Roadmap Avançado – Próximos Passos do Projeto TrustShield

Com o pipeline de deteção de anomalias validado e o modelo Isolation Forest selecionado como campeão, o projeto TrustShield entra agora em duas novas frentes de trabalho que irão elevar o sistema de um motor de predição para uma plataforma completa de inteligência contra fraudes.

Vertente 1: Previsão de Tendências de Fraude (Séries Temporais)

Conforme o seu plano de projeto, o objetivo é "Prever volume e frequência de fraudes até 2040". Esta é uma tarefa de **análise de séries temporais**.

Plano de Ação:

1. **Agregação de Dados:** O primeiro passo é transformar os nossos dados transacionais numa série temporal. Iremos criar um script que agrega o número de anomalias detectadas (e o seu valor monetário) por unidade de tempo (ex: por dia, semana ou mês).
2. **Engenharia de Features para Séries Temporais:** Criaremos features específicas para este tipo de modelo, como médias móveis, lags (valores de períodos anteriores) e indicadores sazonais (ex: mês do ano, dia da semana).
3. **Modelagem com Prophet ou ARIMA:** Conforme o seu plano, utilizaremos modelos especializados como o **Prophet** (desenvolvido pelo Facebook, excelente para dados com sazonalidade) ou o clássico **ARIMA**.
4. **Criação de um Novo Script de Treino (src/models/train_ts_model.py):** Para manter a modularidade, todo este processo será encapsulado num novo script, seguindo a estrutura do seu train_fraud_model.py. Este script irá gerar um artefato de modelo de série temporal (ex: prophet_model_v1.joblib).
5. **Previsão e Visualização:** O modelo treinado será então usado para gerar as previsões até 2040, e os resultados serão visualizados em gráficos de tendência no nosso dashboard.

Vertente 2: Governança e MLOps com MLflow

Para alcançar a "perfeição extrema na engenharia", precisamos de ferramentas que nos permitam gerir o ciclo de vida dos nossos modelos de forma sistemática. O **MLflow** é a escolha ideal para isso e integra-se perfeitamente no nosso pipeline existente.

Plano de Ação:

1. **Rastreamento de Experimentos (Experiment Tracking):**
    a. **Ação:** Iremos integrar o MLflow diretamente no nosso script train_fraud_model.py.
    b. **Benefício:** Com apenas algumas linhas de código, cada execução do script (python train_... --model lof) será registada como um "experimento". O MLflow irá guardar automaticamente:
        i. Os **parâmetros** usados (do config.yaml).
        ii. As **métricas** de avaliação (taxa de anomalias, tempo de inferência).
        iii. O **artefacto do modelo** (.joblib) e outros ficheiros, como os gráficos de análise.

2. **Registo de Modelos (Model Registry):**
    a. **Ação:** Após treinar vários modelos, usaremos a interface do MLflow para comparar os seus resultados e promover o nosso modelo campeão (Isolation Forest) para um "Registo de Modelos".
    b. **Benefício:** Isto cria uma fonte única e versionada dos modelos que estão prontos para produção. O nosso script de inferência (predict.py) pode ser modificado para carregar a versão "em Produção" diretamente do registo, em vez de procurar o ficheiro mais recente numa pasta.
3. **Reprodutibilidade e Deploy:**
    a. **Ação:** O MLflow guarda o ambiente Conda e as dependências exatas de cada experimento.
    b. **Benefício:** Isto garante uma reprodutibilidade de 100% e simplifica drasticamente o processo de deploy, pois podemos empacotar o modelo e todo o seu ambiente de forma padronizada.

Recomendação Estratégica

A minha recomendação é começar pela **Vertente 2: Integração com MLflow**. Ao adicionar esta camada de governança ao pipeline que já construímos, estamos a solidificar o nosso trabalho, a facilitar a comparação de futuros experimentos (incluindo os de séries temporais) e a preparar o terreno para um deploy profissional.

Posso modificar o seu script train_fraud_model.py para incluir a integração completa com o MLflow, se desejar.