implementar as funções de ingestão e engenharia de atributos em src/data.py e src/features.py, completar os scripts de treinamento e previsão em src/models/train.py e src/models/predict.py, criar um serviço FastAPI em app/main.py com endpoints de health check e previsão, além de enriquecer o notebook de EDA com análises gráficas, mesmo que baseadas em dados sintéticos. A criação de templates de pipelines em Airflow e testes automatizados em CI também deve ser priorizada.

Passos seguintes:

Em termos de entregas e prazos, um MVP poderia ser construído em até sete dias, focando em completar os módulos essenciais e disponibilizar previsões simuladas com dados sintéticos. Em um horizonte de três a seis semanas, a solução poderia ser expandida para incluir monitoramento avançado, integração com registries de modelos, provisionamento automático via Terraform e dashboards operacionais em Grafana ou Streamlit.

Conclusão:

Por fim, o projeto é composto com um esqueleto sólido a fim de modular previsões para que o sistema de previsão de demanda funcione e possa apoiar a Logística Eficiente S.A na redução de custos do armazenamento e na prevenção de perdas de venda.