



TESTE TÉCNICO

DATA ENGINEER

Você está recebendo o teste técnico para a vaga de **engenheiro(a) de dados** da Data Science Brigade. Selecionamos um exemplo típico de projeto desenvolvido na DSB para classificação de documentos de texto.

OBJETIVO

Suponha que a equipe de ciência de dados já tenha modelado o problema e produzido um script em python para treinar uma pipeline de Machine Learning dentro do framework PySpark, **queremos que você desenhe um fluxo com serviços da AWS**, que deverá lidar com a ingestão dos dados, a rotina de treinamento recorrente do algoritmo e da aplicação do algoritmo em tempo real.

Você deve entregar um PDF com um diagrama e/ou uma descrição textual explicando a modelagem da pipeline de dados incluindo: os serviços você utilizaria e as etapas e fluxos necessários para modelar este problema. Lembre-se de justificar bem as suas escolhas, esse é o ponto mais importante para nossa avaliação.

As etapas e processos envolvidos bem como os schemas/exemplo de formato de dados são descritos nas seções abaixo.

Opcional: caso deseje adicionalmente criar um mock demonstrando a infraestrutura na AWS, nos comunique e adicione nossos e-mails ao projeto (lukas.mendes@dsbrigade.com e bruno@dsbrigade.com), nos dando permissão suficiente para visualizar os serviços criados.



PROCESSOS E REQUISITOS

- INPUT
 - Os dados de entrada devem ser consumidos a partir de um serviço de mensageria, no formato JSON [[JSON de entrada](#)]
- TREINAMENTO DO ALGORITMO DE MACHINE LEARNING
 - O script que treina o algoritmo de Machine Learning estaria disponível, podemos chamá-lo de `train_ml_model.py`
 - [INPUT] Este script exige que os dados estejam no formato .parquet [[.parquet de entrada para o algoritmo](#)]
 - [OUTPUT] O script gera um diretório que representa o modelo PySpark e possui um nome padrão `doc-classification-model-YYYY-MM-DD`, indicando a data em que este modelo foi treinado.
 - [OUTPUT] É possível escolher para onde salvar o modelo, desde que esteja em um storage resiliente.
 - Este treinamento deverá ocorrer a cada 4 semanas. Toda vez que o algoritmo for treinado, o script irá ler todos os dados históricos disponíveis até a data atual.
- CLASSIFICAÇÃO EM TEMPO REAL
 - Também está disponível um script que aplica o modelo ML pre-treinado a novos dados, usando Spark Structured Streaming: `apply_ml_model.py`
 - [INPUT] Esse script deverá ler os documentos no formato JSON a partir do serviço de mensageria de entrada
 - [INPUT] É preciso indicar ao script qual o modelo a ser aplicado, tipicamente é o modelo que foi treinado mais recentemente.
 - [OUTPUT] O script irá aplicar o modelo nos dados e retornará um JSON de saída [[JSON de saída do algoritmo](#)]
 - [OUTPUT] A saída deverá ser escrita em um novo tópico JSON, para consumo de diversas APIs.



DADOS

1. JSON de entrada

Exemplo de um JSON de entrada:

```
{
  "eventTimestamp": "2019-04-02T21:55:57Z",
  "eventId": "3dc6eb9af4154635b688e7cfeb8db131bfe84ccd",
  "eventSource": "doc.contract",
  "eventVersion": "1.0",
  "eventType": "DocSchedule",
  "doc": {
    "schemaVersion": "1.0",
    "docId": "CF2909E5-12C8-4D42-A996-12672B727B3B",
    "metadata": {
      "accountId": "EA2425CF-EF72-40C6-8DD3-C7D5310FA07D",
      "superTag01": "A",
      "superTag02": "C",
      "otherTags": [
        {"tagName": "otherTag01", "tagValue": "A"},
        {"tagName": "otherTag02", "tagValue": "X"}
      ]
    }
  },
  "payload": <TEXT>
}
```

2. Formato do arquivo .parquet de entrada para o treinamento do algoritmo ML

Para treinar o algoritmo, é preciso que o arquivo parquet seja adaptado a partir dos dados originais e contenha as seguintes colunas:



```
eventTimestamp (TIMESTAMP)
eventSource (STRING)
eventType (STRING)
eventVersion (DECIMAL)
accountId (STRING)
superTag01 (ENUM)
superTag02 (ENUM)
concatOtherTags (STRING) # Example: otherTag01:A|otherTag02:X
Payload (STRING)
```

3. JSON de saída

```
{
  "eventTimestamp": "2019-04-02T22:50:14Z",
  "eventId": "ff0fafe919a5b9e37e66b59ac1639c8f2123ac6a",
  "eventSource": "doc.contract",
  "eventVersion": "1.0",
  "eventType": "DocClassified",
  "doc": {
    "schemaVersion": "1.0",
    "docId": "CF2909E5-12C8-4D42-A996-12672B727B3B",
    "accountId": "EA2425CF-EF72-40C6-8DD3-C7D5310FA07D",
    "class": "SPAM",
    "metrics": {
      "score": 0.8,
      "report": {
        "docScore": 0.85,
        "docScoreConfidence": 0.9,
        "accountScore": 0.75,
        "accountScoreConfidence": 0.9
      }
    }
  }
}
```