

# 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.

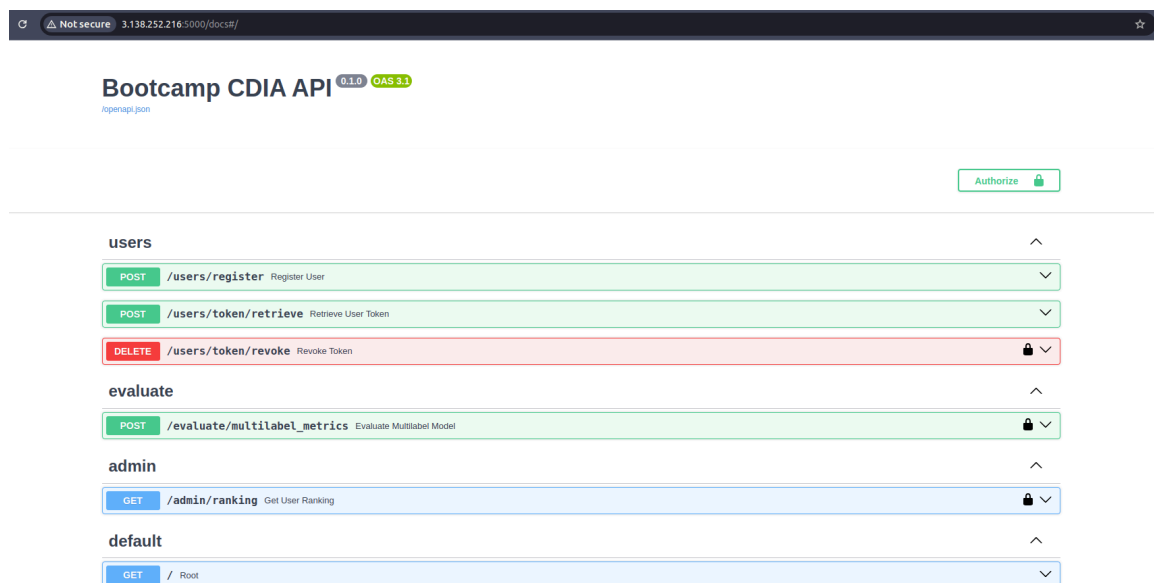
# Instruções de uso

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:

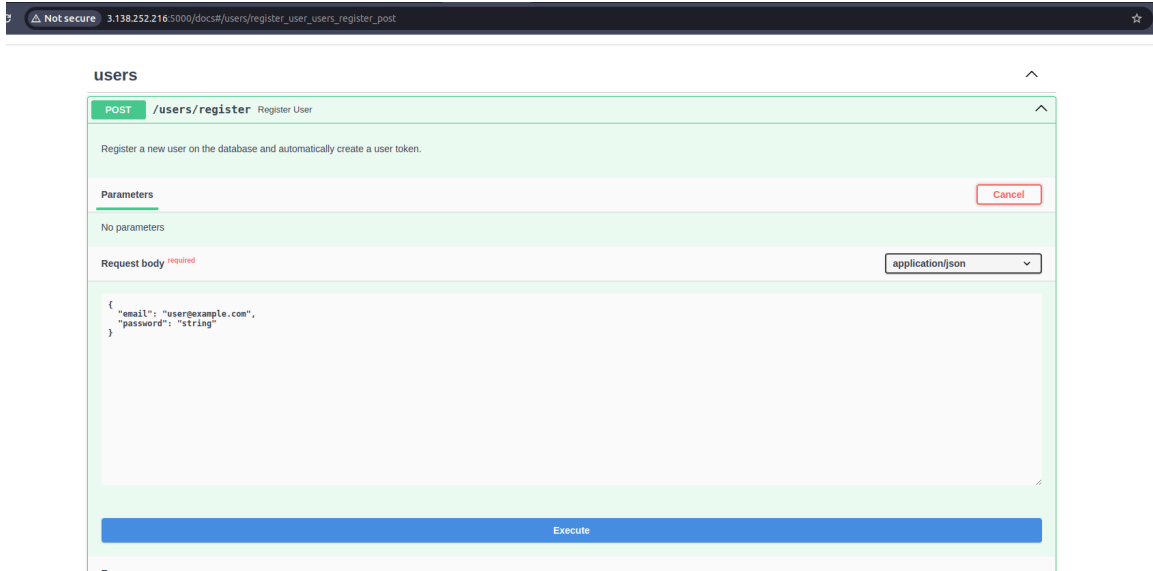
Para verificar os endpoints disponíveis e o seu uso, acesse sua documentação do swagger em:

> <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**:

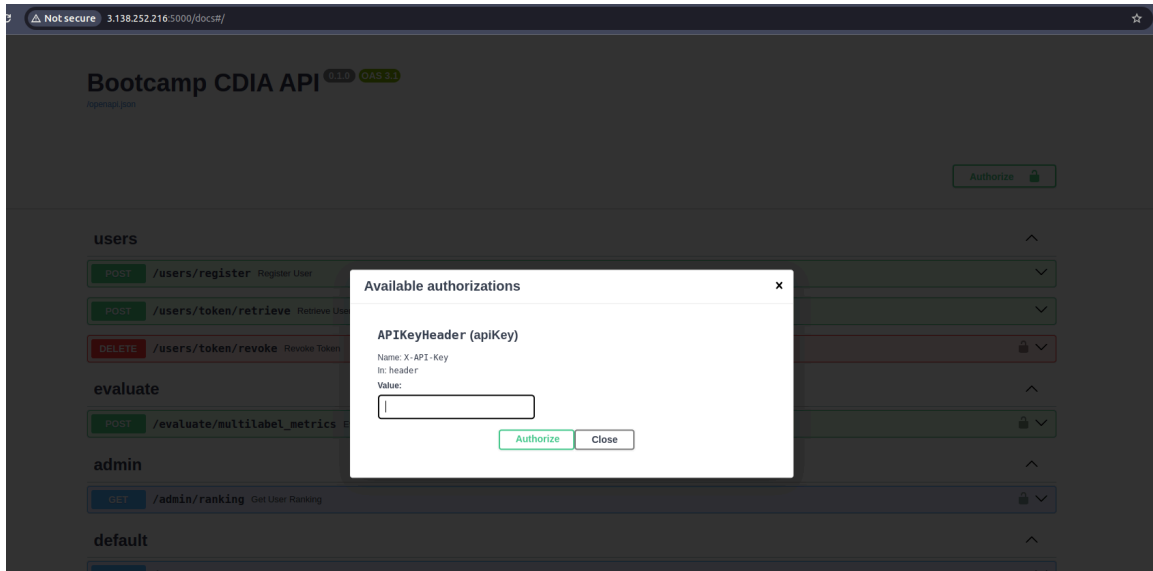
```
import requests

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**.

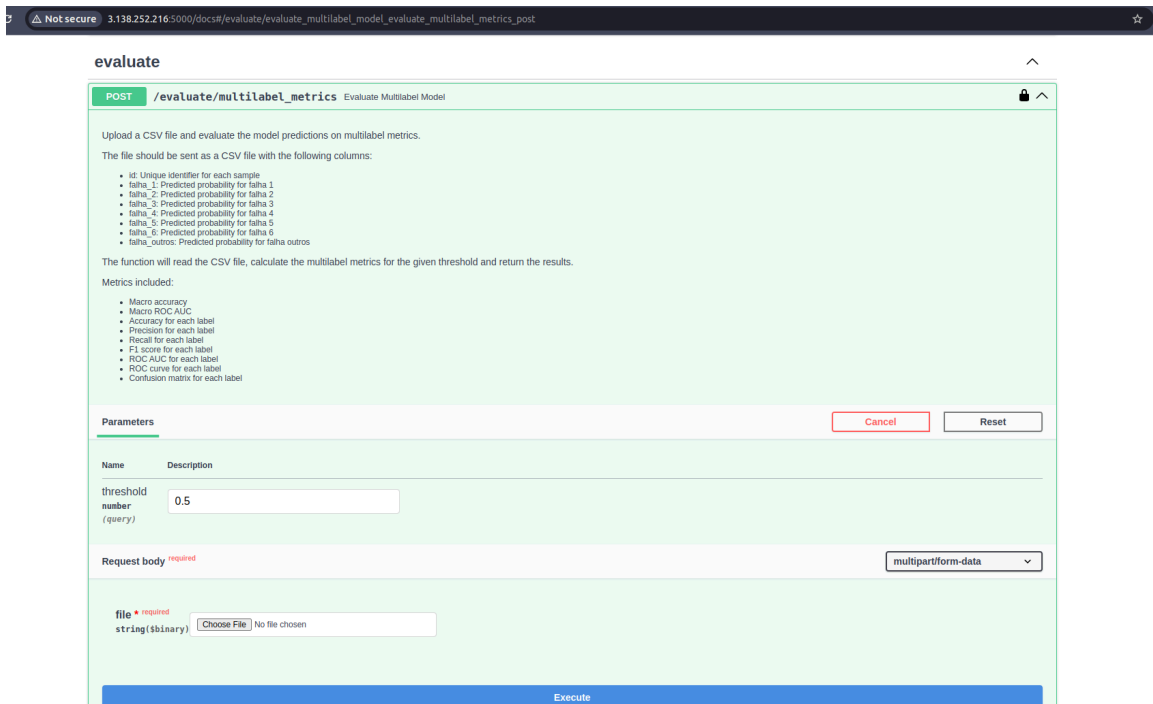
Esse é o principal e único endpoint que vocês devem utilizar para receber as métricas de avaliação das predições dos seus modelos no conjunto “em produção”.

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

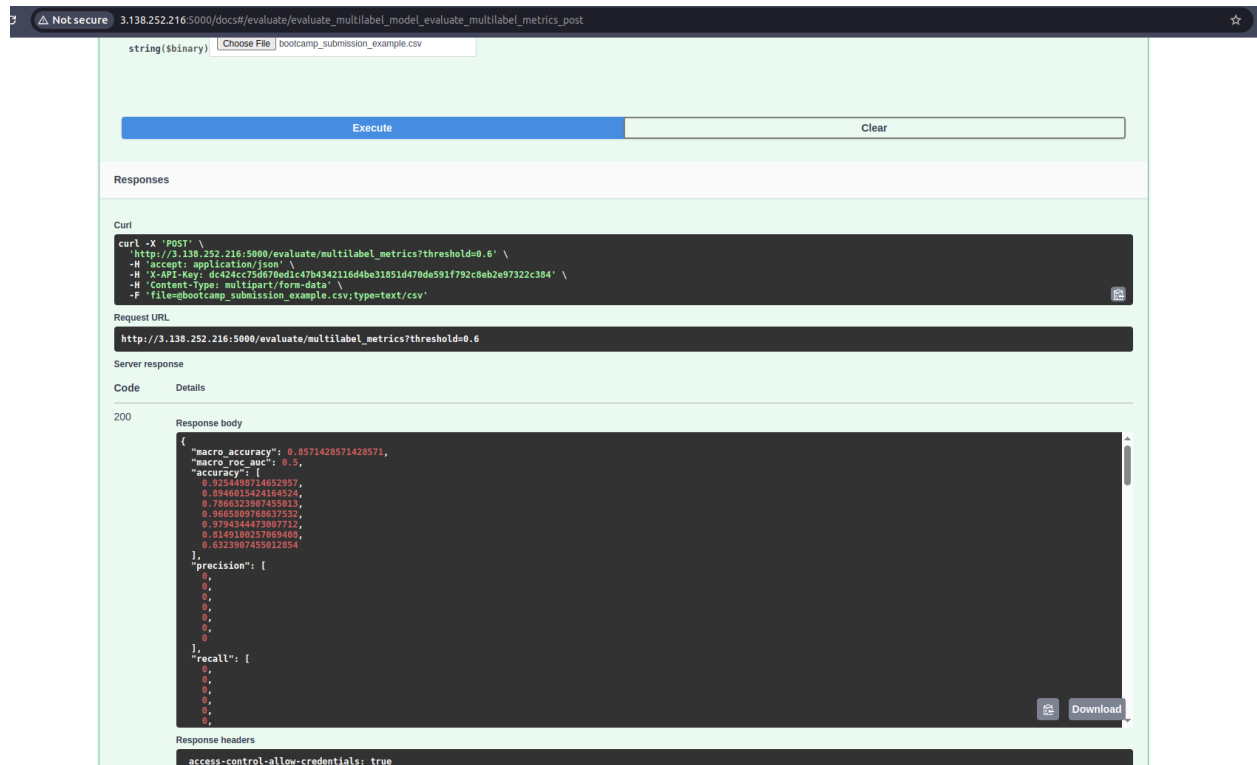


Clique em Authorize e logo após em close.

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:

```
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.



**Bootcamp Ciência de Dados e Inteligência Artificial**

### Contextualização

Uma empresa do setor siderúrgico contratou você para a criação de um sistema inteligente de controle de qualidade para chapas de aço inoxidável. Essa empresa forneceu um conjunto de dados contendo informações extraídas a partir de imagens de superfície das chapas, com o objetivo de detectar e classificar defeitos automaticamente. Cada amostra no conjunto de dados é composta por 29 indicadores que descrevem aspectos geométricos e estatísticos do defeito identificado, como área, perímetro, índices de orientação, luminosidade e proporção de bordas. Além dessas características, cada amostra é rotulada com uma das sete possíveis classes de defeitos (sete categorias específicas e uma categoria genérica de "outros defeitos").

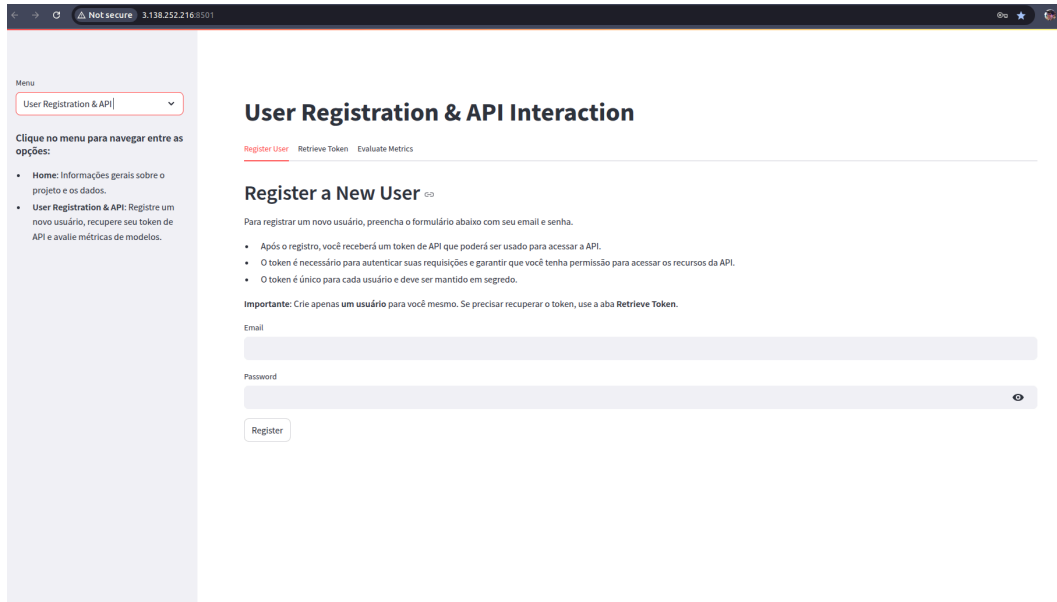
O sistema deverá ser capaz de, a partir do cadastro de uma nova imagem (ou conjunto de medições), **prever a classe do defeito e retornar a probabilidade associada**. Além disso, a empresa espera que você extraia insights da operação e dos defeitos e gere visualizações de dados.

### Descrição dos Dados

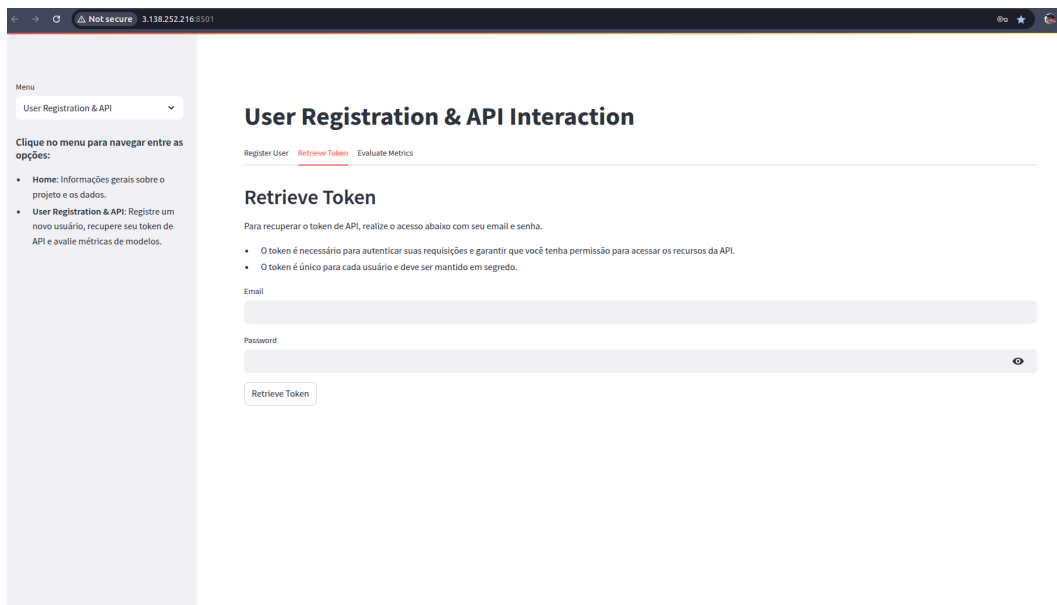
Serão disponibilizados dois arquivos de dados:

- `Bootcamp_train.csv`: use-o para explorar, treinar, avaliar seus modelos.

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.

Menu

User Registration & API

Clique no menu para navegar entre as opções:

- Home: Informações gerais sobre o projeto e os dados.
- User Registration & API: Registre um novo usuário, recupere seu token de API e avalie métricas de modelos.

Register UserRetrieve TokenEvaluate Metrics

## Evaluate Metrics

Para avaliar as métricas de classificação binária de cada classe do modelo, faça o upload de um arquivo CSV contendo as predições do seu modelo.

- O arquivo deve conter as colunas `id`, `falha_1`, `falha_2`, `falha_3`, `falha_4`, `falha_5`, `falha_6` e `falha_outros`.
- As colunas devem conter as probabilidades preditas para cada classe do seu modelo para o conjunto de dados de teste disponibilizado.
- O arquivo deve ser enviado como um arquivo CSV.
- O threshold deve ser um valor entre 0.0 e 1.0, que será usado para calcular as métricas de avaliação binárias de cada classe.
- O token de API é necessário para autenticar suas requisições e garantir que você tenha permissão para acessar os recursos da API.

Download CSV Submission Example

Como usar Python para requisitar a API Evaluate Metrics com o seu Token

```
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://app:5000/evaluate/multiLabelMetrics", headers=headers, files=files, params=params)
print(response.json())
```

Acessando a documentação do Swagger da API

Você pode explorar a documentação da API e seus endpoints de teste de forma interativa usando a UI do Swagger em: [Swagger Docs](#)

API Token

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

## Evaluate Metrics

Para avaliar as métricas de classificação binária de cada classe do modelo, faça o upload de um arquivo CSV contendo as predições do seu modelo.

- O arquivo deve conter as colunas `id`, `falha_1`, `falha_2`, `falha_3`, `falha_4`, `falha_5`, `falha_6` e `falha_outros`.
- As colunas devem conter as probabilidades preditas para cada classe do seu modelo para o conjunto de dados de teste disponibilizado.
- O arquivo deve ser enviado como um arquivo CSV.
- O threshold deve ser um valor entre 0.0 e 1.0, que será usado para calcular as métricas de avaliação binárias de cada classe.
- O token de API é necessário para autenticar suas requisições e garantir que você tenha permissão para acessar os recursos da API.

Download CSV Submission Example

Como usar Python para requisitar a API Evaluate Metrics com o seu Token

Acessando a documentação do Swagger da API

API Token

Upload CSV File

Drag and drop file here

Limit 200MB per file • CSV

Browse files

Threshold

0.00

0.50

1.00

Evaluate

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

85.71%

Macro ROC AUC

0.50

### Per-Class Metrics

### Per-Class Metrics

### Accuracy

### Per-Class Metrics

### Per-Class Metrics

### Accuracy

Class 1  
92.54%

Class 2

89.46%

Class 3

78.66%

Class 4  
96.66%

Class 5  
97.94%

Class 6  
81.49%

Class 7  
63.24%

## Precision

Class 1  
0.00

Class 2  
0.00

Class 3  
0.00

Class 4  
0.00

Class 5  
0.00

Class 6  
0.00

Class 7

0.00

### Recall

Class 1  
0.00

Class 2  
0.00

Class 3  
0.00

Class 4  
0.00

Class 5  
0.00

Class 6  
0.00

Class 7

0.00

### F1 Score

Class 1  
0.00

Class 2  
0.00

Class 3  
0.00

Class 4  
0.00

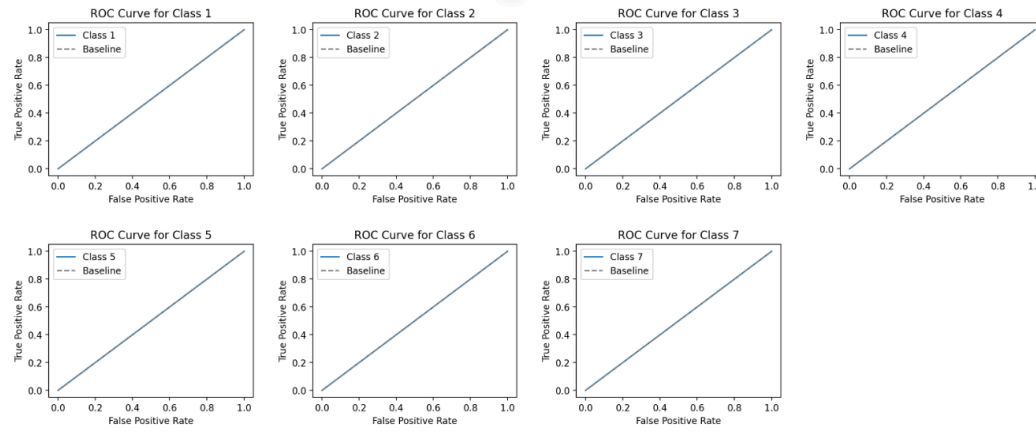
Class 5  
0.00

Class 6  
0.00

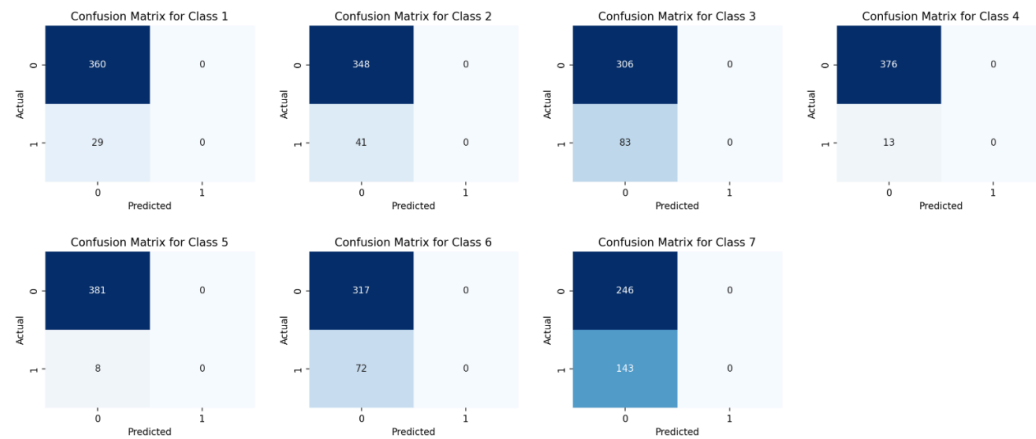
Class 7

0.00

## ROC Curves



## Confusion Matrices



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

```
1 def save_predictions_for_submission(  
2     ids: list[int], y_pred_proba: list[np.array]  
3 ) -> pd.DataFrame:  
4     """  
5     Save the predictions for submission.  
6  
7     This function takes a list of IDs and a list of prediction probabilities,  
8     and creates a DataFrame with the predictions. The DataFrame is then saved  
9     to a CSV file for submission.  
10  
11     The DataFrame will have the following columns:  
12     - id: The ID of the sample.  
13     - falha_1: Probability of class 1.  
14     - falha_2: Probability of class 2.  
15     - falha_3: Probability of class 3.  
16     - falha_4: Probability of class 4.  
17     - falha_5: Probability of class 5.  
18     - falha_6: Probability of class 6.  
19     - falha_outros: Probability of other classes.  
20  
21     Args:  
22     ids (list[int]): List of IDs corresponding to the predictions.  
23     y_pred_proba (list[np.array]): List of prediction probabilities for each class.  
24         Each element in the list is a 2D array where each row corresponds to a sample  
25         and each column corresponds to a class.  
26  
27     Example:  
28     >>> bootcamp_test = pd.read_csv("bootcamp_test.csv")  
29     >>> X_bootcamp_test = bootcamp_test[bootcamp_test.columns[1:-7]]  
30     >>> y_bootcamp_test = bootcamp_test[bootcamp_test.columns[-7:]]  
31     >>> ...  
32     >>> ids = bootcamp_test["id"].tolist()  
33     >>> y_pred_proba = model.predict_proba(X_bootcamp_test)  
34     >>> ...  
35     >>> save_predictions_for_submission(ids, y_pred_proba)  
36     id falha_1 falha_2 falha_3 falha_4 falha_5 falha_6 falha_outros  
37     1746 0.1 0.9 ... ... ... ...  
38     1747 0.2 0.8 ... ... ... ...  
39     1748 0.3 0.7 ... ... ... ...  
40  
41     Returns:  
42     pd.DataFrame: DataFrame containing the predictions.  
43     """  
44  
45     # Define the columns for the predictions  
46     columns = [  
47         "falha_1",  
48         "falha_2",  
49         "falha_3",  
50         "falha_4",  
51         "falha_5",  
52         "falha_6",  
53         "falha_outros",  
54     ]  
55  
56     # Convert the list of prediction probabilities to a 2D array where  
57     # each row corresponds to a sample and each column corresponds to a class  
58     y_pred_proba_transposed = np.array([proba[:, 1] for proba in y_pred_proba]).T  
59  
60     # Create a DataFrame for the predictions  
61     predictions = pd.DataFrame(y_pred_proba_transposed, columns=columns)  
62     predictions.insert(0, "id", ids)  
63  
64     # Save the DataFrame to a CSV file  
65     predictions.to_csv("submission.csv", index=False)  
66  
67     return predictions
```

Modificar a variável "columns" com os modos de falha especificados na página de submissão\*