Golden Data

DISRUPTIVE ARCHITECTURES:
IOT, IOB & GENERATIVE IA

98774 - Gabriel Previ de Oliveira (2TDSPF)
97850 - Gustavo Soares Fosaluza (2TDSPF)
551692 - Mateus Vinicius da Conceição (2TDSPV)
552000 - Pedro Henrique Figueiredo (2TDSPV)
550871 - Vitor da Silva Ruas (2TDSPV)

Problema

Vivemos na era da informação, diariamente somos bombardeados com milhares de informações, e é assim que tendências são geradas. A mutabilidade do setor de comércio eletrônico, traz consigo diversos desafios.

1

Divulgação de produtos

- Falta de precisão, ao divulgar anúncios

2

Retenção dos clientes

- A precariedade na compreensão dos clientes no pós compra, os afastam das empresas

3

Fidelização dos clientes

- Muitas empresas tem dificuldade em estabelecer clientes fiéis

Solução

Nossa solução é uma plataforma B2B, nosso objetivo é solucionar a dor das empresas

Visão Geral

Golden Data é uma plataforma que conecta diretamente empresas e clientes para otimizar a personalização dos serviços, especialmente para empresas de comércio eletrônico.

Cliente

Uma parte da plataforma será para o cliente, é nela que coletaremos os dados através de pesquisas que serão respondidas pelo usuário

Personalização

Receba anúncios e descontos que correspondem aos seus interesses e hábitos de consumo.

Descontos exclusivos

Cupons personalizados entregues diretamente por email, facilitando a obtenção de benefícios.

Relevância

Acesso a ofertas e produtos que realmente importam para você.

Mercado Alvo

Para onde está direcionado nosso foco?

Empresas que atuam no setor de e-commerce

O Golden Data pretende atender aos mais diversos segmentos de e-commerce do país

Empresa

Do lado da empresa, teremos uma plataforma por assinatura com avaliação gratuita e dois planos: Plano Beta, em que a empresa pagará o valor integral pelos serviços e o plano Sócios, que oferecerá desconto nos serviços,ter permissão de distribuir seus próprios cupons e ter acesso aos emails dos clientes, além de obter insights mais profundos e detalhados

Identificação de Tendências

Análise avançada de dados para identificar novas oportunidades de mercado.

Segmentação avançada

Uso Machine Learning e IA para segmentar o público alvo de forma precisa

Aumento nas conversões

Campanhas de marketing direcionadas aumentam as chances de venda.

Bibliotecas, Frameworks e outras tecnologias

Para o desenvolvimento do modelo de Machine Learning, utilizaremos algumas ferramentas

- **Python:** Linguagem principal
- **Pandas e NumPy:** Manipulação e limpeza de dados
- **Scikit-learn:** Modelagem, pré-processamento e avaliação
- **TensorFlow/Keras:** Redes neurais
- **Matplotlib/Seaborn:** Visualização de dados
- **Flask/FastAPI**: APIs RESTful para deploy
- **Docker:**Containerização e orquestração
- **Azure:** Hospedagem em nuvem

Bibliotecas utilizadas no protótipo

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

```
from faker import Faker
import random
```

Bibliotecas utilizadas no protótipo

- pandas: Manipulação e análise de dados em estruturas como DataFrame.
- LabelEncoder (sklearn): Converte dados categóricos em numéricos.
- StandardScaler (sklearn): Padroniza os dados com média 0 e desvio padrão 1.
- train_test_split (sklearn): Divide os dados em conjuntos de treino e teste.
- KNeighborsClassifier (sklearn): Algoritmo de classificação KNN baseado em vizinhos mais próximos.
- DecisionTreeClassifier (sklearn): Algoritmo que cria uma árvore de decisão para classificação.
- accuracy_score, precision_score, recall_score (sklearn): Métricas de desempenho do modelo.
- faker.Faker: Gera dados fictícios para testes.
- random: Gera números e seleções aleatórias.

Composição do protótipo

O nosso modelo protótipo é composto por dois componentes principais, um programa em Python e um Notebook do Pandas, feito no Google Colab

A lógica em Python “puro”, é responsável por gerar nossos dados para testes. utilizando a biblioteca Faker é possível gerar dados fictícios, combinamos com as funcionalidades da lib random para diversificar e randomizar os dados gerados e podemos escolher a quantidade que quisermos, basta alterar o valor da variável. Executamos tudo isso dentro de uma função que posteriormente é chamada passando a quantidade de linhas de dados como parâmetro. Após a geração dos dados eles são armazenados em um dataframe com colunas dedicadas e importado para um arquivo csv.

O arquivo csv gerado é consumido pelo nosso segundo componente, o notebook feito no Google Colab. Dentro dele são feitas todas as análises e treinamentos dos dados.

dataset_generator.py

```python
import pandas as pd
from faker import Faker
import random

fake = Faker()

def generate_data(num_records):
    data = []
    for _ in range(num_records):

        costumer_id = fake.random_int(min=1000, max=9999)
        age = fake.random_int(min=18, max=80)
        gender = random.choice(['M', 'F'])
        location = random.choice(['SP','RJ','BA','DF','MG','RS','SC','PR','AM','PA'])
        purchase_amount = round(random.uniform(20.0, 10000.0), 2)
        product_category = random.choice(['Eletronicos', 'Livros', 'Saúde', 'Moda', 'Esportes', 'Beleza', 'Casa', 'Brinquedos', 'Games'])
        clicked_ad = random.choice(['Sim', 'Não'])
        browsing_history = [fake.url() for _ in range(random.randint(1, 10))]
        interested_score = round(random.uniform(0.5, 1.0), 2)
        churn_risk = random.choice(['Low', 'Medium', 'High'])

        data.append([costumer_id, age, gender, location, purchase_amount, product_category, clicked_ad, browsing_history, interested_score, churn_risk])

    return data

num_records = 10000
data = generate_data(num_records)

columns = [
    'Customer_ID',
    'Age',
    'Gender',
    'Location',
    'Purchase_Amount',
    'Product_Category',
    'Clicked_Ad',
    'Browsing_History',
    'Interest_Score',
    'Churn_Risk'
]

df = pd.DataFrame(data, columns=columns)

df.to_csv('costumer_behavior.csv', index=False)

print('Arqivo gerado: costumer_behavior.csv')
```

You, 4 months ago • adicionando arquivos do projeto

GD-CostumerBehavior.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Última edição em 19 de maio

+ Código + Texto

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

Imporanto dados

O arquivo csv estará disponível no repositório do projeto

```python
dados = pd.read_csv('costumer_behavior.csv')
dados.head()
```

	Customer_ID	Age	Gender	Location	Purchase_Amount	Product_Category	Clicked_Ad	Browsing_History	Interest_Score	Churn_Risk
0	1462	21	F	MG	3308.26	Beleza	Não	['http://zuniga.net/', 'http://www.hoffman-dun...	0.67	Low
1	4465	57	F	SC	3081.58	Games	Não	['https://martin.com/', 'https://mills.com/', ...	0.62	Low
2	2001	27	F	RS	563.87	Eletronicos	Não	['http://price-joyce.com/', 'https://sims.com/...	0.85	High
3	8886	66	M	SC	6858.62	Saúde	Não	['http://hudson.org/', 'http://hernandez.biz/'...	0.60	Medium
4	5215	59	F	AM	7410.20	Games	Sim	['http://ray.com/', 'https://www.smith.info/',...	0.79	High

```python
# Tamanho do conjunto de dados
dados.shape
print(f'O conjunto de dados tem {dados.shape[0]} linhas e {dados.shape[1]} colunas.')
```

```
O conjunto de dados tem 10000 linhas e 10 colunas.
```

```python
# Nome das colunas
dados.columns
```

```
Index(['Customer_ID', 'Age', 'Gender', 'Location', 'Purchase_Amount',
       'Product_Category', 'Clicked_Ad', 'Browsing_History', 'Interest_Score',
       'Churn_Risk'],
      dtype='object')
```

```python
# Estatísticas descritivas das colunas numéricas
dados.describe()
```

	Customer_ID	Age	Purchase_Amount	Interest_Score
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	5498.503800	49.071400	5016.003426	0.751862

Shopify API

- Descrição: Permite acessar dados de produtos, pedidos, clientes e muito mais em lojas online.

WooCommerce REST API

- Descrição: Oferece dados sobre produtos, pedidos e clientes para lojas WooCommerce.

Clearbit API

- Descrição: Fornece dados de perfil e informações sobre empresas e contatos. Pode ajudar a preencher dados de perfil com informações reais.

FullContact API

- Descrição: Oferece informações sobre perfis de contatos, incluindo dados demográficos e sociais.

Mixpanel API

- Descrição: Permite coletar e analisar dados de comportamento de usuários em seu aplicativo ou site.

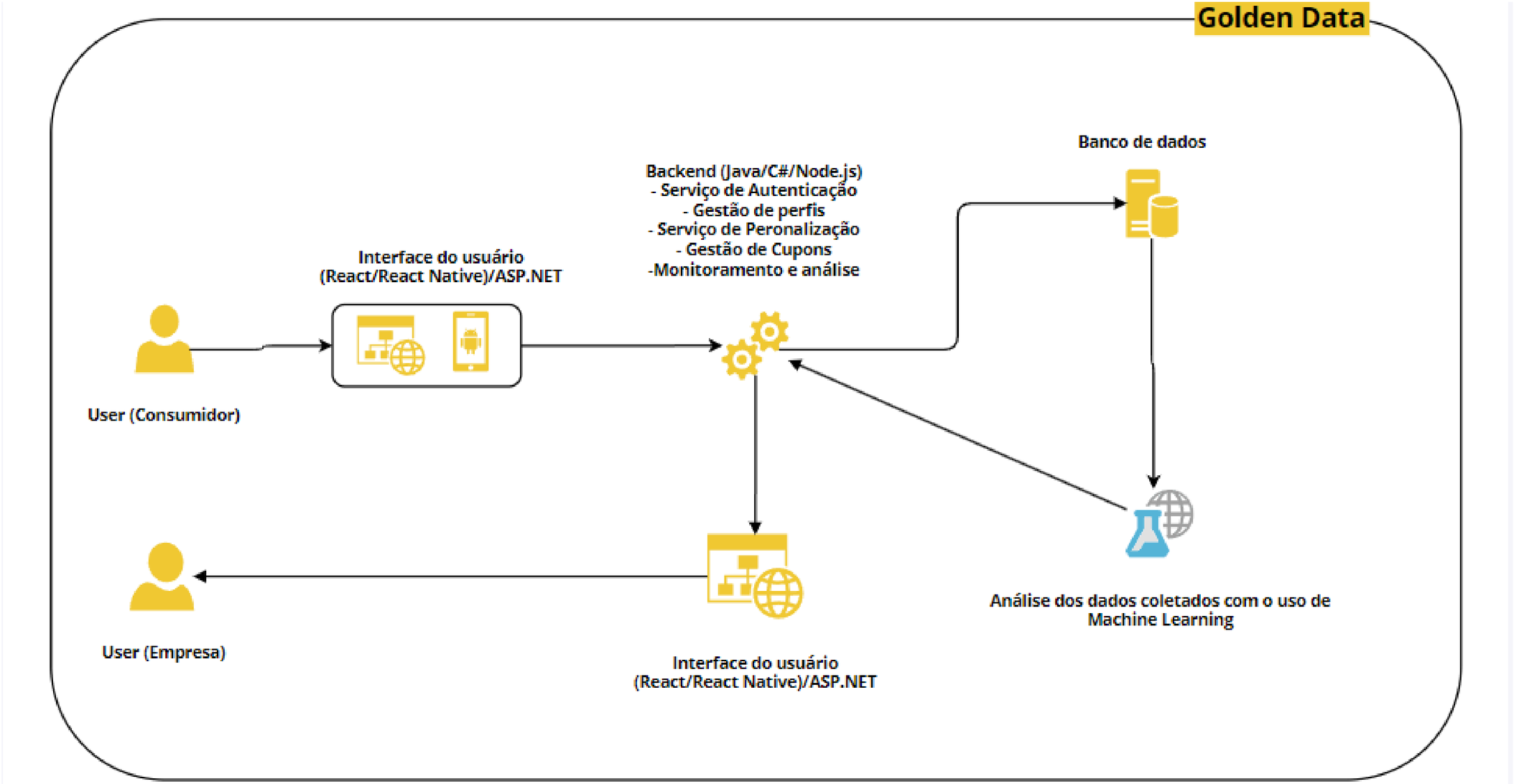
Google Analytics API

- Descrição: Fornece dados sobre o tráfego do site, comportamento dos usuários e mais.

Amazon Product Advertising API

- Descrição: Permite acessar informações sobre produtos, preços e avaliações na Amazon.

Arquitetura da plataforma



1. Usuários (Consumidor e Empresa)

- Existem dois tipos de usuários:
  - User (Consumidor): Este grupo de usuários interage com a plataforma para indicar seus interesses e receber cupons e anúncios personalizados de acordo com essas preferências.
  - User (Empresa): As empresas utilizam a plataforma para acessar os dados fornecidos pelos consumidores, criando campanhas personalizadas e enviando cupons e ofertas diretamente aos consumidores.

2. Interface do Usuário (React/React Native/Kotlin/ASP.NET)

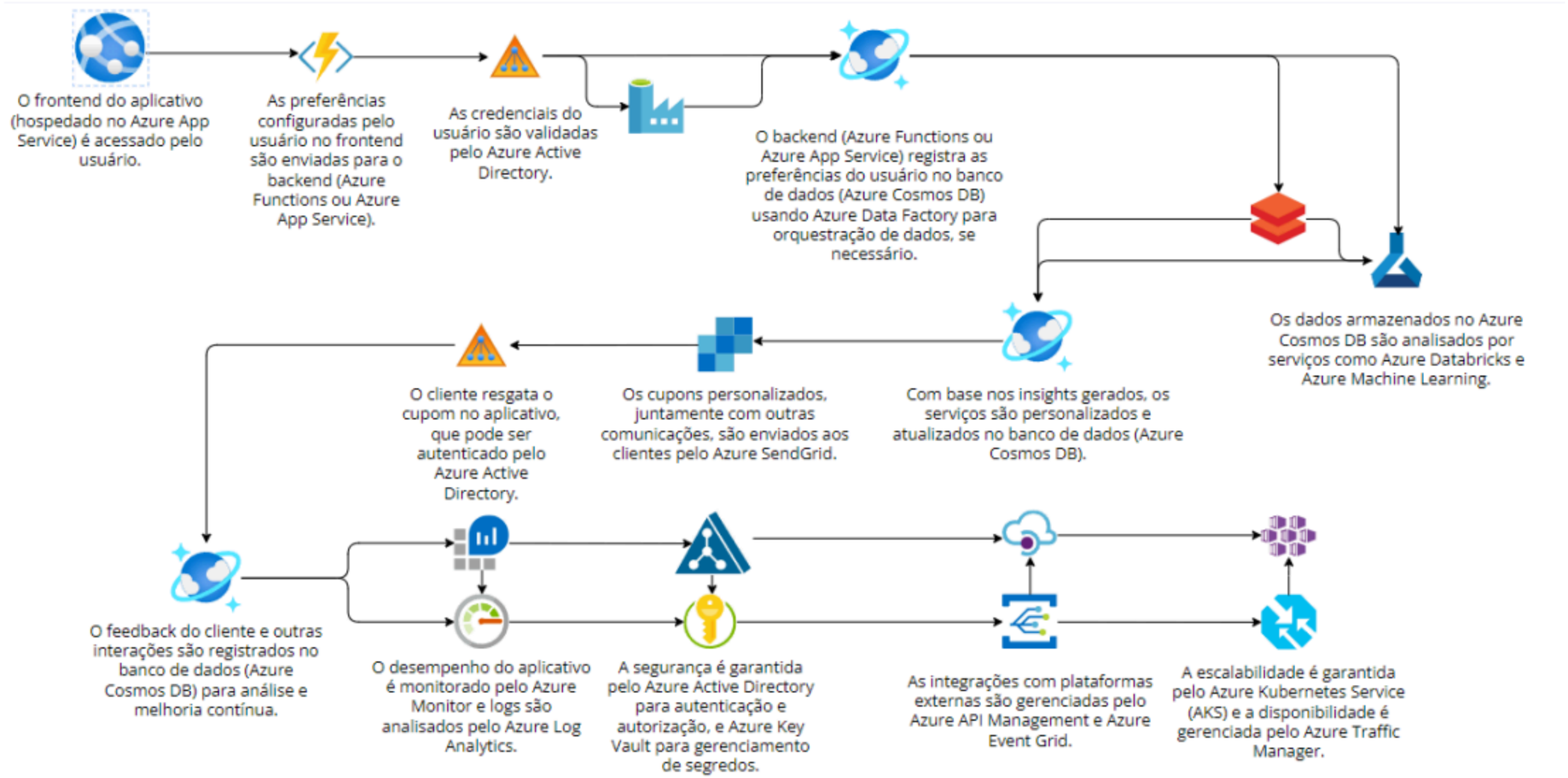
- Mobile (Kotlin/React Native):
  - A parte mobile da interface do usuário será desenvolvida com Kotlin (para aplicativos nativos Android) e React Native (para uma experiência multiplataforma). A escolha de Kotlin para Android permite aproveitar os recursos nativos da plataforma, garantindo desempenho otimizado.
  - React Native será usado para fornecer uma interface que funcione em diversas plataformas móveis, incluindo iOS e Android.
- Web (ASP.NET):
  - No ambiente web, a interface será desenvolvida usando ASP.NET, fornecendo uma interface robusta para empresas e consumidores interagirem com a plataforma diretamente de navegadores.

3. Backend (Java/C#/Node.js)

- O backend é responsável por toda a lógica de negócios e é desenvolvido com Java, C# ou Node.js. Ele suporta múltiplos serviços, incluindo:
  - Serviço de Autenticação: Garante que apenas usuários registrados possam acessar a plataforma, diferenciando o login de consumidores e empresas.
  - Gestão de Perfis: Armazena e gerencia dados de perfil dos consumidores, como preferências, e os perfis das empresas, incluindo suas campanhas de anúncios.
  - Serviço de Personalização: Baseado nas preferências fornecidas pelos consumidores, ele cria e oferece recomendações e cupons personalizados.
  - Gestão de Cupons: Gerencia o ciclo de vida dos cupons gerados pelas empresas e enviados para os consumidores.
  - Monitoramento e Análise: Este serviço monitora as interações e a atividade dentro da plataforma, fornecendo insights para otimização do serviço.

4. Banco de Dados
   - O banco de dados armazena as informações dos perfis dos consumidores e das empresas, além de dados relacionados aos cupons e atividades na plataforma. Esse componente é essencial para fornecer as informações necessárias ao backend para que a personalização de serviços seja eficiente.
5. Análise de Dados com Machine Learning
   - A análise de dados é realizada com técnicas de Machine Learning que processam as informações coletadas, buscando padrões nos interesses e comportamentos dos consumidores. Essa análise permite que as empresas ofereçam campanhas mais direcionadas e eficazes, com base nos insights gerados.
   - A personalização não se limita apenas às preferências explícitas dos consumidores, mas também pode ser ajustada com base nos comportamentos identificados pela análise de dados.

Arquitetura Azure



- Frontend no Azure App Service: O aplicativo é acessado pelo usuário, e as preferências configuradas no frontend são enviadas para o backend.
- Autenticação via Azure Active Directory: As credenciais do usuário são validadas pelo Azure Active Directory, garantindo segurança e controle de acesso.
- Backend no Azure Functions ou App Service: O backend recebe as preferências do usuário e as armazena no banco de dados Azure Cosmos DB, utilizando Azure Data Factory, se necessário, para orquestrar os dados.
- Análise de dados no Cosmos DB: Os dados armazenados no Cosmos DB são analisados por serviços como Azure Databricks e Azure Machine Learning, para gerar insights.
- Personalização e envio de cupons: Com base nos insights gerados, os cupons personalizados são enviados aos clientes via Azure SendGrid.
- Escalabilidade e integração com outros serviços: O sistema é escalável usando Azure Kubernetes Service (AKS), e a disponibilidade é gerenciada pelo Azure Traffic Manager. As integrações com plataformas externas são gerenciadas pelo Azure API Management e Azure Event Grid.
- Monitoramento e segurança: O desempenho do aplicativo é monitorado pelo Azure Monitor e Azure Log Analytics, enquanto a segurança é reforçada com Azure Key Vault para gerenciamento de segredos.
- Feedback do cliente e melhorias contínuas: O feedback do cliente e interações são armazenados no Azure Cosmos DB para análise e melhoria contínua.

Golden Data

Projeto final

.NET + Python

Nosso projeto final surgiu de uma oportunidade de integrar o aprendizado de máquina com o projeto de Web API do Golden Data feito com ASP .NET

API em .NET Core para previsão de sentimento a partir de comentários e avaliações de clientes, usando aprendizado de máquina com o ML.NET. A API lê comentários e avaliações de clientes, passa por um modelo de ML treinado, e retorna se o sentimento é "Positivo" ou "Negativo". Ele não usa técnicas avançadas de Processamento de Linguagem Natural (NLP) para interpretar o contexto ou as nuances do texto do comentário. Em vez disso, a decisão sobre o sentimento é influenciada mais pela avaliação numérica (de 1 a 5) do que pelo conteúdo textual do comentário. Este modelo foi criado conforme os passos ensinados em aula com algumas modificações para atender os requisitos do resultado desejado.

.NET + Python

Nosso projeto final surgiu de uma oportunidade de integrar o aprendizado de máquina com o projeto de Web API do Golden Data feito com ASP .NET

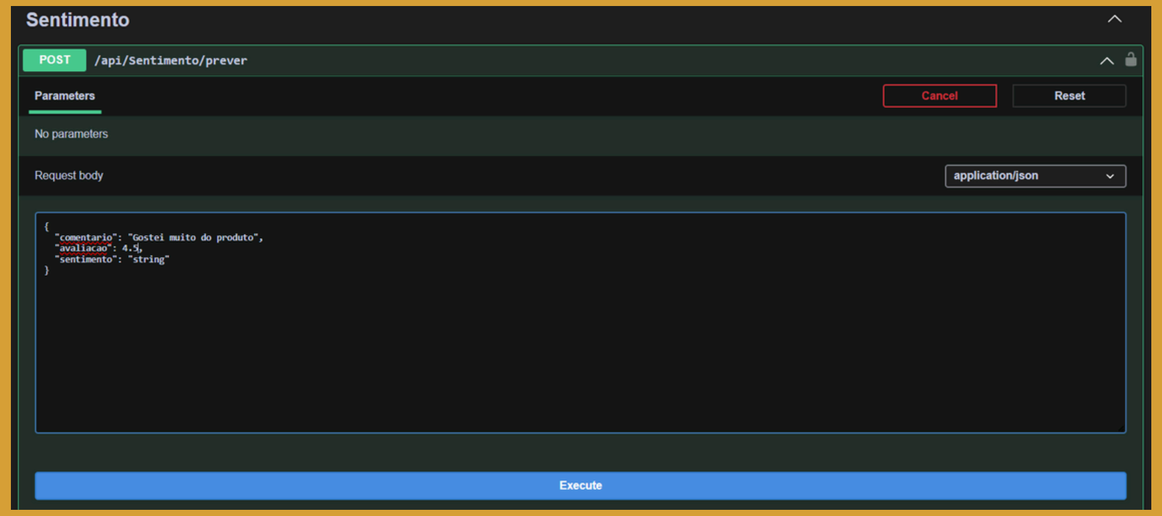
Para o funcionamento adequado da API, é imprescindível que ela receba os dados necessários por meio do endpoint. Esses dados são extraídos de dois arquivos CSV: um para o treinamento e outro para o teste. A criação desses arquivos é realizada por um sistema simples desenvolvido em Python, que randomiza os dados, os divide em conjuntos de treinamento e teste, e, posteriormente, os armazena nos respectivos arquivos CSV.

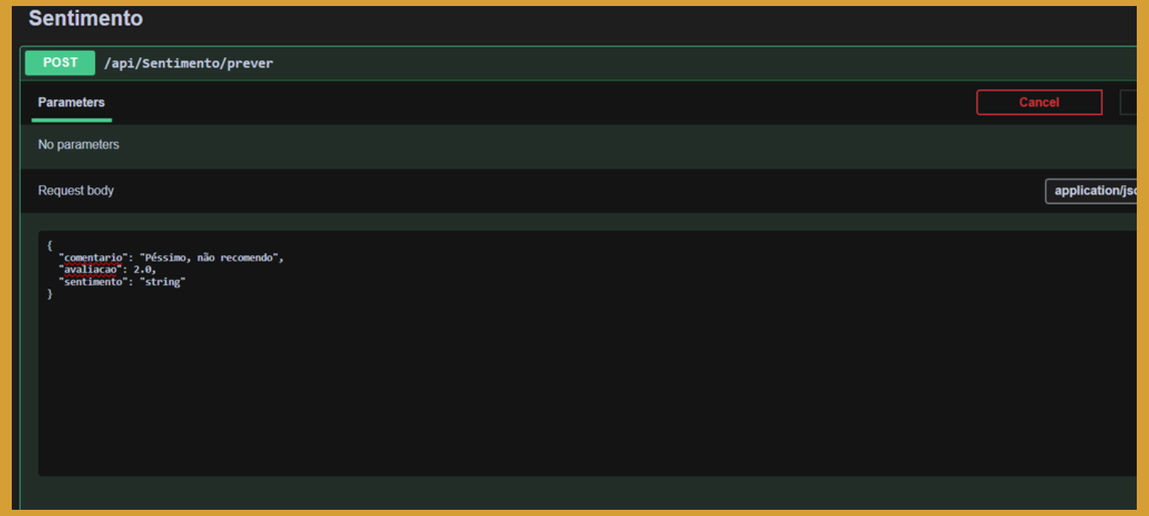
Análise de sentimento

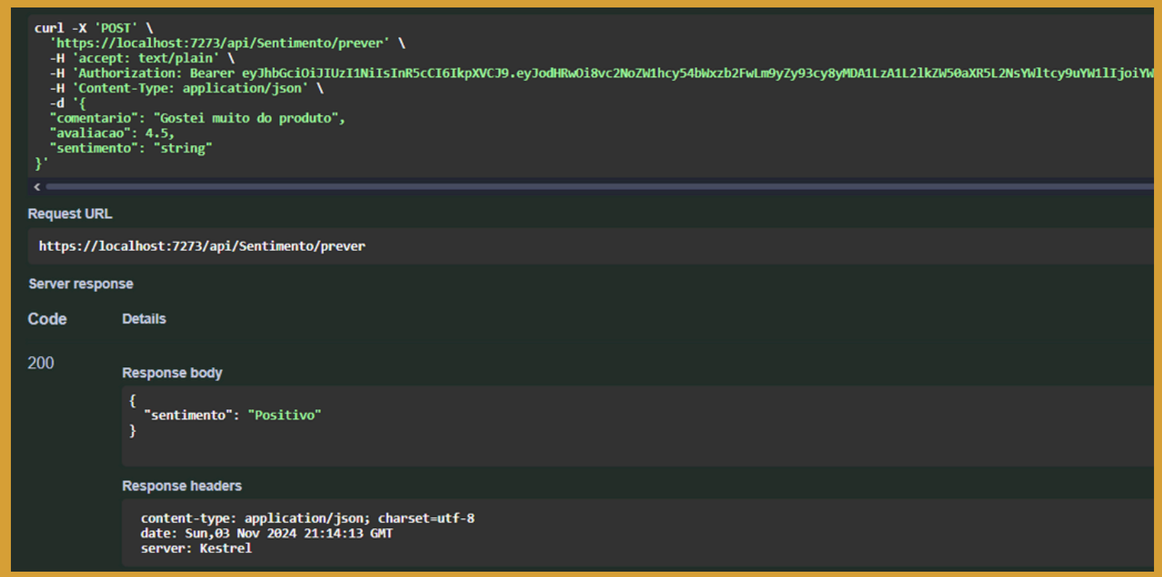
```
// Método para treinar o modelo e salvar como arquivo .zip
1 reference
private void TreinarModelo()
{
    var pastaModelo = Path.GetDirectoryName(caminhoModelo);
    if (!Directory.Exists(pastaModelo))
    {
        Directory.CreateDirectory(pastaModelo);
        Console.WriteLine($"Diretório criado: {pastaModelo}");
    }

    // Carregar dados de treinamento
    IDataView dadosTreinamento = mlContext.Data.LoadFromTextFile<FeedbackInput>(
        path: caminhoTreinamento, hasHeader: true, separatorChar: ',');

    // Definir a pipeline de transformações e treinamento
    var pipeline = mlContext.Transforms.Text.FeaturizeText(
            outputColumnName: "ComentarioFeaturizado",
            inputColumnName: nameof(FeedbackInput.Comentario))
        .Append(mlContext.Transforms.Concatenate(
            "Features",
            "ComentarioFeaturizado",
            nameof(FeedbackInput.Avaliacao)))
        .Append(mlContext.Transforms.Conversion.MapValueToKey(
            outputColumnName: "Label",
            inputColumnName: nameof(FeedbackInput.Sentimento)))
        .Append(mlContext.MulticlassClassification.Trainers.OneVersusAll(
            mlContext.BinaryClassification.Trainers.SdcaLogisticRegression()))
        .Append(mlContext.Transforms.Conversion.MapKeyToValue(
            outputColumnName: "PredictedLabel",
            inputColumnName: "PredictedLabel"));

    // Treinamento do modelo
    var modelo = pipeline.Fit(dadosTreinamento);

    // Salvar o modelo treinado em um arquivo .zip
    mlContext.Model.Save(modelo, dadosTreinamento.Schema, caminhoModelo);
    Console.WriteLine($"Modelo treinado e salvo em: {caminhoModelo}");
}
```

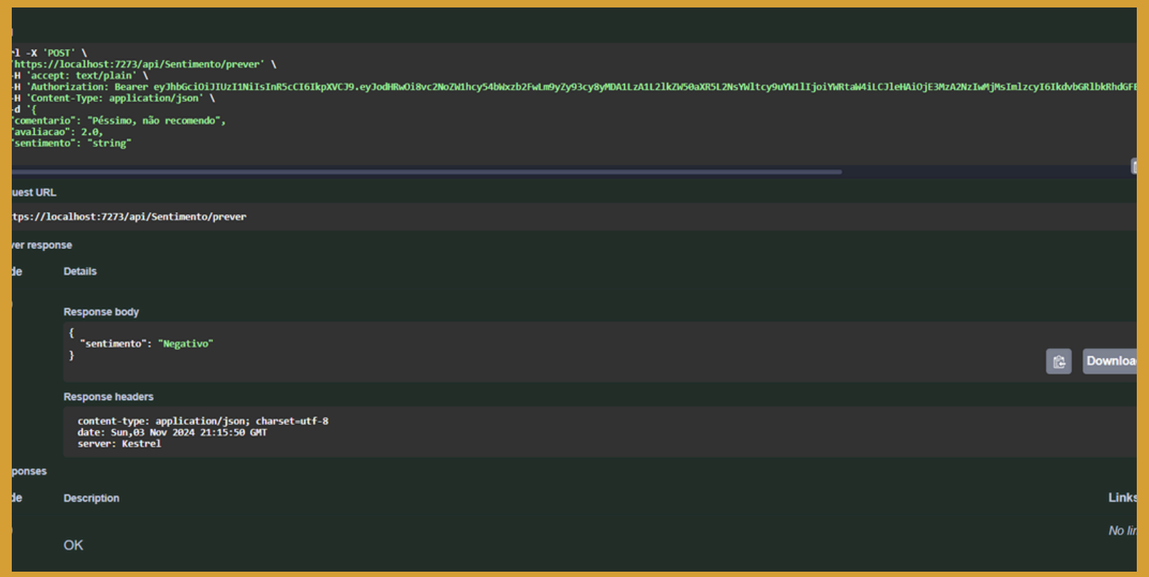
```
# Função para gerar dados aleatórios
def gerar_dados_aleatorios(num_dados):
    dados = []
    for _ in range(num_dados):
        sentimento = random.choice(["Positivo", "Neutro", "Negativo"])
        if sentimento == "Positivo":
            comentario = random.choice(comentarios_positivos)
            avaliacao = random.randint(4, 5)  # Avaliação alta
        elif sentimento == "Neutro":
            comentario = random.choice(comentarios_neutros)
            avaliacao = 3  # Avaliação média
        else:
            comentario = random.choice(comentarios_negativos)
            avaliacao = random.randint(1, 2)  # Avaliação baixa
        dados.append((comentario, avaliacao, sentimento))
    return dados

# Gerar 100 dados aleatórios
num_dados = 100
dados_aleatorios = gerar_dados_aleatorios(num_dados)

# Criar DataFrame a partir dos dados gerados
df = pd.DataFrame(dados_aleatorios, columns=["Comentario", "Avaliacao", "Sentimento"])

# Dividir em 80% para treino e 20% para teste
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)

# Salvar em arquivos CSV
train_df.to_csv("feeling-train.csv", index=False)
test_df.to_csv("feeling-test.csv", index=False)

print("Arquivos train_data.csv e test_data.csv foram gerados com sucesso!")
```

Análise de sentimento









Reflexão

- O que funcionou ?
- O que não funcionou ?
- O que foi aprendido ?
- O que faríamos de diferente ?
-

Reflexão

O que funcionou?

Embora o projeto final tenha uma estrutura simples, alcançamos nosso objetivo principal de implementar aprendizado de máquina para recomendação de produtos e análise de sentimentos. De certa forma, conseguimos atingir ambos os propósitos: desenvolvemos um protótipo e implementamos uma API de análise de sentimentos no projeto em .NET. Conseguimos, ainda, utilizar várias das ferramentas e bibliotecas mencionadas na descrição do projeto e integrá-las a um ambiente Web.

Reflexão

O que não funcionou?

- Não conseguimos integrar serviços externos, como as APIs do Shopify, Google Analytics e Amazon, que poderiam fornecer dados essenciais para nossa plataforma. Parte dessas APIs são pagas e possuem custos elevados.
- Não tivemos tempo suficiente para explorar técnicas avançadas de Processamento de Linguagem Natural (NLP) que nos permitiriam interpretar melhor o contexto e as nuances dos comentários, em parte devido à mudança de objetivo do projeto, que inicialmente focava em recomendação de produtos.
- A arquitetura em nuvem foi limitada; apesar de ser abordada na disciplina de DevOps, a infraestrutura disponível não contemplava suporte a IA e Machine Learning.

Reflexão

O que foi aprendido?

- Lidar com dados é uma tarefa complexa e trabalhosa, desde a coleta e criação da base até as etapas de análise, classificação, treinamento, teste e maturação da inteligência artificial.

- Aprendemos o valor e a aplicabilidade de Deep Analytics, Machine Learning e IA, que são cada vez mais incorporados a plataformas, veículos, celulares, indústrias, serviços e automações em geral.

Reflexão

O que faríamos diferente?

Com mais tempo e recursos, integraríamos os modelos de análise de sentimentos e recomendação de produtos em uma API Web para criar a base do sistema Golden Data. Usaríamos APIs auxiliares, como Shopify, Google Analytics e Amazon, para enriquecer os dados e aumentar a precisão das recomendações. Além disso, exploraríamos técnicas de NLP para entender melhor o contexto dos comentários e oferecer análises mais precisas e relevantes.

Futuro do Projeto

Para o futuro da plataforma Golden Data, a expansão poderia incluir segmentação avançada e personalização em tempo real, permitindo que empresas ajustem campanhas conforme o comportamento atual dos clientes. Além disso, a integração de inteligência artificial possibilitaria uma análise profunda das preferências dos consumidores, ampliando o impacto das campanhas de marketing. Ferramentas de automação permitiriam o envio de conteúdos personalizados automaticamente, enquanto métricas preditivas de ROI ajudariam a otimizar investimentos em marketing. Por fim, um módulo interativo de feedback direto dos clientes ajudaria as empresas a adaptar seus serviços e campanhas de acordo com as necessidades reais dos consumidores. Essas melhorias fortaleceriam a proposta da Golden Data de personalização e segmentação precisa, aumentando o engajamento e o retorno sobre investimento (ROI).