# API Bootcamp CDIA

## LINKS:

› **API:** http://34.193.187.218:5000
› **Dashboard de suporte:** http://34.193.187.218:8501

## Objetivo

A API foi desenvolvida com o objetivo de **SIMULAR** o uso do modelo em produção com novos dados fora do conjunto de treinamento disponível para avaliação do seu desempenho.

## IMPORTANTE:

- A disponibilidade da API e do dashboard não é garantida. Pode ocorrer indisponibilidade ou travamentos no uso!

- Prefira a utilização da API em vez do dashboard

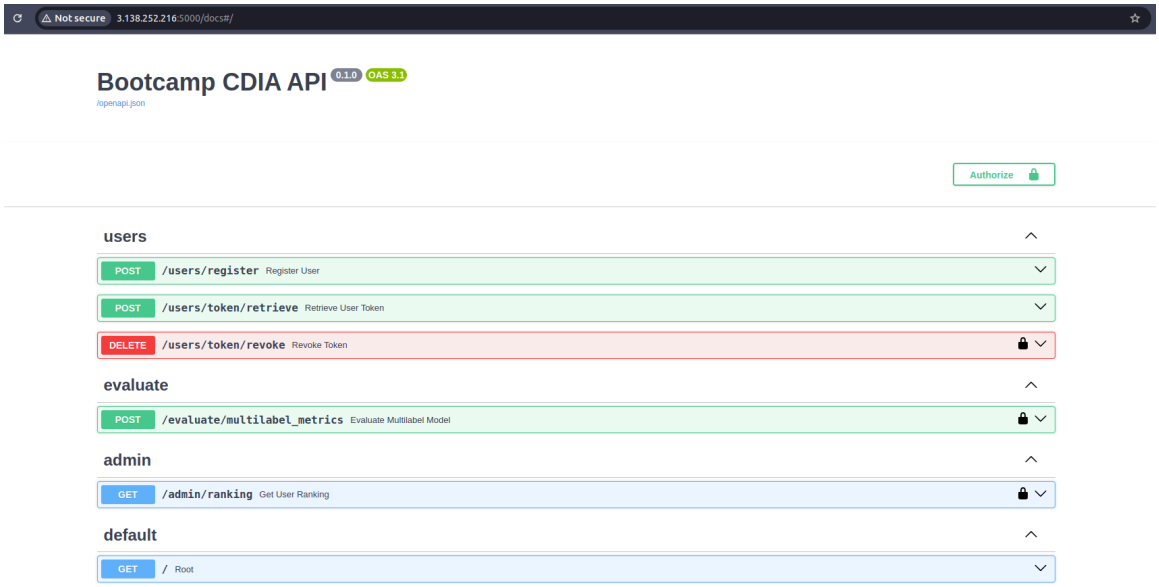
- NÃO É OBRIGATÓRIO realizar o teste do seu modelo usando a API. Porém, seria interessante se conseguirem.

# Instruções de uso

Existem **duas formas** de fazer uso da API de avaliação, a primeira ao utilizá-la diretamente pelos seus endpoints e a segunda forma seria pelo dashboard em Streamlit disponibilizado.

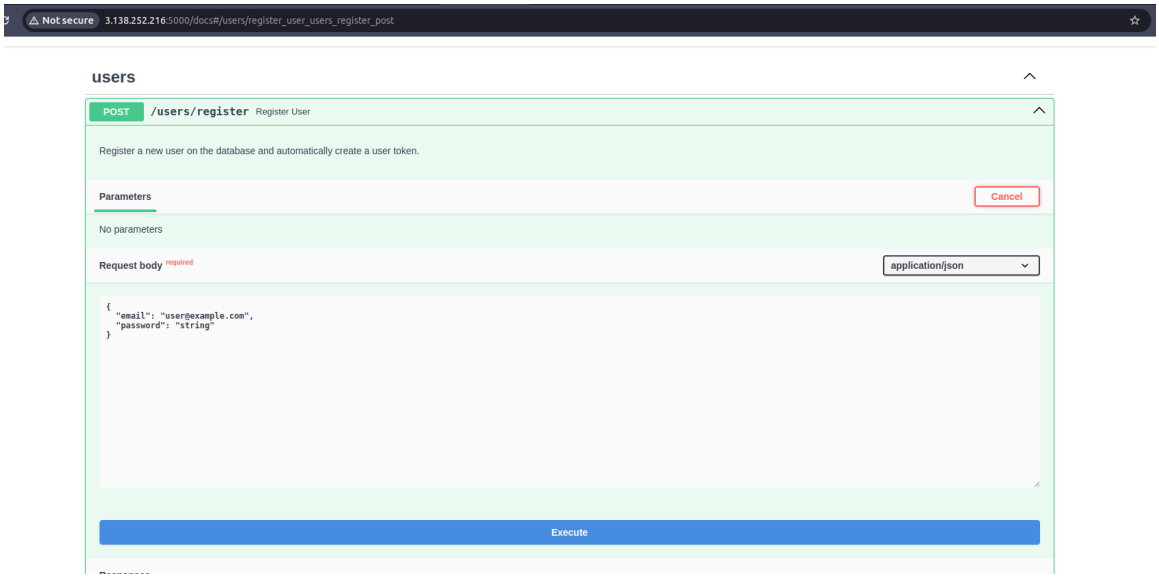
## 1) Utilizando a API diretamente:

Para verificar os endpoints disponíveis e o seu uso, acesse sua documentação do swagger em:
› http://34.193.187.218:5000/docs




No Swagger, no endpoint **/user/register** você deve fazer seu cadastro com um email e senha:

Você pode testar o uso do endpoint diretamente pelo swagger clicando no endpoint e depois em “Try it out”. Altere os campos do Request Body e clique em “Execute”.



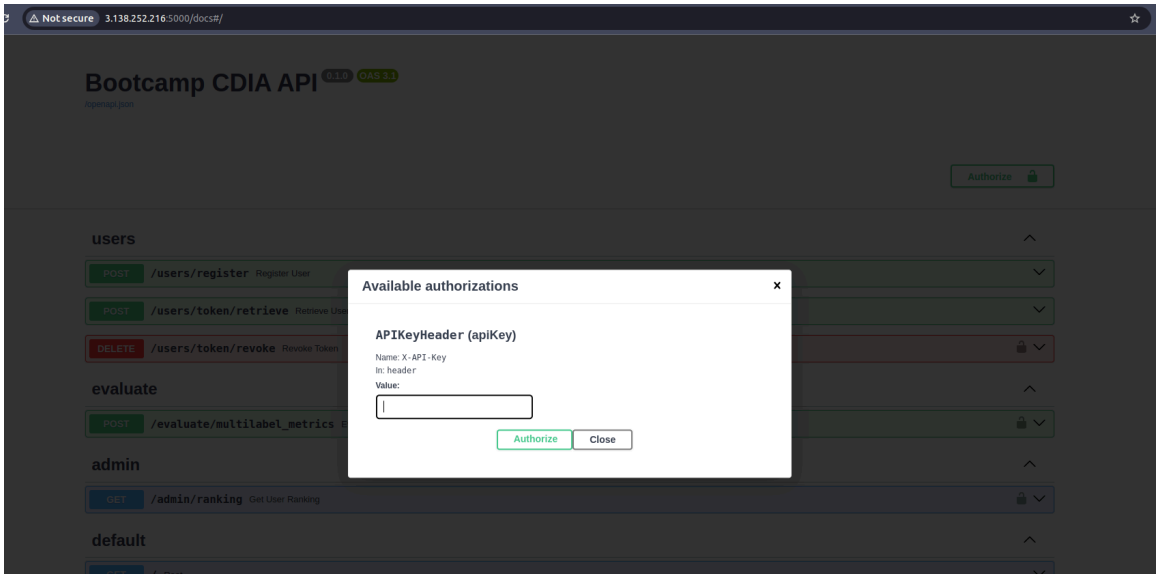

Ou por código em Python usando a biblioteca **requests**:

```
import requests

response = requests.post(
  "http:/34.193.187.218:5000/users/register",
  json={"email": "seu-email@gmail.com", "password": "sua-senha"},
)
```

Se tudo der certo, você deve receber abaixo uma resposta com um **token** gerado. Esse token é a forma de autenticação da API para a utilização do **endpoint /evaluate/multilabel_metrics.**

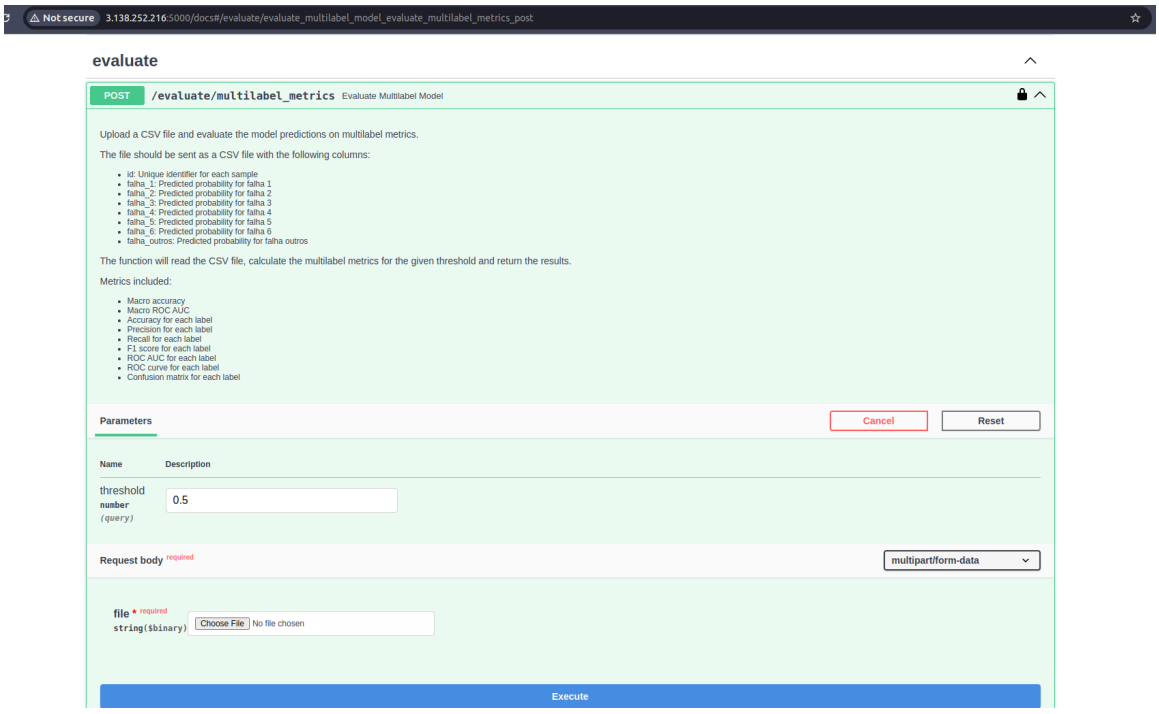
Esse é o principal e único endpoint que vocês devem utilizar para receber as métricas de avaliação das predições dos seus modelos no conjunto “em produção”.

Antes de usá-lo, clique no botão **“Authorize”** no canto superior direito do Swagger e coloque o seu token.

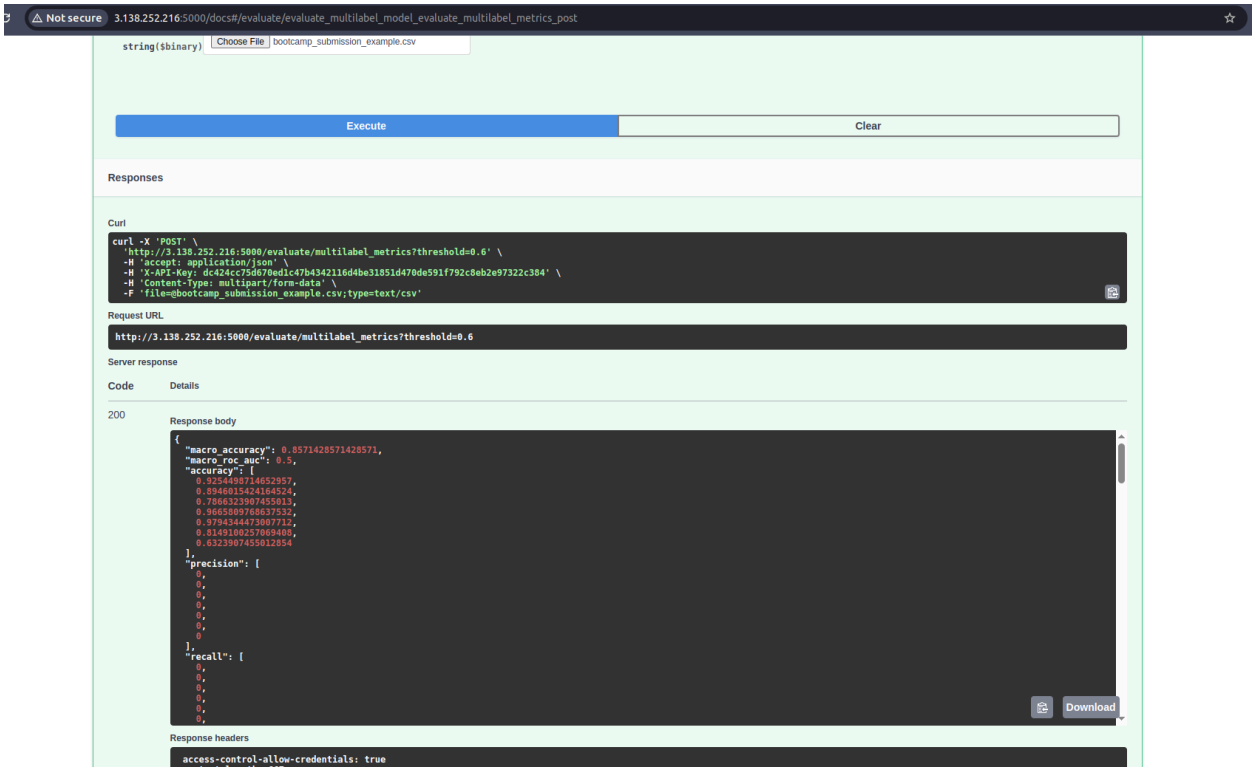



Clique em Authorize e logo após em close.

Vá para o endpoint **/evaluate/multilabel_metrics** e defina nos parâmetros o threshold de decisão do seu modelo e envie o arquivo CSV com as predições no Request Body.



Se o arquivo estiver formatado corretamente conforme o arquivo disponibilizado **“bootcamp_submission_example.csv”,** você deve receber um JSON com as principais métricas de classificação binária para cada uma das classes de falha.




Para fazer o mesmo processo em Python, simplesmente faça uma requisição POST da seguinte forma:

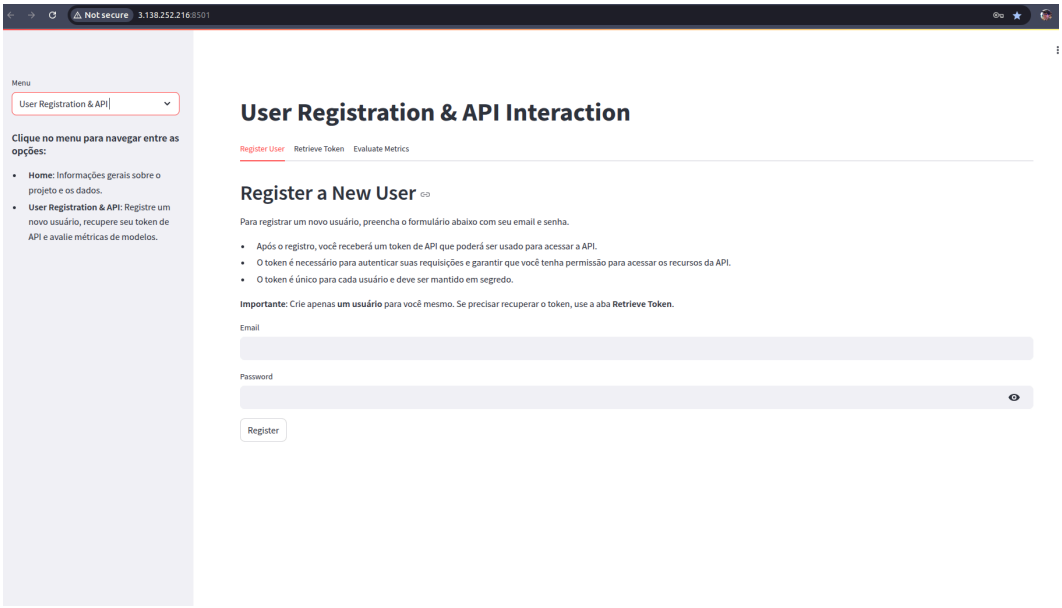
```python
import requests

headers = {{"X-API-Key": "<your_token_here>"}}
files = {{"file": open("path_to_your_file.csv", "rb")}}
params = {{"threshold": 0.5}}
response = requests.post(
  "http:/34.193.187.218:5000//evaluate/multilabel_metrics",
  headers=headers,
  files=files,
  params=params,
)
print(response.json())
```

# 2) Utilizando o dashboard de suporte

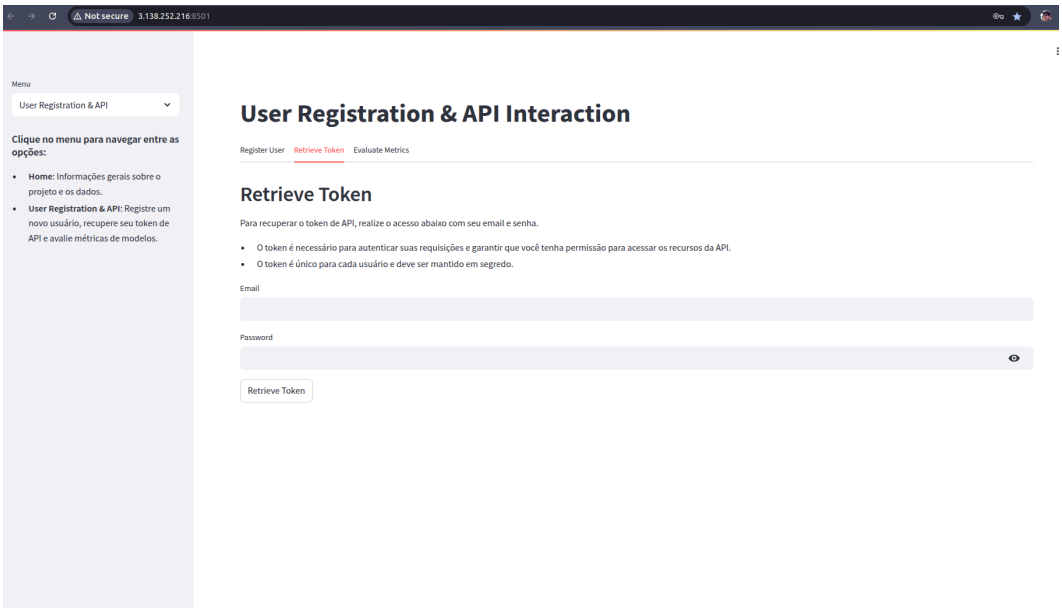
Uma alternativa ao uso da API pelo Swagger ou Python, você pode utilizar o dashboard em streamlit desenvolvido e disponibilizado em: http://34.193.187.218:8501/

A página inicial do dashboard contém a descrição do desafio, e clicando no menu lateral temos acesso à área de cadastro e uso das APIs.


Você pode utilizá-la para cadastro do usuário:

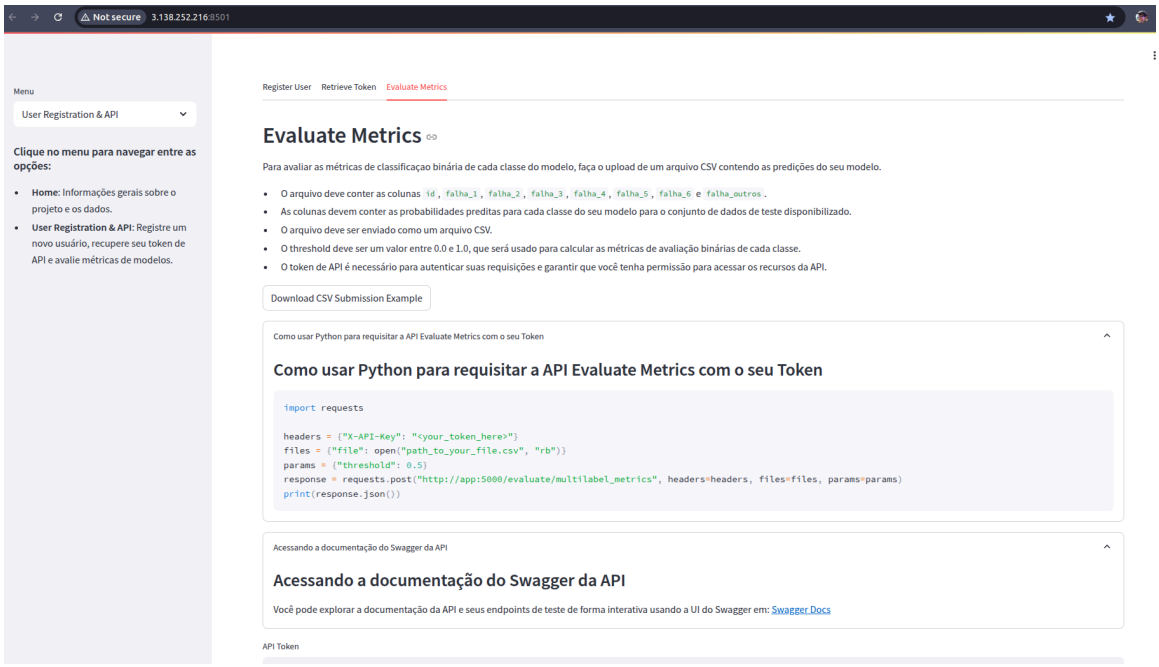



Buscar o token com seu email e senha cadastrados, caso perca o token:

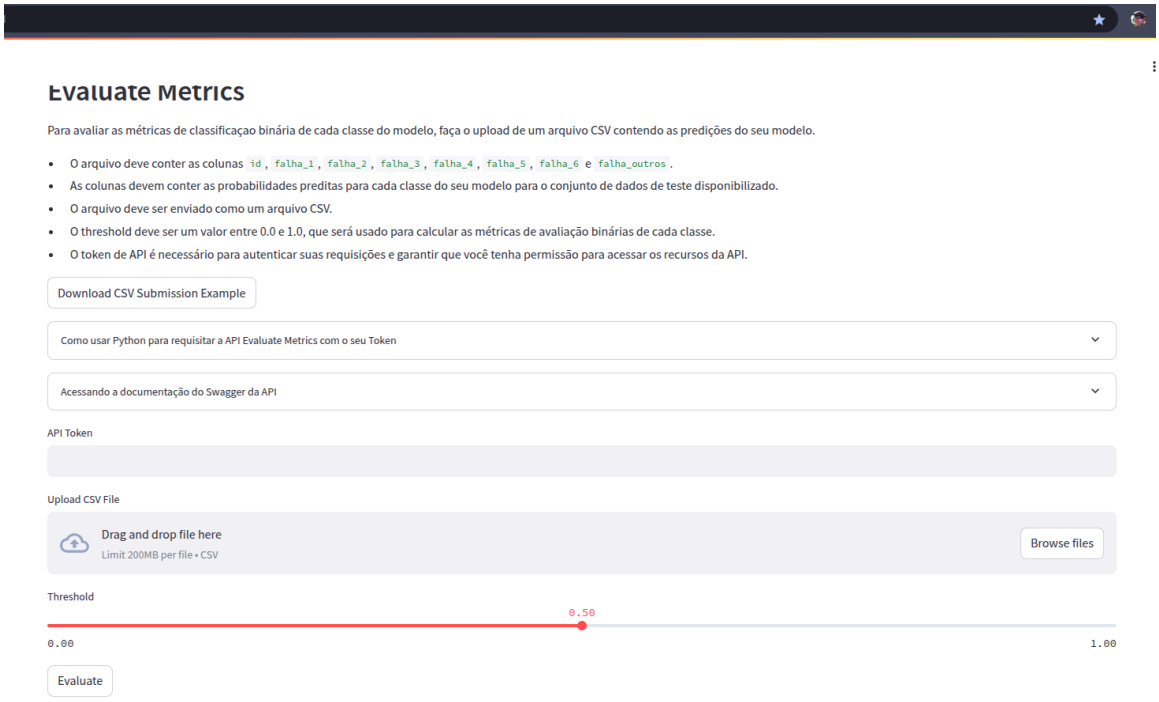



E a última aba para utilizar o endpoint de avaliação das predições:

› Note que cada página possui documentações instruindo como utilizar em Python também.




Coloque o seu token e o arquivo CSV com as predições para cada coluna de falha.




Ao clicar em **“Evaluate”** ficará disponível as métricas em JSON para baixar e também visualizações das métricas abaixo.

Evaluate

## Metrics JSON


```
{
  "macro_accuracy" : 0.8571428571428571
  "macro_roc_auc" : 0.5
  "accuracy" : [...]
  "precision" : [...]
  "recall" : [...]
  "f1_score" : [...]
  "roc_auc" : [...]
  "roc" : [...]
  "confusion_matrix" : [...]
}
```


Download Metrics as JSON

## Metrics Visualization

Overall Metrics

### Overall Metrics

| Macro Accuracy | Macro ROC AUC |
|---|---|
| 85.71% | 0.50 |

Per-Class Metrics

### Per-Class Metrics

#### Accuracy

Per-Class Metrics

### Per-Class Metrics

#### Accuracy

| Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Class 7 |
|---|---|---|---|---|---|---|
| 92.54% | 89.46% | 78.66% | 96.66% | 97.94% | 81.49% | 63.24% |

#### Precision

| Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Class 7 |
|---|---|---|---|---|---|---|
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

#### Recall

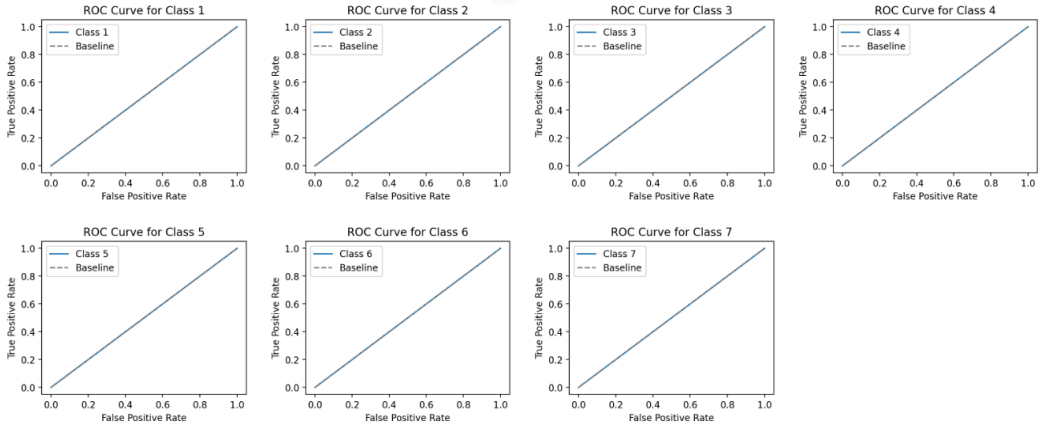
| Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Class 7 |
|---|---|---|---|---|---|---|
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

#### F1 Score

| Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Class 7 |
|---|---|---|---|---|---|---|
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

ROC Curves

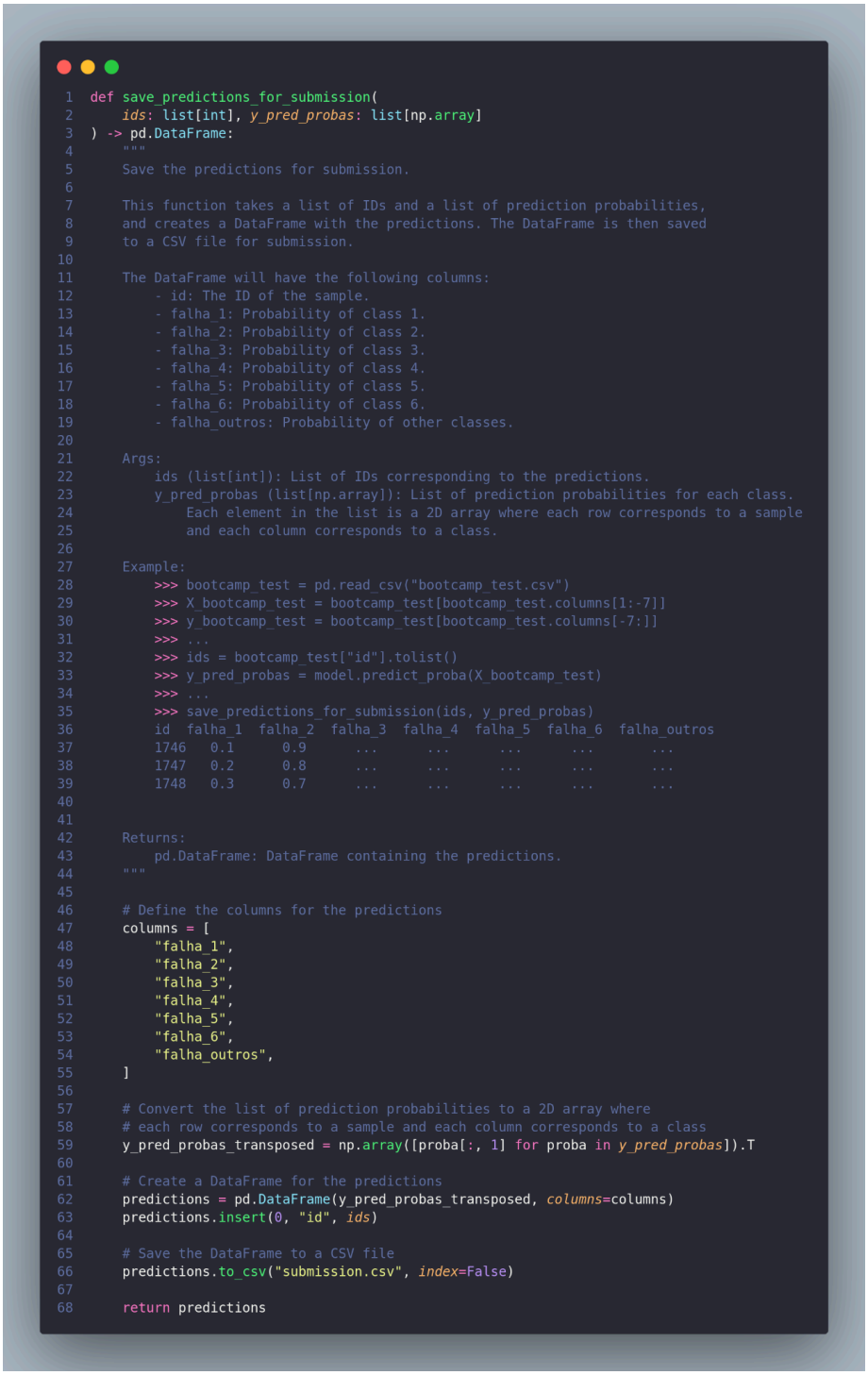
## ROC Curves




Confusion Matrices

## Confusion Matrices

# Anexo: Snippet de código para gerar arquivo CSV com as predições para a API



```python
def save_predictions_for_submission(
    ids: list[int], y_pred_probas: list[np.array]
) -> pd.DataFrame:
    """
    Save the predictions for submission.

    This function takes a list of IDs and a list of prediction probabilities,
    and creates a DataFrame with the predictions. The DataFrame is then saved
    to a CSV file for submission.

    The DataFrame will have the following columns:
        - id: The ID of the sample.
        - falha_1: Probability of class 1.
        - falha_2: Probability of class 2.
        - falha_3: Probability of class 3.
        - falha_4: Probability of class 4.
        - falha_5: Probability of class 5.
        - falha_6: Probability of class 6.
        - falha_outros: Probability of other classes.

    Args:
        ids (list[int]): List of IDs corresponding to the predictions.
        y_pred_probas (list[np.array]): List of prediction probabilities for each class.
            Each element in the list is a 2D array where each row corresponds to a sample
            and each column corresponds to a class.

    Example:
        >>> bootcamp_test = pd.read_csv("bootcamp_test.csv")
        >>> X_bootcamp_test = bootcamp_test[bootcamp_test.columns[1:-7]]
        >>> y_bootcamp_test = bootcamp_test[bootcamp_test.columns[-7:]]
        >>> ...
        >>> ids = bootcamp_test["id"].tolist()
        >>> y_pred_probas = model.predict_proba(X_bootcamp_test)
        >>> ...
        >>> save_predictions_for_submission(ids, y_pred_probas)
        id  falha_1  falha_2  falha_3  falha_4  falha_5  falha_6  falha_outros
        1746   0.1      0.9      ...      ...      ...      ...       ...
        1747   0.2      0.8      ...      ...      ...      ...       ...
        1748   0.3      0.7      ...      ...      ...      ...       ...


    Returns:
        pd.DataFrame: DataFrame containing the predictions.
    """

    # Define the columns for the predictions
    columns = [
        "falha_1",
        "falha_2",
        "falha_3",
        "falha_4",
        "falha_5",
        "falha_6",
        "falha_outros",
    ]

    # Convert the list of prediction probabilities to a 2D array where
    # each row corresponds to a sample and each column corresponds to a class
    y_pred_probas_transposed = np.array([proba[:, 1] for proba in y_pred_probas]).T

    # Create a DataFrame for the predictions
    predictions = pd.DataFrame(y_pred_probas_transposed, columns=columns)
    predictions.insert(0, "id", ids)

    # Save the DataFrame to a CSV file
    predictions.to_csv("submission.csv", index=False)

    return predictions
```

Modificar a variável “columns” com os modos de falha especificados na página de submissão*