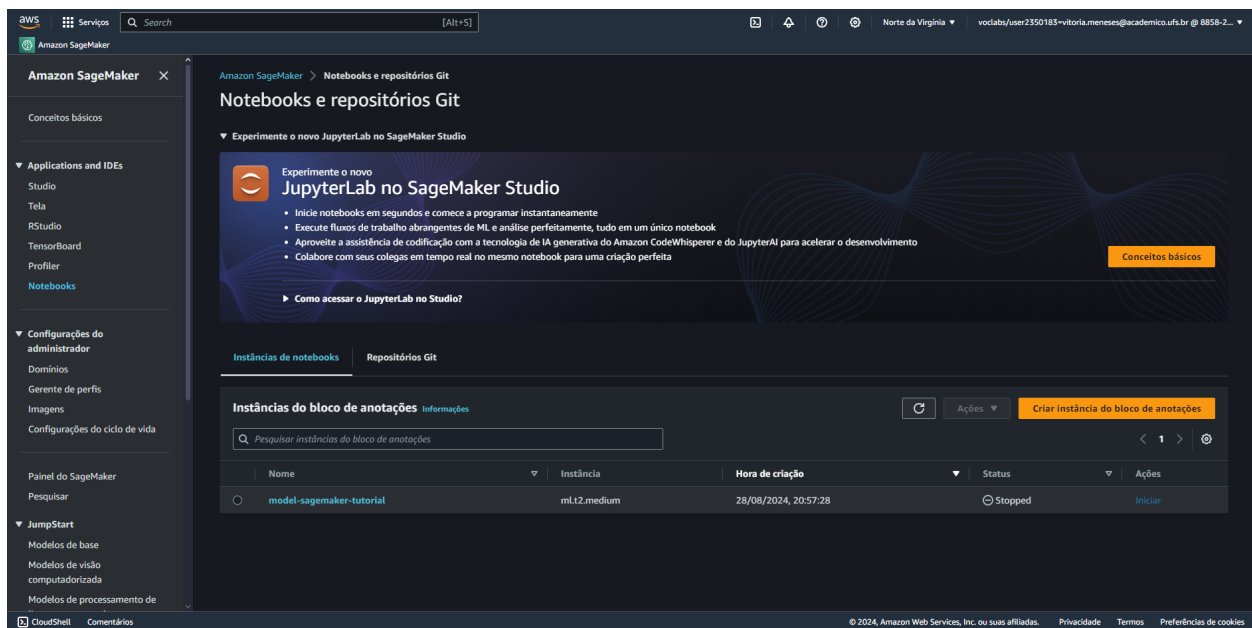


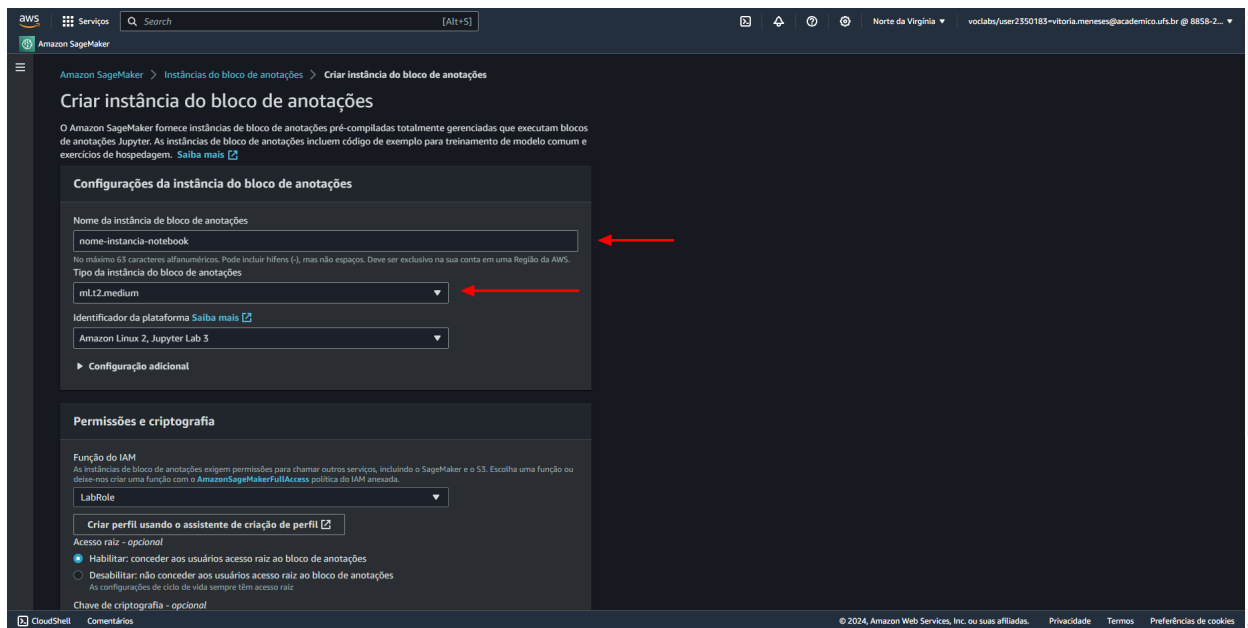
# Amazon sagemaker

Abrindo o console da AWS, iremos digitar na barra de busca 'Amazon Sagemaker', e após isso, na aba 'Notebooks' presente na barra lateral da tela:

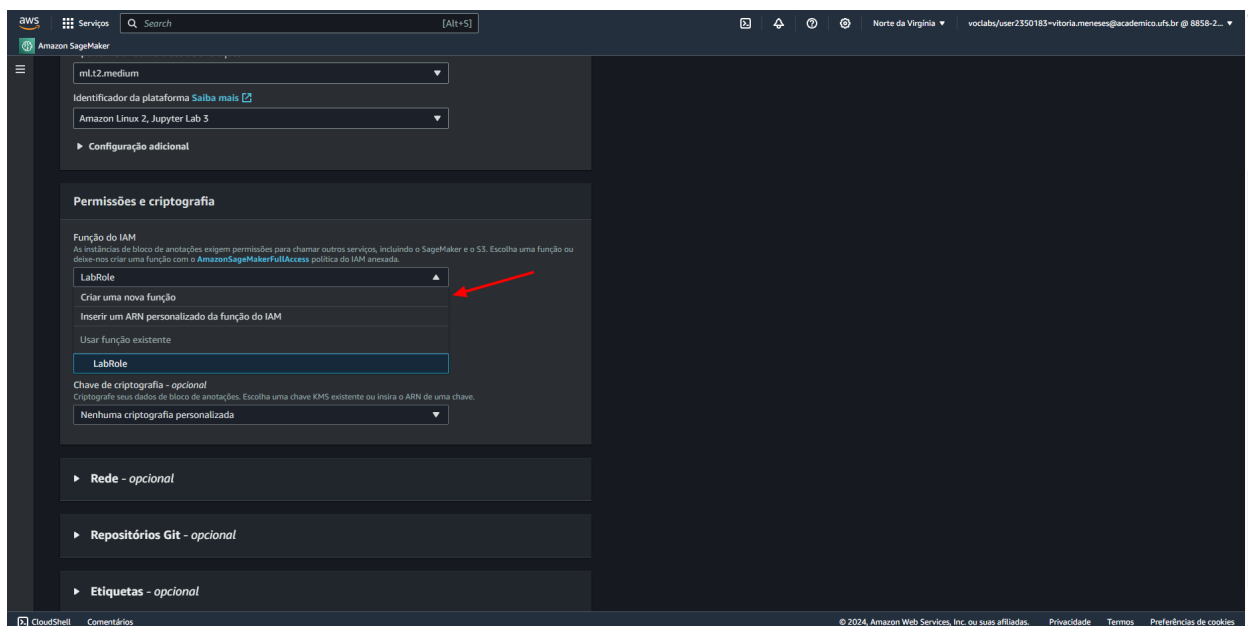


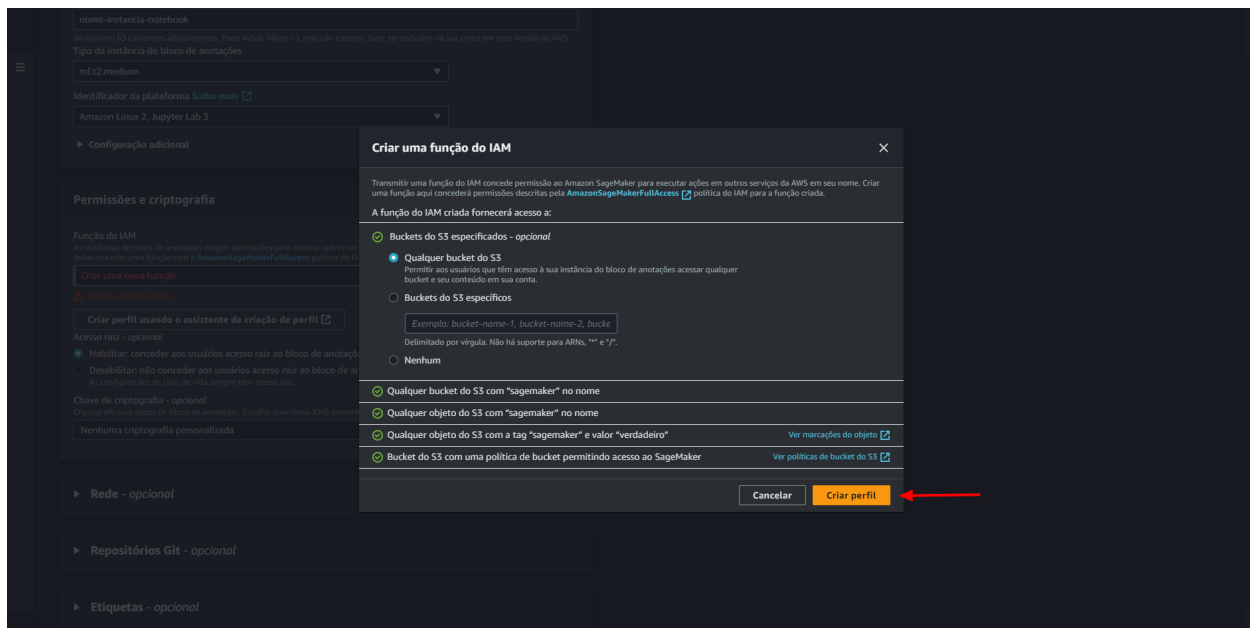
Ao clicar no botão "Criar nova instância do bloco de anotações, um novo projeto com um arquivo Jupyter Notebook será criado, com as configurações a seguir:

- Nome da instancia: nome de sua preferência, considerando as restrições abaixo do campo de input de nome.
- Tipo da instância: para este caso de estudo, será utilizada ml.t2.medium

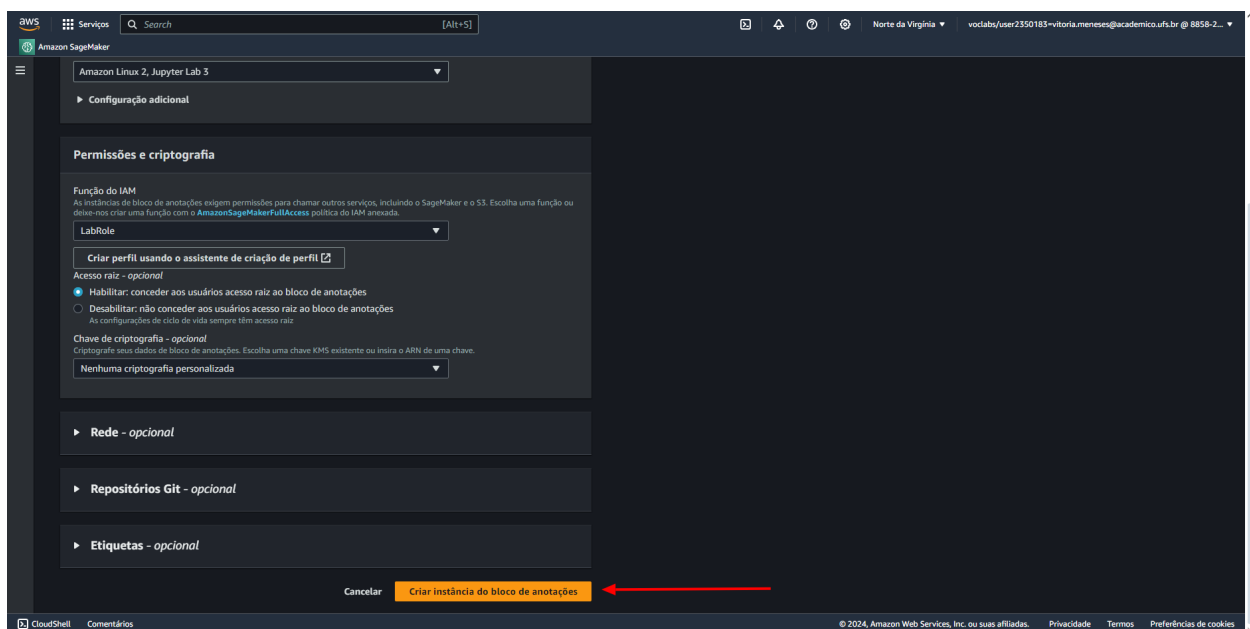


- Função do IAM: clicando no dropdown 'LabRole' e depois em 'Criar Nova Função', será aberta uma aba para as especificações da nova função que está sendo criada, juntamente com o perfil associado a ela:

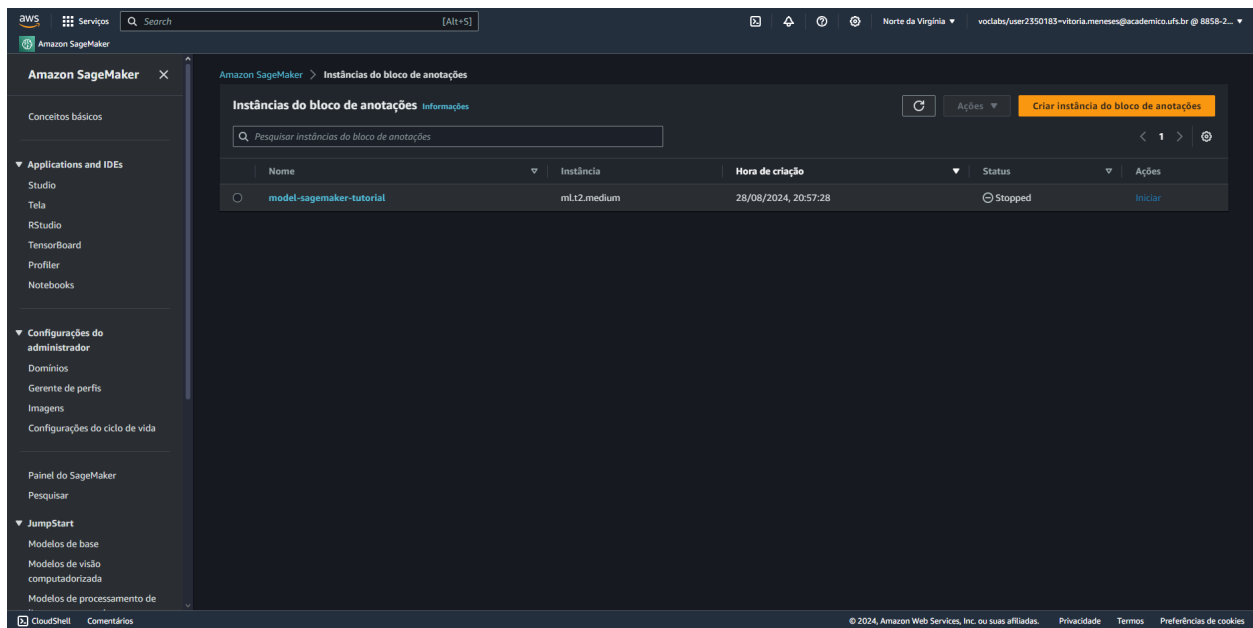




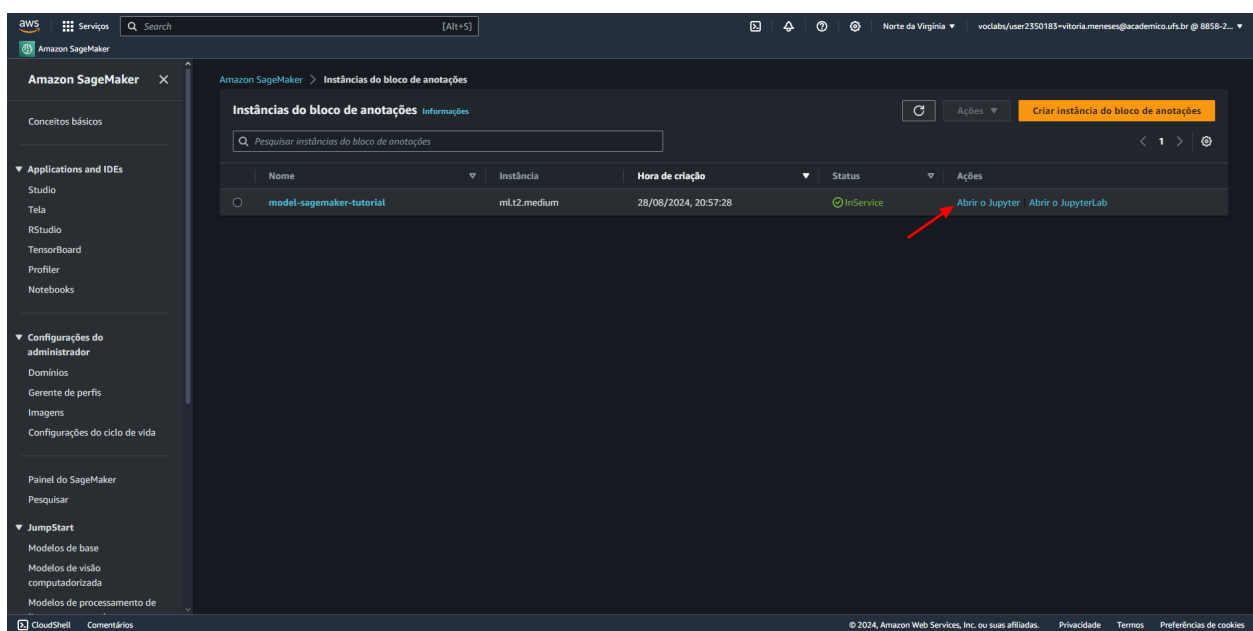
Por fim, clicamos em 'Criar instância do bloco de anotações':



Com isso, somos redirecionados para a página de instâncias do bloco de anotações, com a nova instância adicionada:



Após criada, a instância leva alguns minutos para poder ser iniciada, esse processo pode ser acompanhado na coluna chamada 'Status'. Quando o status da instância mudar para 'InService', clicamos em abrir Jupyter para termos acesso ao projeto com o Notebook inicial criado.





Abrindo o arquivo .ipynb, iremos utilizar o algoritmo XGBoost e o dataset Iris, para classificação de flores Iris em 3 espécies diferentes. Para isso, iremos seguir os passos padrões do sagemaker para o tratamento de modelos: preparação dos dados, treinamento do modelo e deploy desse modelo.

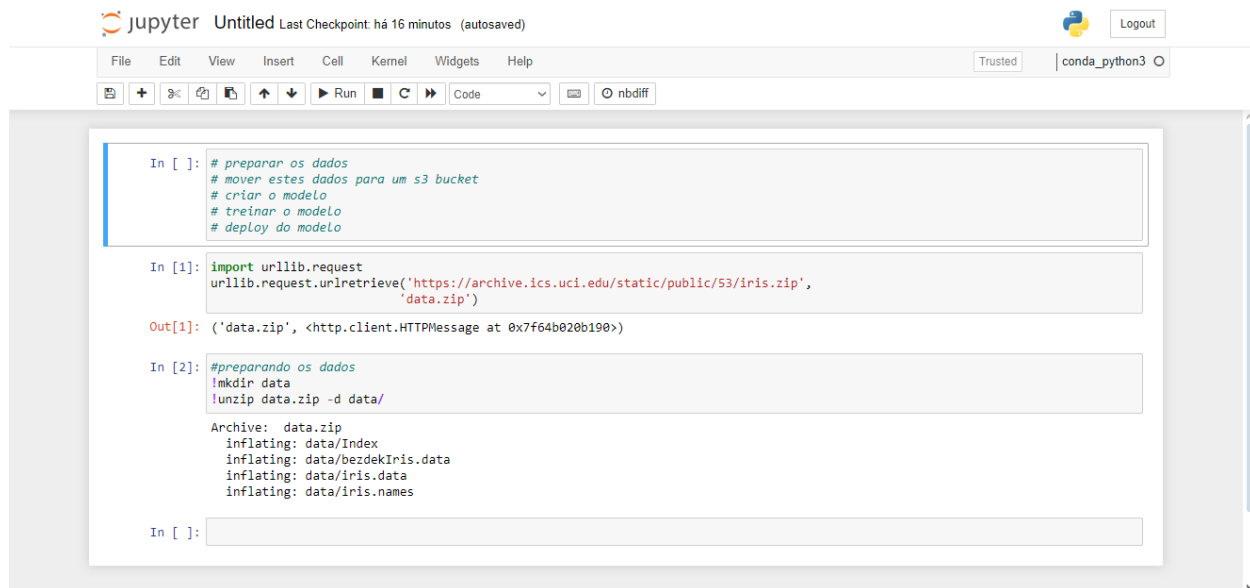
## 1. Preparação

Antes de começar a treinar um modelo de ML, é importante preparar os dados que vão ser utilizados. Nesse sentido, o SageMaker disponibiliza o acesso a ferramentas que permitem o **armazenamento e o tratamento de dados**. Como, por exemplo, Jupyter Notebooks, que será a tecnologia utilizada

Além disso, por meio da plataforma, é possível também usar o Amazon S3 para fazer o armazenamento em nuvem tanto de dados brutos como limpos, sem se preocupar com limites de tamanho.

Para começarmos a preparar os dados, primeiro precisamos extrair-los da pasta zip do link: <https://archive.ics.edu/static/public/53/iris.zip> para o diretório no

projeto do Notebook, que no caso foi criado com o nome 'data'.



The screenshot shows a Jupyter Notebook interface with the title 'Untitled' and a status bar indicating 'Last Checkpoint: há 16 minutos (autosaved)'. The interface includes a top menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for file operations, running, and saving. The notebook content area displays three code cells. The first cell contains a list of tasks in Portuguese: '# preparar os dados', '# mover estes dados para um s3 bucket', '# criar o modelo', '# treinar o modelo', and '# deploy do modelo'. The second cell contains a Python code snippet that imports 'urllib.request' and uses 'urllib.request.urlretrieve' to download a file from 'https://archive.ics.uci.edu/static/public/53/iris.zip' to 'data.zip'. The output of this cell shows the file path and a confirmation message. The third cell contains a shell command to create a directory 'data' and unzip the file: '!mkdir data' and '!unzip data.zip -d data/'. The output of this cell shows the file being unzipped and the contents of the 'data' directory: 'Archive: data.zip', 'inflating: data/Index', 'inflating: data/bezdekIris.data', 'inflating: data/iris.data', and 'inflating: data/iris.names'. The notebook is running on a 'conda\_python3' kernel.

```
In [ ]: # preparar os dados
# mover estes dados para um s3 bucket
# criar o modelo
# treinar o modelo
# deploy do modelo

In [1]: import urllib.request
urllib.request.urlretrieve('https://archive.ics.uci.edu/static/public/53/iris.zip',
'data.zip')

Out[1]: ('data.zip', <http.client.HTTPMessage at 0x7f64b020b190>)

In [2]: #preparando os dados
!mkdir data
!unzip data.zip -d data/

Archive: data.zip
  inflating: data/Index
  inflating: data/bezdekIris.data
  inflating: data/iris.data
  inflating: data/iris.names

In [ ]:
```

Após a extração dos dados para o diretório 'data', os dados começam a ser preparados utilizando a biblioteca pandas, que irá ler o arquivo csv que foi movido para 'data'. Em seguida, as três espécies de plantas do dataset são enumeradas, para facilitar a categorização com a utilização do algoritmo.

Também já são estabelecidos os dados que serão utilizados para o treinamento do modelo e os dados para a validação desse modelo.

```
jupyter CN Last Checkpoint: há uma hora (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3
Archive: data.zip
  inflating: data/Index
  inflating: data/bezdekIris.data
  inflating: data/iris.data
  inflating: data/iris.names

In [11]: #passo a passo da preparação

import pandas as pd

#ler os dados
data = pd.read_csv('data/iris.data', header=None)

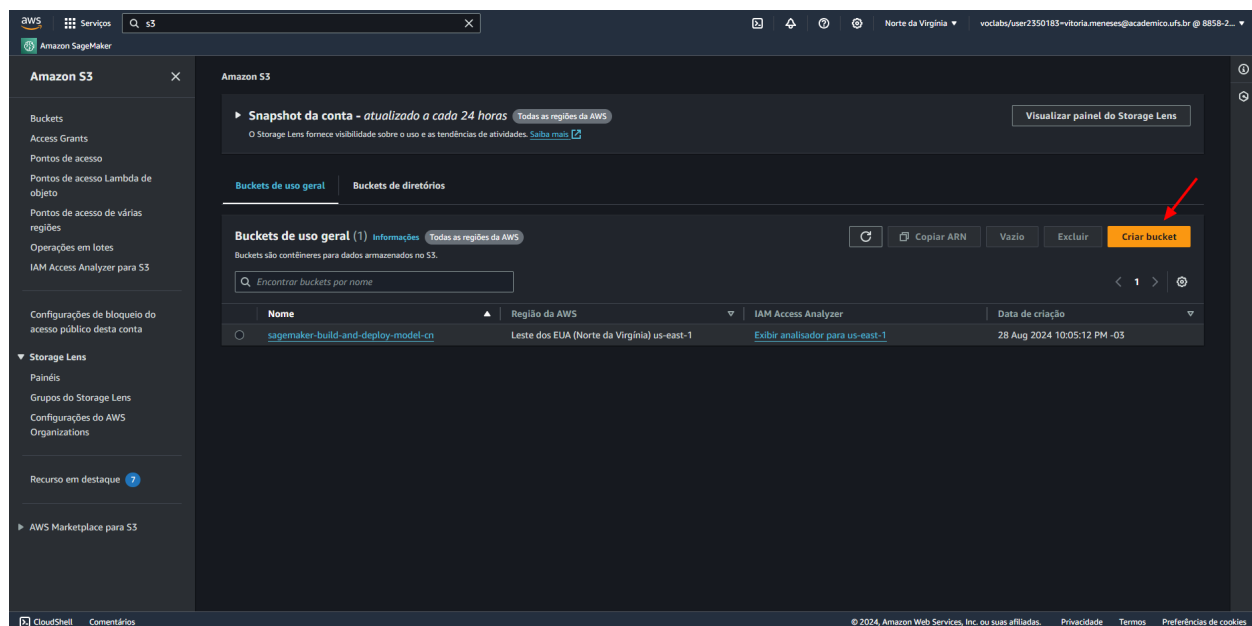
#converter para valores numericos
data[4] = data[4].replace('Iris-setosa', 0)
data[4] = data[4].replace('Iris-virginica', 1)
data[4] = data[4].replace('Iris-versicolor', 2)

#shuffle
data = data.sample(frac=1).reset_index(drop=True)

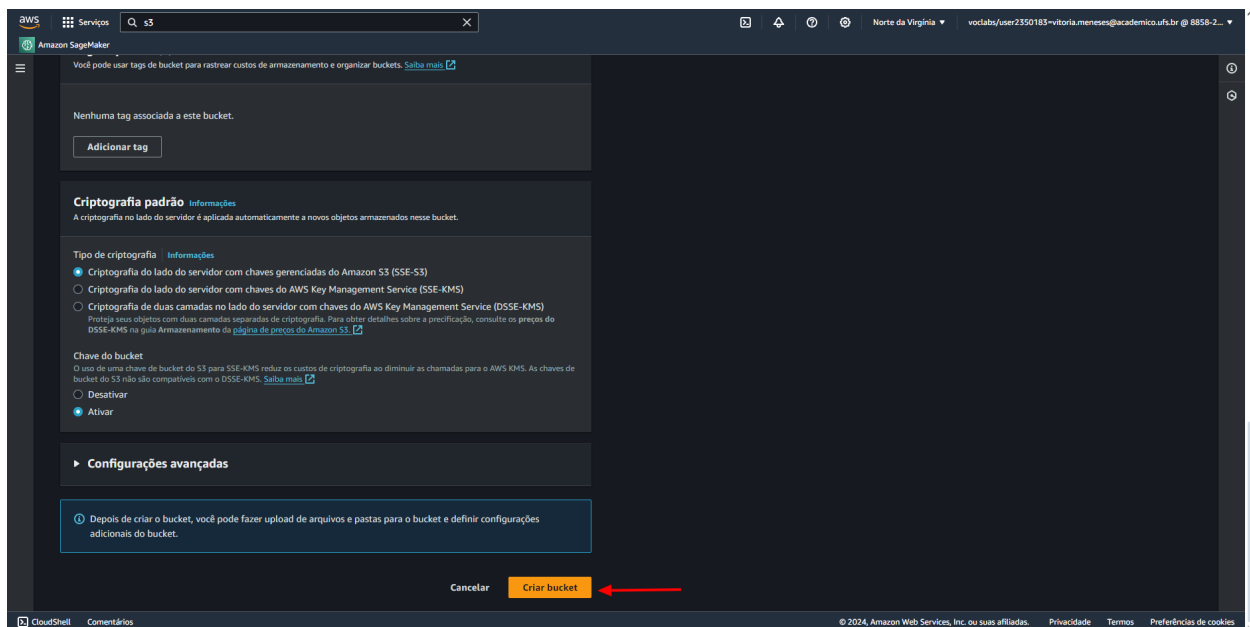
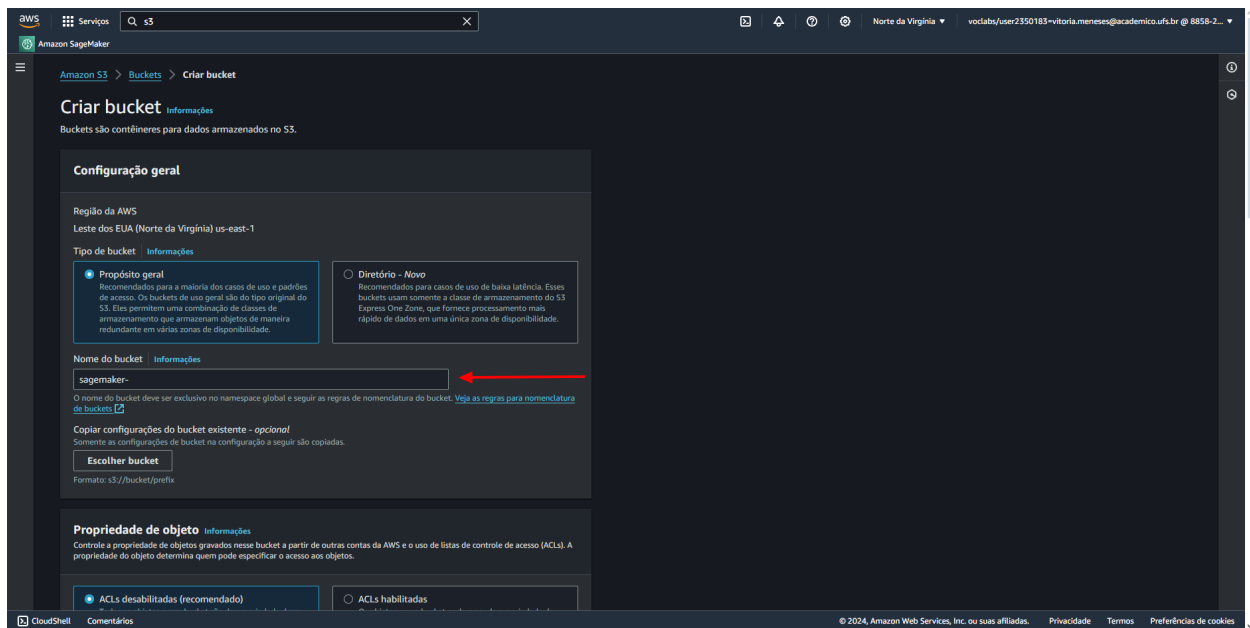
#modificar o rótulo da coluna de indice
data = data[[4, 0, 1, 2, 3]] #definir a coluna de categorias como a primeira do dataset

#dividir os dados em um conjunto de treinamento e um conjunto de validação
#possui 150 instancias
#80% será para treinamento e 20% para validação
train_data = data[:120] #até o indice 120
val_data = data[120:] #a partir do indice 120
```

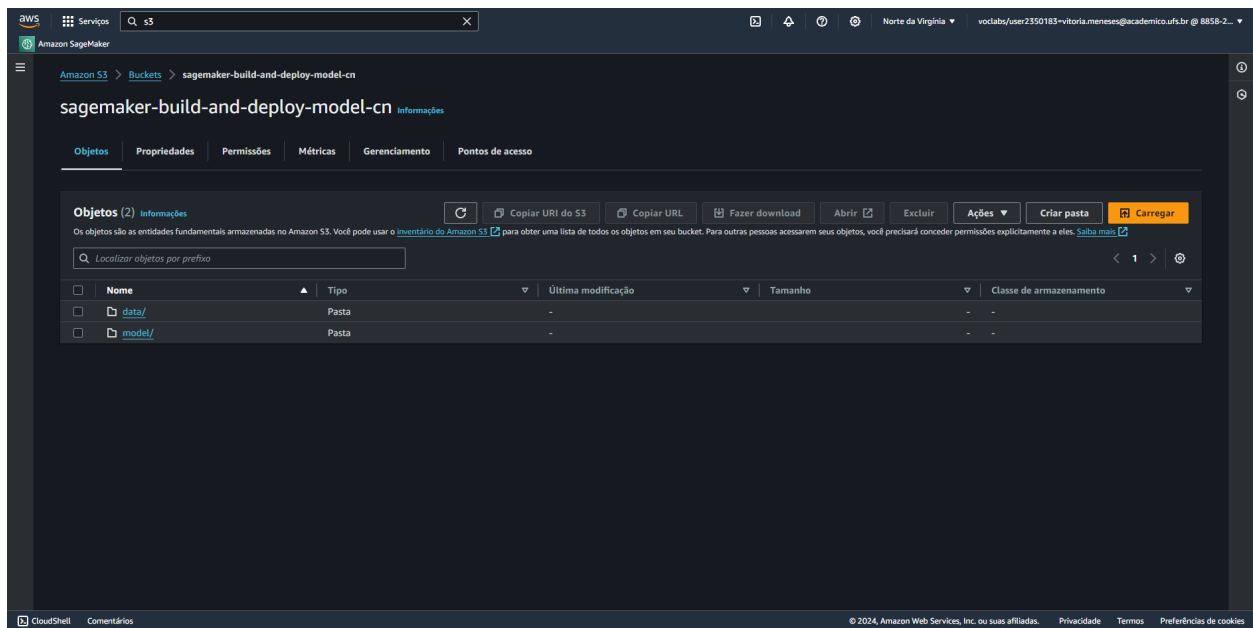
Com os dados preparados, iremos armazená-los em um bucket, utilizando o Amazon S3. Para isso, no console da AWS iremos procurar Amazon S3 e clicar na opção de 'Criar bucket':



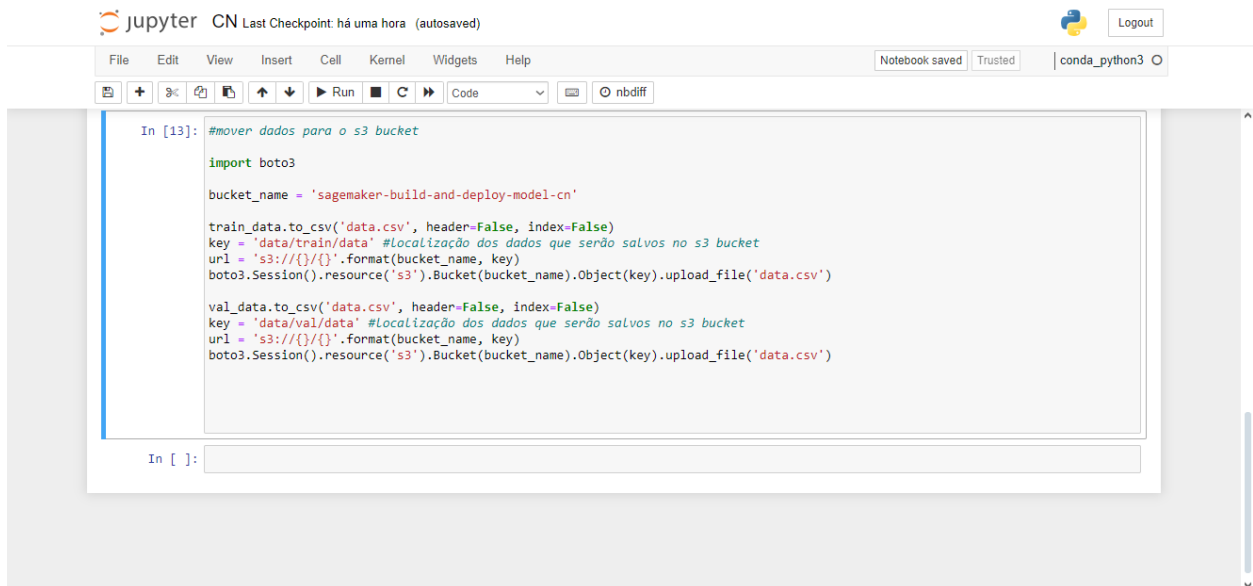
Posteriormente, será definido o nome do bucket e sua criação:







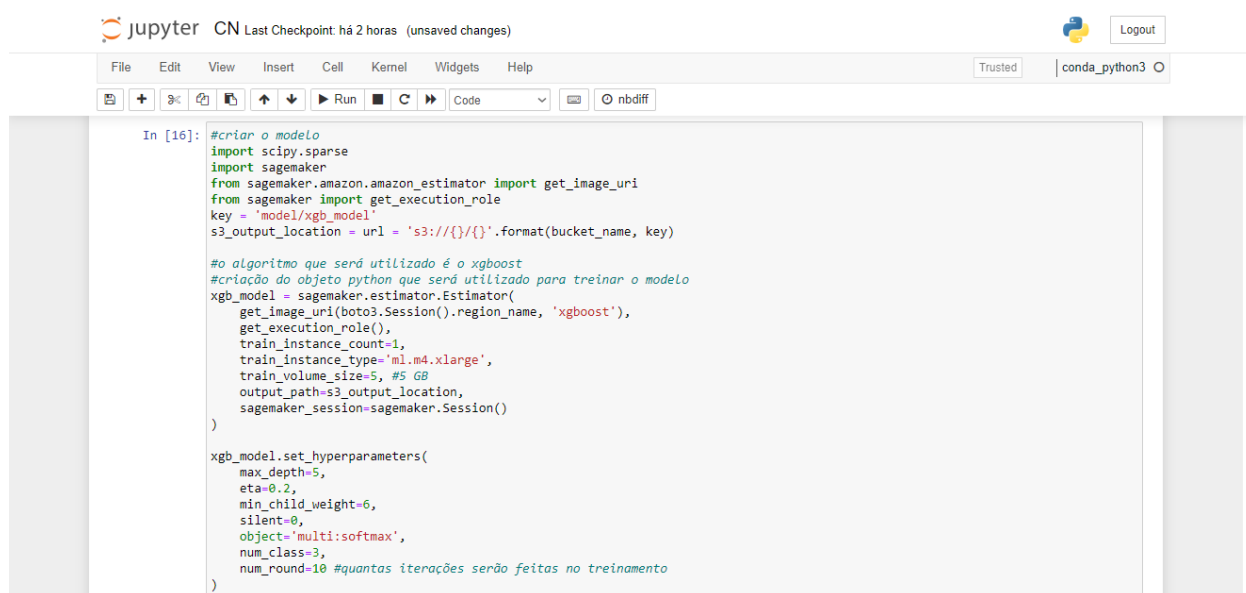
Com o bucket criado, podemos armazenar os dados já preparados nele da seguinte forma:



## 2. Treinamento

Nesta etapa, uma das tarefas é a escolha dos algoritmos de *Machine Learning* que serão usados no modelo. Neste exemplo iremos utilizar o XGBoost, um algoritmo muito utilizado para regressão, classificação (binária e multiclasse) e problemas de classificação.

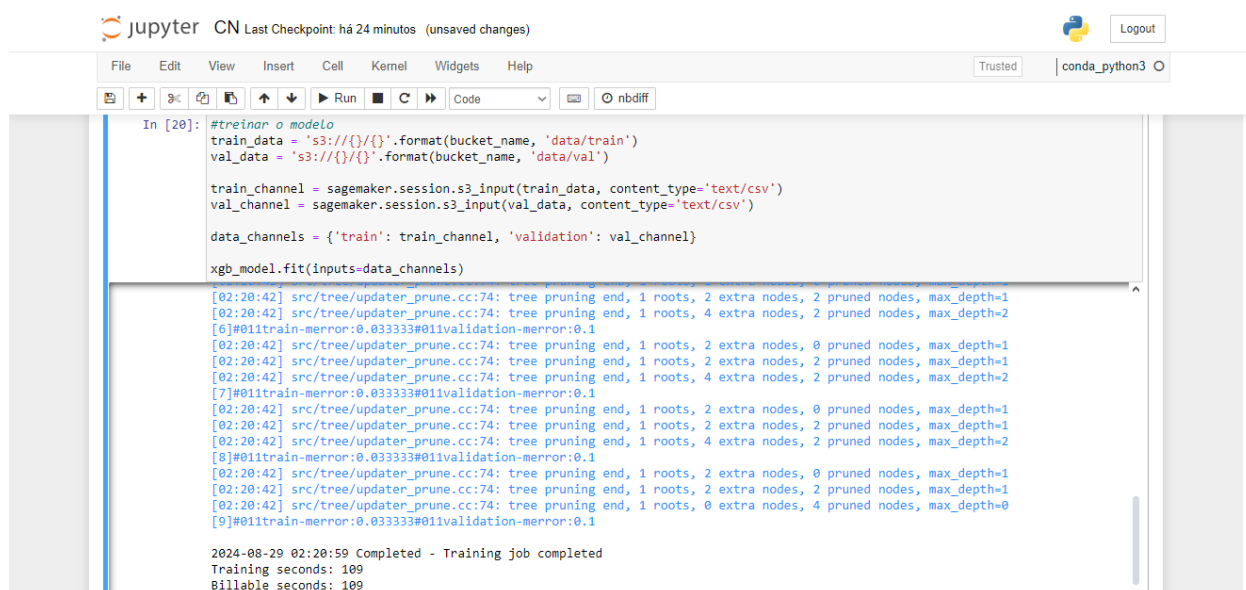
Para utilizá-lo precisamos definir os seus hiperparâmetros e criar o objeto python que será utilizado para treinar o modelo.



```
In [16]: #criar o modelo
import scipy.sparse
import sagemaker
from sagemaker.amazon.amazon_estimator import get_image_uri
from sagemaker import get_execution_role
key = 'model/xgb_model'
s3_output_location = url = 's3://{}/{}'.format(bucket_name, key)

#o algoritmo que será utilizado é o xgboost
#criação do objeto python que será utilizado para treinar o modelo
xgb_model = sagemaker.estimator.Estimator(
    get_image_uri(boto3.Session().region_name, 'xgboost'),
    get_execution_role(),
    train_instance_count=1,
    train_instance_type='ml.m4.xlarge',
    train_volume_size=5, #5 GB
    output_path=s3_output_location,
    sagemaker_session=sagemaker.Session()
)

xgb_model.set_hyperparameters(
    max_depth=5,
    eta=0.2,
    min_child_weight=6,
    silent=0,
    objective='multi:softmax',
    num_class=3,
    num_round=10 #quantas iterações serão feitas no treinamento
)
```



```
In [20]: #treinar o modelo
train_data = 's3://{}/{}'.format(bucket_name, 'data/train')
val_data = 's3://{}/{}'.format(bucket_name, 'data/val')

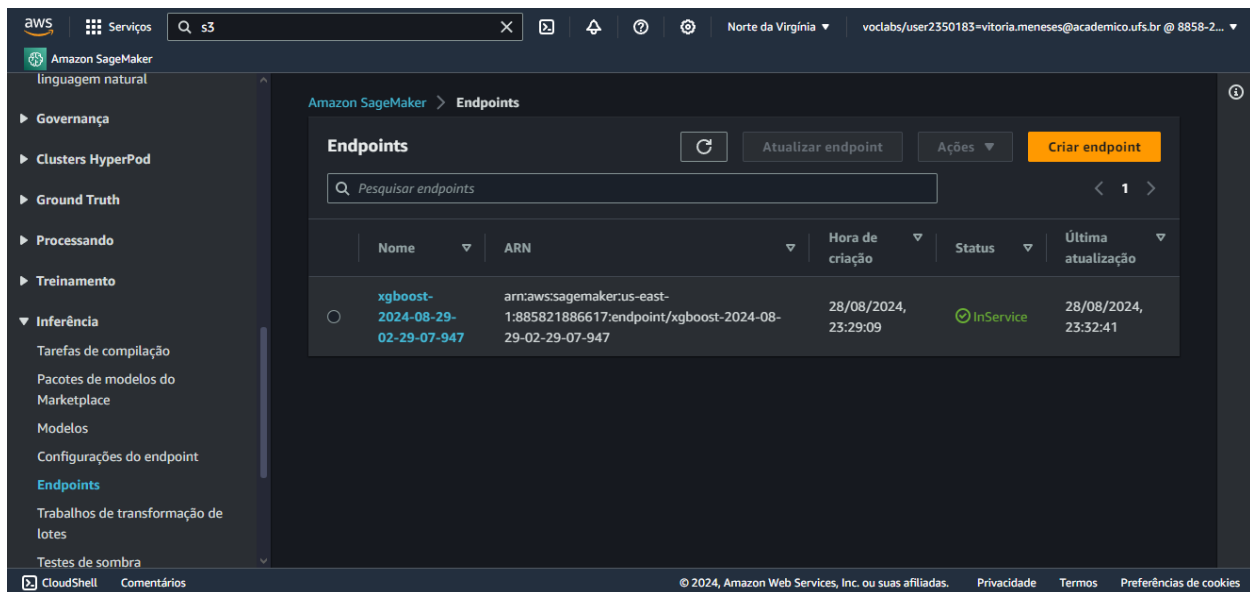
train_channel = sagemaker.session.s3_input(train_data, content_type='text/csv')
val_channel = sagemaker.session.s3_input(val_data, content_type='text/csv')

data_channels = {'train': train_channel, 'validation': val_channel}

xgb_model.fit(inputs=data_channels)

[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 2 pruned nodes, max_depth=1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 2 pruned nodes, max_depth=2
[6]#011train-merror:0.033333#011validation-merror:0.1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 2 pruned nodes, max_depth=1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 2 pruned nodes, max_depth=2
[7]#011train-merror:0.033333#011validation-merror:0.1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 2 pruned nodes, max_depth=1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 2 pruned nodes, max_depth=2
[8]#011train-merror:0.033333#011validation-merror:0.1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 2 pruned nodes, max_depth=1
[02:20:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 4 pruned nodes, max_depth=0
[9]#011train-merror:0.033333#011validation-merror:0.1

2024-08-29 02:20:59 Completed - Training job completed
Training seconds: 109
Billable seconds: 109
```



### 3. Implantação (ou deploy) do modelo

Por último, após o treinamento, o modelo pode ser implantado em um *endpoint* do Amazon SageMaker, fazendo previsões em tempo real com base em novos dados de entrada.

