PicPay

Case Machine Learning Engineer

Escopo

Este teste consiste em criar uma solução de transformação de dados, treino de modelo e escoragem online. Para isso deverá ser entregue um **link de um repositório GitHub** contendo a seguinte estrutura:

- **/src/** - Códigos da API
- **/notebook/** - Contém o arquivo notebook com as transformações do dado, respostas das perguntas e treinamento do modelo
- **/docs/** - Desenho da arquitetura
- **/tests/** - Testes unitários

Regras / Orientações

- Clone esse repositório para usar como base no seu case
- Você terá **15 dias corridos** a partir do recebimento deste email para fazer a entrega final via `Github`, em um repositório público e o link do repositório deverá ser enviado para a plataforma Gupy em resposta ao email de recebimento do desafio;
- Durante todo o período o **time estará disponível** para dúvidas no email `data.mlops@picpay.com`;
- O foco do teste é avaliar como você se sai em um desafio de rotinas de Engenheiro de Machine Learning bem como você lida ao aprender novas tecnologias;
- Caso não consiga terminar 100% do proposto, recomendamos que faça as entregas mesmo assim para que o time possa avaliar seu desempenho;
- O uso de ferramentas como **Google** e **ChatGPT** é permitido porém, iremos avaliar e questionar a solução entregue durante a entrevista técnica;

CheckList de Entrega

- A API deverá ser feita em **Python** e Conteinerizada no docker. A API deverá ter os seguintes endpoints:
  - `/model/predict/`
    - Endpoint onde deverá receber um payload com as informações do voo e retornar a previsão do atraso no destino
  - `/model/load/`
    - Endpoint onde deverá receber o arquivo .pkl do modelo e deixar a API pronta para realizar predições
  - `/model/history/`
    - Endpoint onde deverá exibir o histórico de predições realizadas (o payload de entrada + as saídas preditas)
  - `/health/`
    - Endpoint que irá retornar a saúde da API
- O Notebook deverá ser exportado no formato **.ipynb** e estar dentro do repositório git.

    - Deverá realizar as transformações utilizando spark:
    - Responder o conjunto de perguntas contidas nesse documento
- **Desenho** da arquitetura:
    - Apresentar um desenho simples de como essa arquitetura poderia funcionar dentro de um ambiente Cloud;
    - O desenho da arquitetura pode ser apenas uma **imagem** (.png, .jpg)

Você deverá apresentar a solução durante a entrevista técnica

Ambiente

Spark

Para a execução das tarefas que utilizam spark, você pode utilizar o ambiente da sua preferência ou escolher uma das opções abaixo:

- **Databricks Community Edition** (recomendado) - Acesse o Databricks Community Edition, cadastre-se com seu e-mail e verifique sua conta. A partir daí, vá ao painel inicial clique em **Home**, depois em **Create** e escolha **Notebook;**
- **Google Colab** - O Google Colab é uma plataforma de desenvolvimento interativo. Porém, ele não possui spark nativamente configurado. Você pode seguir o tutorial desse link para habilitar o spark e fazer seu teste por lá

Base de Dados

Base de Informações de Voos

A base de dados de voos contém informações detalhadas sobre voos ocorridos em 2013. Ela pode ser obtida através deste link. (*repo_git/notebook/airports-database.zip*) As colunas do conjunto de dados são:

Coluna	Descrição
id	Um identificador único para cada registro de voo.
year	O ano em que o voo ocorreu (2013 neste conjunto de dados).
month	O mês em que o voo ocorreu (1 a 12).
day	O dia do mês em que o voo ocorreu (1 a 31).
dep_time	O horário local real de partida do voo, no formato 24 horas (hhmm).
sched_dep_time	O horário local programado de partida do voo, no formato 24 horas (hhmm).
dep_delay	A diferença entre os horários real e programado de partida do voo, em minutos. Um valor positivo indica uma partida atrasada, enquanto um valor negativo indica uma partida adiantada.

arr_time	O horário local real de chegada do voo, no formato 24 horas (hhmm).
sched_arr_time	O horário local programado de chegada do voo, no formato 24 horas (hhmm).
arr_delay	A diferença entre os horários real e programado de chegada do voo, em minutos. Um valor positivo indica uma chegada atrasada, enquanto um valor negativo indica uma chegada adiantada.
carrier	O código de duas letras da companhia aérea do voo.
flight	O número do voo.
tailnum	O identificador único da aeronave usada no voo.
origin	O código de três letras do aeroporto de origem do voo.
dest	O código de três letras do aeroporto de destino do voo.
air_time	A duração do voo, em minutos.
distance	A distância entre os aeroportos de origem e destino, em milhas.
hour	O componente da hora do horário programado de partida, no horário local.
minute	O componente dos minutos do horário programado de partida, no horário local.
time_hour	O horário programado de partida do voo, no formato local e de data-hora (yyyy-mm-dd hh
name	O nome da companhia aérea do voo.

Base de Coordenadas e Clima

Para enriquecer a base de dados, você precisará utilizar duas APIs externas:

1. **Weatherbit API**: Fornece dados históricos sobre as condições meteorológicas.
    - **Site**: Weatherbit API
    - **Como se cadastrar**: Crie uma conta no site para obter a chave da API (API Key).
    -
2. **AirportDB API**: Fornece informações detalhadas sobre aeroportos, incluindo coordenadas geográficas.
    - **Site**: AirportDB API
    - **Como se cadastrar**: Crie uma conta no site para obter a chave da API (API Token).

PicPay

Perguntas

Responda às seguintes perguntas utilizando PySpark:

1. Qual é o número total de voos no conjunto de dados?
2. Quantos voos foram cancelados? (Considerando que voos cancelados têm **`dep_time`** e **`arr_time`** nulos)
3. Qual é o atraso médio na partida dos voos (**`dep_delay`**)?
4. Quais são os 5 aeroportos com maior número de pousos?
5. Qual é a rota mais frequente (par **origin-dest**)?
6. Quais são as 5 companhias aéreas com maior tempo médio de atraso na chegada? (Exiba também o tempo)
7. Qual é o dia da semana com maior número de voos?
8. Qual o percentual mensal dos voos tiveram atraso na partida superior a 30 minutos?
9. Qual a origem mais comum para voos que pousaram em Seattle (**SEA**)?
10. Qual é a média de atraso na partida dos voos (**`dep_delay`**) para cada dia da semana?
11. Qual é a rota que teve o maior tempo de voo médio (**air_time**)?
12. Para cada aeroporto de origem, qual é o aeroporto de destino mais comum?
13. Quais são as 3 rotas que tiveram a maior variação no tempo médio de voo (**air_time**) ?
14. Qual é a média de atraso na chegada para voos que tiveram atraso na partida superior a 1 hora?
15. Qual é a média de voos diários para cada mês do ano?
16. Quais são as 3 rotas mais comuns que tiveram atrasos na chegada superiores a 30 minutos?
17. Para cada origem, qual o principal destino?

Material de Apoio

Enriquecimento da Base de Dados

Usando as APIs do Weatherbit e AirportDB, enriqueça a base de dados de voos com informações sobre a velocidade do vento para os aeroportos de origem e destino. Utilize as coordenadas dos aeroportos fornecidas pela API AirportDB e obtenha os dados meteorológicos históricos da API Weatherbit para as datas correspondentes aos voos.

Weatherbit API

Campos Relevantes:

- `wind_spd`: Velocidade do vento em metros por segundo (m/s).

Exemplo de Chamada:

```
import requests

weatherbit_key = 'SUA_WEATHERBIT_API_KEY'
latitude = 40.7128
longitude = -74.0060
start_date = '2023-01-01'
end_date = '2023-01-02'  # Sempre adicione +1 dia em relação ao start_date

url = 'https://api.weatherbit.io/v2.0/history/daily'
params = {
    'lat': latitude,
    'lon': longitude,
    'start_date': start_date,
    'end_date': end_date,
    'key': weatherbit_key,
}
headers = {
    'Accept': 'application/json',
}
response = requests.get(url, params=params, headers=headers)
data = response.json()
data
```

AirportDB API

Campos Relevantes:

- `latitude_deg`: Latitude do aeroporto.
- `longitude_deg`: Longitude do aeroporto.

Exemplo de Chamada:

```
import requests
airportdb_key = 'SUA_AIRPORTDB_API_KEY'
airport_code = 'JFK'
url =
f"https://airportdb.io/api/v1/airport/K{airport_code}?apiToken={airportdb_key}"
response = requests.get(url)
data = response.json()
data
```

Observe que é necessário adicionar a letra K antes de realizar a chamada ao endpoint.

Pergunta final: Enriqueça a base de dados de voos com as condições meteorológicas (velocidade do vento) para os aeroportos de origem e destino. Mostre as informações enriquecidas para os 5 voos com maior atraso na chegada.

Modelo de ML

O objetivo desse teste não é avaliar a capacidade de criar modelo ou a performance, isso não será levado em consideração durante o teste. Apenas a gestão do arquivo **.pkl** (Save, load, Predict). E o levantamento das métricas de previsão.

Sinta-se à vontade para criar o modelo da forma que preferir utilizando os campos que preferir.

Não esqueça de:

1. Tratar os valores nulos;
2. Separar os dados em **X** (features) e **Y** (target);
3. Dividir os dados em conjuntos de treinamento e teste;
4. Validar seu modelo no conjunto de testes;
5. Gerar algumas métricas (lembrando que o modelo não precisa performar);
6. Salvar o modelo treinado para utilizá-lo na API posteriormente.

API

No template você irá encontrar uma API feita em FastAPI. Essa API já contém **algumas rotas** que você pode se basear para desenvolver sua aplicação. Além disso existe também uma classe de exemplo para **persistência de dados em memória**. Lembre-se que sua aplicação deverá retornar um histórico de predições realizadas. Essas predições podem ser persistidas em memória ou em um arquivo local em disco.

A persistência do .pkl do modelo (`/model/load/`) poderá ser feita localmente ou em memória.

Não se esqueça de:

- Desenvolver todas as rotas solicitadas no início do documento e os testes que julgar necessário.
- Commitar tudo em um repositório git, pois é com ele que vamos avaliar a entrega
- Entregar a aplicação configurada pra rodar em um container