Faculdade XPe

Capítulo 07

MONITORAMENTO E MANUTENÇÃO DE MODELOS

Luciano Tadeu P.A.Pereira

2024

Neste capítulo veremos uma etapa muito importante no ciclo de vida de um algoritmo de Machine Learning que é o monitoramento e manutenção dele. Será apresentada a importância do monitoramento e manutenção, com as formas e detecção desses acontecimentos.

Importância do Monitoramento e Manutenção de Modelos

Quanto estamos falando em monitoramento de manutenção de um modelo de Machine Learning, estamos mencionando principalmente o aspecto de desempenho contínuo com sua precisão das previsões feitas ao longo do tempo. Modelos de Machine Learning podem sofrer de degradação de performance conforme o cenário no qual foram treinados evolui, ou novos dados começam a apresentar padrões diferentes daqueles previamente aprendidos. Sem o acompanhamento adequado, o modelo pode se tornar irrelevante ou apresentar vieses prejudiciais, levando a decisões incorretas que impactam negativamente os negócios.

Monitorar os modelos é fundamental para garantir que resultados das predições do algoritmo estejam confiáveis para tomada de decisão da equipe de negócio. Esse monitoramento, quando feito de forma automatizada, é possível detectar anomalias de formas rápida e precisa.

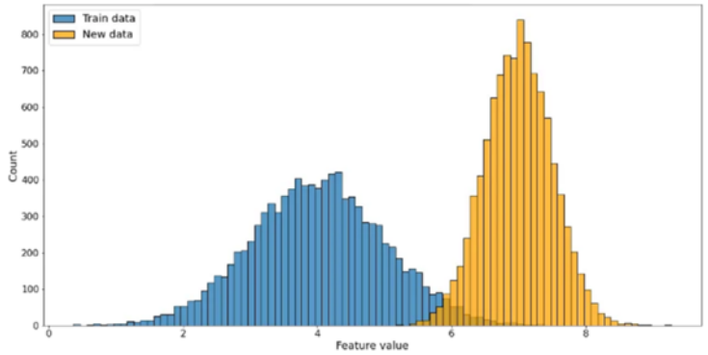
Além disso, o monitoramento permite que a equipe de ciência de dados ajuste os hiperparâmetros e técnicas utilizadas no treinamento inicial. O processo de manutenção regular também serve para validar se as suposições feitas durante a construção do modelo permanecem válidas com o tempo, evitando o envelhecimento prematuro do sistema de aprendizado.

Formas de Monitoramento de Modelos

Existem diversas formas de monitoramento de modelos de Machine Learning, cada uma focada em diferentes aspectos da performance. A monitoração de métricas é uma das abordagens mais comuns e envolve acompanhar indicadores como acurácia, precisão, recall, F1-score e outras métricas relevantes ao longo do tempo. Qualquer queda nesses valores pode ser um sinal de que o modelo não está mais performando adequadamente, exigindo análise detalhada.

Outra abordagem envolve o monitoramento de “data drift”, que é o fenômeno em que as distribuições de um conjunto de features sofre alteração com relação aos dados que foram treinados. Abaixo é apresentado uma imagem em que apresenta a alteração do comportamento de um feature após o ambiente de produção, mostrando o que é o data drif.

Figura 1 - Exemplo de indicativo de Data Drif



Fonte: Detecting Data Drift with Machine Learning | Blog post by Hennie de Harder | BigData Republic - BigData Republic.

Ferramentas de detecção de drift de forma automática são muito importantes para que os cientistas de dados tenham agilidade de identificar o que está acontecendo com seu algoritmo. Abaixo é apresentado um exemplo de pipeline para automatização desse processo.

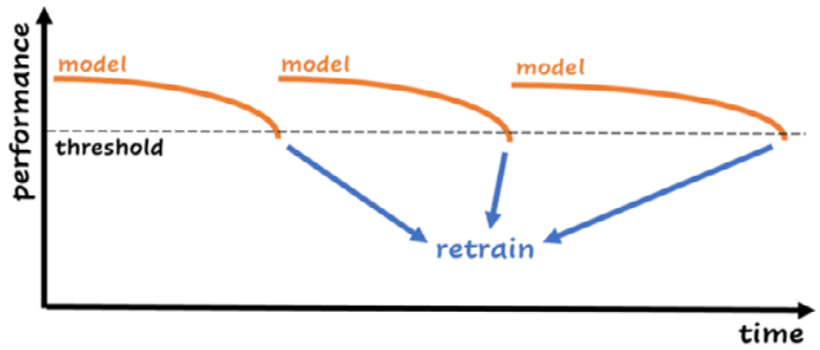
```
import pandas as pd
from sklearn.model_selection import train_test_split
from scipy import stats
import numpy as np

# Carregar o dataset Iris diretamente do Pandas
def load_iris_data():
    # O conjunto de dados Iris já está disponível no pandas
    iris_data = pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv")
    return iris_data
# Função para pré-processar os dados (nesse caso, não é necessário muito processamento)
def preprocess_data(df):
    df.fillna(0, inplace=True)  # Garantir que não há valores nulos
    return df
# Função para detectar data drift usando o teste de Kolmogorov-Smirnov
def detect_data_drift(train_data, prod_data, threshold=0.05):
    drift_report = {}

    for column in train_data.columns:
        if train_data[column].dtype in [np.float64, np.int64]:
            # Aplicar o teste de Kolmogorov-Smirnov para variáveis contínuas
            stat, p_value = stats.ks_2samp(train_data[column], prod_data[column])
            drift_report[column] = {
                'KS_statistic': stat,
                'p_value': p_value,
                'drift_detected': p_value < threshold  # Se o p-valor for menor que o threshold, houve drift
            }

    return drift_report
# Função para exibir o relatório de data drift
def display_drift_report(drift_report):
    print("\nRelatório de Data Drift:")
    for column, report in drift_report.items():
        print(f"Variável: {column}")
        print(f"  KS Estatística: {report['KS_statistic']:.4f}")
        print(f"  P-valor: {report['p_value']:.4f}")
        print(f"  Data Drift Detectado: {'Sim' if report['drift_detected'] else 'Não'}")
        print()
# Pipeline de detecção de data drift
def data_drift_pipeline(threshold=0.05):
    # Carregar o dataset Iris
    iris_data = load_iris_data()

    # Pré-processar os dados
    iris_data = preprocess_data(iris_data)

    # Dividir os dados entre "treinamento" e "produção" (aqui dividimos de forma aleatória)
    train_data, prod_data = train_test_split(iris_data, test_size=0.5, random_state=42)

    # Verificar se há data drift
    print("Verificando Data Drift...")
    drift_report = detect_data_drift(train_data, prod_data, threshold)

    # Exibir o relatório de data drift
    display_drift_report(drift_report)
# Executar o pipeline com o dataset Iris
data_drift_pipeline(threshold=0.05)
```

Retreinamento de Algoritmos

O retreinamento de algoritmos de Machine Learning é uma estratégia necessária para garantir que o modelo continue relevante e atualizado com novos dados. À medida que os dados de produção mudam ou crescem, o modelo deve ser reavaliado e, muitas vezes, retreinado para incorporar essas novas informações. Esse processo envolve usar um novo conjunto de dados, ou incluir dados mais recentes no treinamento, de modo a ajustar o modelo às novas realidades.

Um dos desafios do retreinamento é identificar o momento correto para realizá-lo. A degradação no desempenho ou a detecção de drift nos dados pode indicar a necessidade de retreinamento. Dependendo do setor ou da aplicação, o ciclo de retreinamento pode variar desde intervalos regulares (por exemplo, mensal ou trimestralmente) até ser acionado automaticamente quando o monitoramento indica mudanças significativas. A imagem abaixo mostra o ciclo de retreinamento de um algoritmo onde, após a quebra de performance, é realizado o retreinamento.

Figura 2 - Exemplo necessidade de retreinamento.



Fonte: Detecting Data Drift with Machine Learning | Blog post by Hennie de Harder | BigData Republic - BigData Republic.

O retreinamento também deve considerar questões como o “overfitting” e “underfitting”. Durante o processo, ajustes nos hiperparâmetros podem ser necessários para garantir que o novo modelo não se adapte apenas aos novos dados, mas também continue generalizando bem os dados antigos. Técnicas como validação cruzada e a escolha correta do conjunto de dados de treinamento são essenciais para um retreinamento eficaz.

Faculdade XPe

xpeducacao.com.br