Integrantes do Grupo:
Augusto Dalcin Peiter

Murillo Masson Guapo

Thales Stuczynski

○ Escolha do dataset e problema abordado.

https://www.kaggle.com/datasets/ziya07/customer-churn-prediction-dataset/data

O dataset escolhido trata de customer churn, uma das sugestões levantadas no enunciado do trabalho.

Esse dataset em específico contém dados de 1000 usuários, com as seguintes colunas.

Customer_ID: A unique identifier for each customer.

Age: Idade (ranging from 18 to 70 years).

Gender: Gênero (0 = Male, 1 = Female).

Monthly_Spending: Gasto mensal (de 50 até 500 dólares).

Subscription_Length: Anos de inscrição no serviço (ranging from 1 to 10 years).

Support_Interactions: Interações com serviço de suporte ao cliente (ranging from 0 to 5).

Churn: Variável indicando se consumidor cancelou serviço (1) ou permaneceu (0).

○ Metodologia e ferramentas utilizadas.

Utilizamos dois modelos, um baseado no algoritmo de RandomForest e outro utilizando XGBoost, rodamos múltiplas versões dos modelos guardando informações básicas de performance e alterando os parâmetros de inicialização em cada rodada. Todas versões são gravadas no mlflow para

que em seguida possamos selecionar o melhor modelo para colocar em Produção, isso é feito baseado no seu valor de “accuracy”.

O ambiente inteiro é instanciado localmente, usamos FastAPI para possibilitar chamadas de API ao modelo servido, tendo disponíveis as funções “predict”, “predict_csv” e “check_drift”.

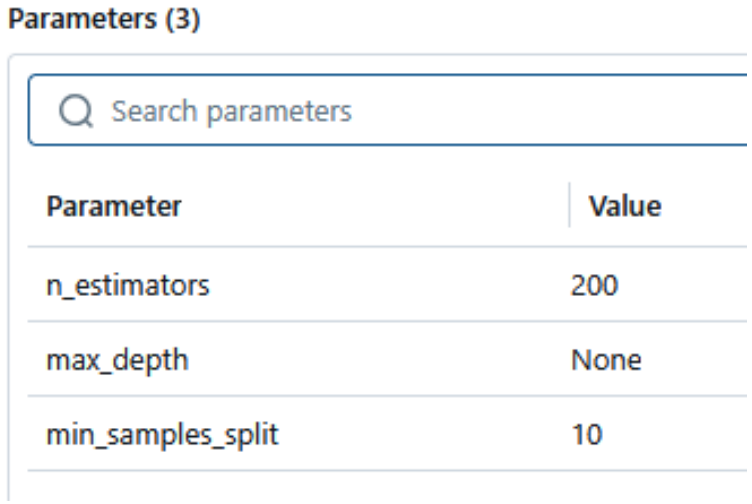
Se rodado o monitor.py, ou usado “check_drift” será feita a verificação de drift no modelo, se retornar positivo, é feito o re-treino do modelo com novo dataset automaticamente.

Ferramentas utilizadas:

- jupyter notebooks para fazer análise exploratória dos dados
- python 3.10 para rodar os códigos
- VS Code como IDE
- Scripts .bat para conveniência de instalação de dependências
- Github como repositório
- FastAPI para chamadas de API nos modelos

○ **Resultados e métricas dos modelos.**

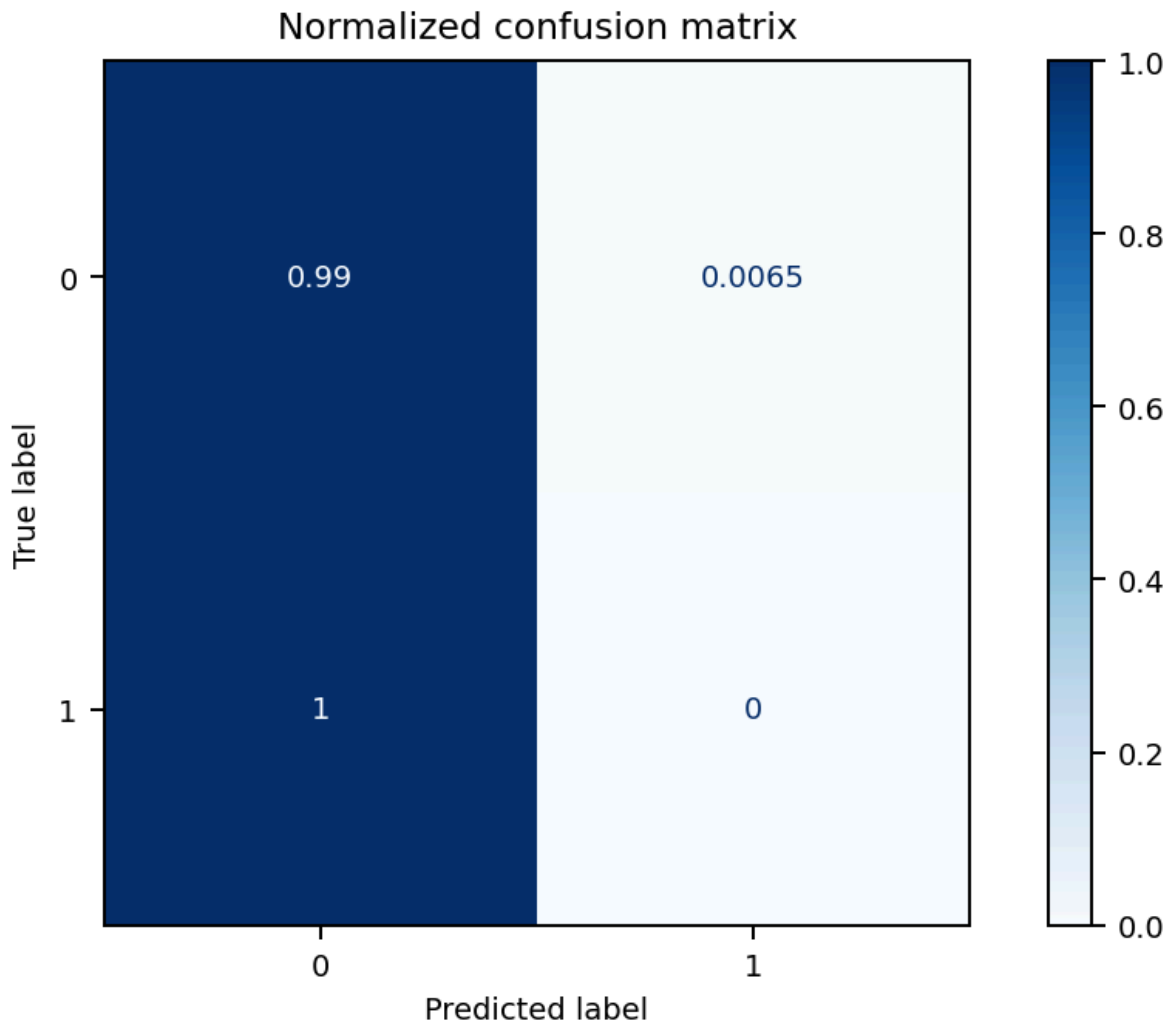
RandomForest Model



Metrics (12)

Search metrics

Metric	Value
accuracy	0.765
precision	0
recall	0
true_negatives	153
false_positives	1
false_negatives	46
true_positives	0
example_count	200
accuracy_score	0.765
recall_score	0
precision_score	0
f1_score	0



XGBoost

Parameters (3)

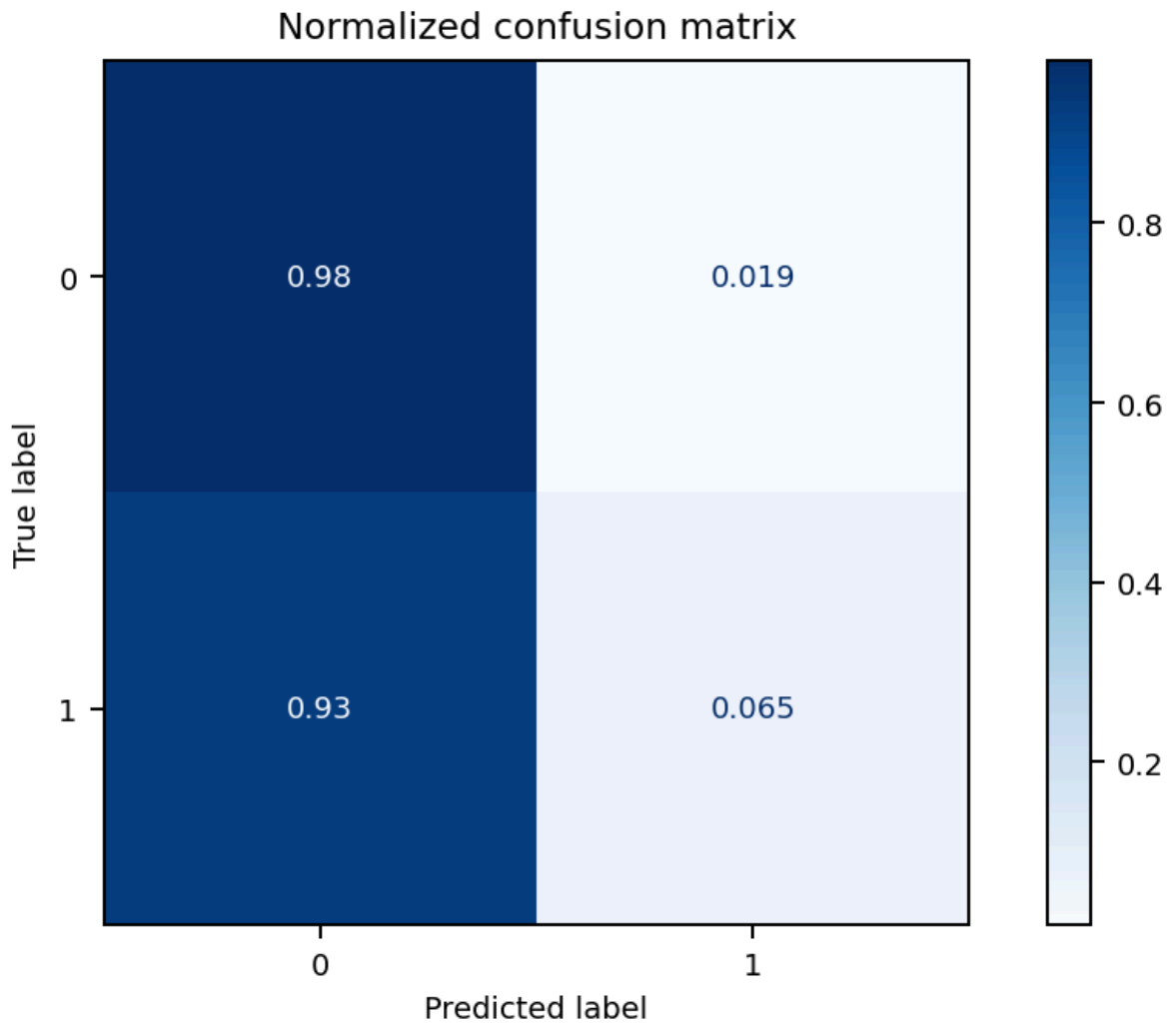
Search parameters

Parameter	Value
n_estimators	100
max_depth	3
learning_rate	0.1

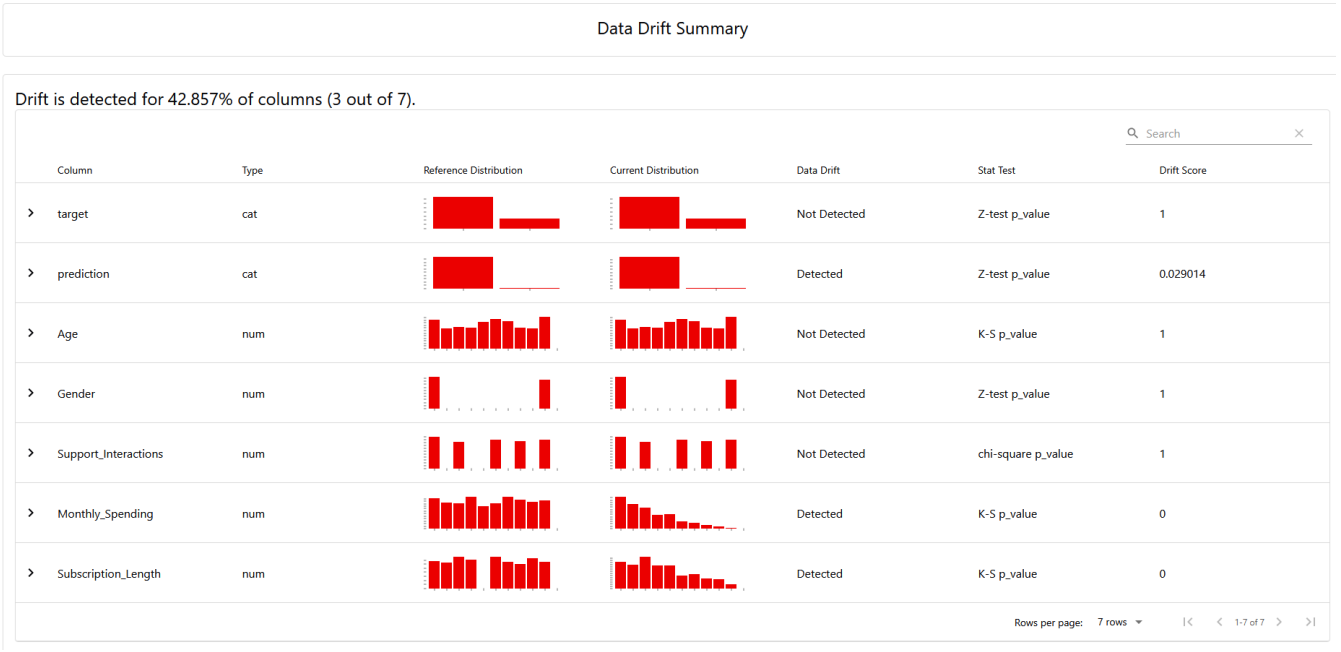
Metrics (12)

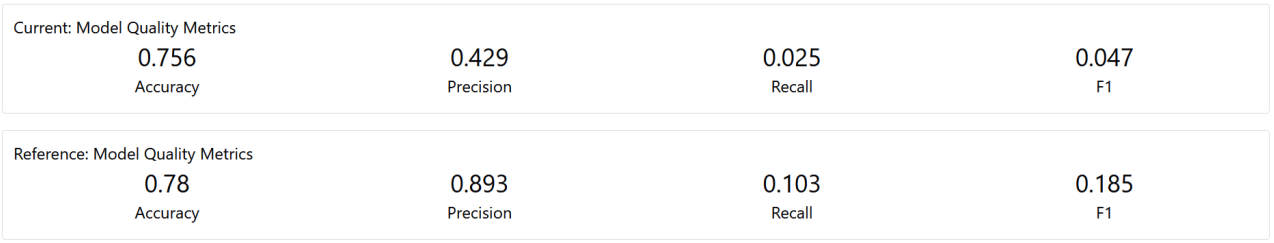
Search metrics

Metric	Value
accuracy	0.77
precision	0.5
recall	0.06521739130434782
true_negatives	151
false_positives	3
false_negatives	43
true_positives	3
example_count	200
accuracy_score	0.77
recall_score	0.06521739130434782
precision_score	0.5
f1_score	0.11538461538461539



XGBoost check-drift





○ Fluxo completo do pipeline

O fluxo ocorre em um ambiente local, os modelos são treinados em múltiplas variações de parâmetros, em seguida eles são registrados, e os modelos com os melhores scores são selecionados para serem promovidos ao estágio de Production.

Em seguida, podemos levantar os modelos para podermos rodar nossas APIs desenvolvidas em FastAPI. Além disso, é possível checar o modelo com um diferente dataset para verificar data drift, e caso aconteça, treinar o modelo com os novos dados.

○ Considerações finais

Considerando o objetivo dessa cadeira, nós focamos em desenvolver um pipeline de MLOps, utilizando um dataset significantemente simples, para evitar desviar o foco do trabalho e levantar um ambiente de MLOps. Em termos desse projeto, existem muitas melhoras que poderiam ser feitas, o trabalho em si foi bem interessante, principalmente desenvolvendo com bibliotecas que não tínhamos familiaridade.