

FairPEK - Documentação

Instalação

October 17, 2022

1 Introdução

Esta documentação foi criada com o objetivo de guiar o desenvolvedor de Software a entender, configurar e manter este sistema, que é dividido em 4 etapas principais:

- **Engenharia de dados:** Etapa criada com o objetivo de simular processos de transformação e limpeza de dados.
- **Pipeline de IA:** Etapa para execução de um Pipeline que simula o desenvolvimento de uma aplicação automatizada de IA, desde uma categorização dos dados mais específica do que na etapa anterior, passando pelo algoritmo utilizado e finalizando obtendo métricas para determinar qualidade do resultado final.
- **Autonomia do Pipeline (Componente MAPE-K):** Etapa que executa um componente para automatizar todas as etapas do Pipeline, com o objetivo de evitar com que perca-se tempo em execuções manuais que podem demorar dependendo do algoritmo e do conjunto de dados utilizado.
- **Interface:** Etapa criada com o objetivo de simular a etapa anterior, porém de modo a proporcionar uma experiência de usuário mais simples e intuitiva. É dividida em duas partes:
 - **Frontend:** Parte visual, exibida em um navegador.
 - **Backend:** Parte onde o Frontend se comunica para obter os dados e montar o visual corretamente, de forma que corresponda a configurações utilizadas pelo Componente MAPE-K.

2 Programas necessários para instalação

- **Python:** É a linguagem de programação utilizada para montar e executar todas as etapas com a exceção do Frontend da interface. É disponível no site <https://www.python.org/> e é necessária a versão **3.8** ou superior.

- **Node.js:** É o programa necessário para montar o Frontend da interface. É disponível no site <https://nodejs.org/> e foi testado na versão **16.14.2**, embora outras versões podem ser executadas sem problemas de compatibilidade.
- **Git:** É o programa necessário para realizar o download do código-fonte e realizar atualizações no mesmo. É disponível no site <https://git-scm.com/> e foi testado na versão **2.35.1**, embora outras versões podem ser executadas sem problemas de compatibilidade.

3 Instalação do sistema

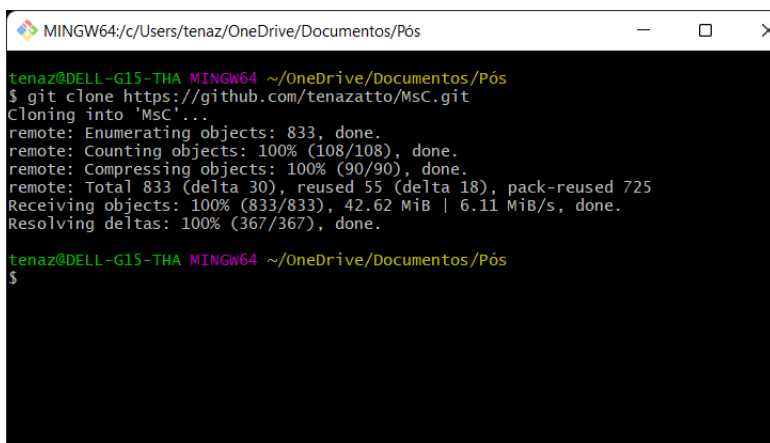
A partir desta parte, os exemplos serão realizados utilizando o Git Bash no Sistema Operacional Windows. Entretanto, no Linux e no Mac os passos são semelhantes por ambos também utilizarem esta linha de comando.

3.1 Obtenção do código-fonte

O sistema se encontra no repositório <https://github.com/tenazatto/MsC>. Para obter seu código-fonte, basta digitar o seguinte comando:

```
git clone https://github.com/tenazatto/MsC.git
```

O Git baixará todos os arquivos e após o download é possível ver a pasta e seus arquivos na pasta **MsC**



```
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós
$ git clone https://github.com/tenazatto/MsC.git
Cloning into 'MsC'...
remote: Enumerating objects: 833, done.
remote: Counting objects: 100% (108/108), done.
remote: Compressing objects: 100% (90/90), done.
remote: Total 833 (delta 30), reused 55 (delta 18), pack-reused 725
Receiving objects: 100% (833/833), 42.62 MiB | 6.11 MiB/s, done.
Resolving deltas: 100% (367/367), done.

tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós
$
```

3.2 Montagem de ambiente

Para evitar problemas de versão com bibliotecas de outros projetos instalados, é possível criar um ambiente virtual para realizar a instalação das bibliotecas

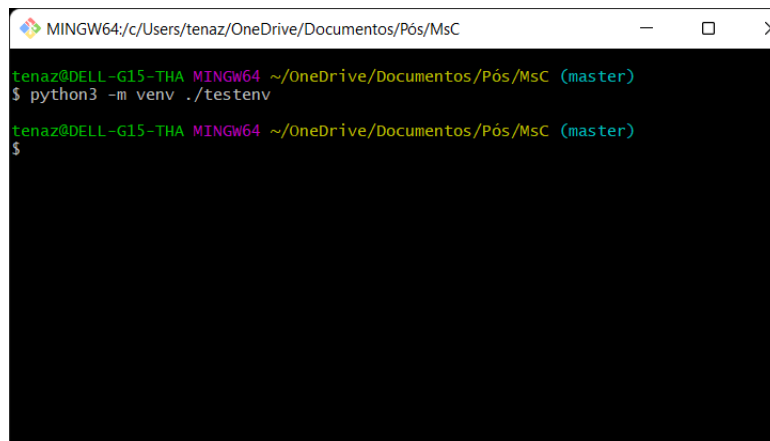
separadamente. Para criar, é necessário o **virtualenv** instalado no Python. Caso ele não esteja instalado, ele é obtido através do comando:

```
pip install virtualenv
```

Para criar um novo ambiente virtual, é preciso digitar o comando

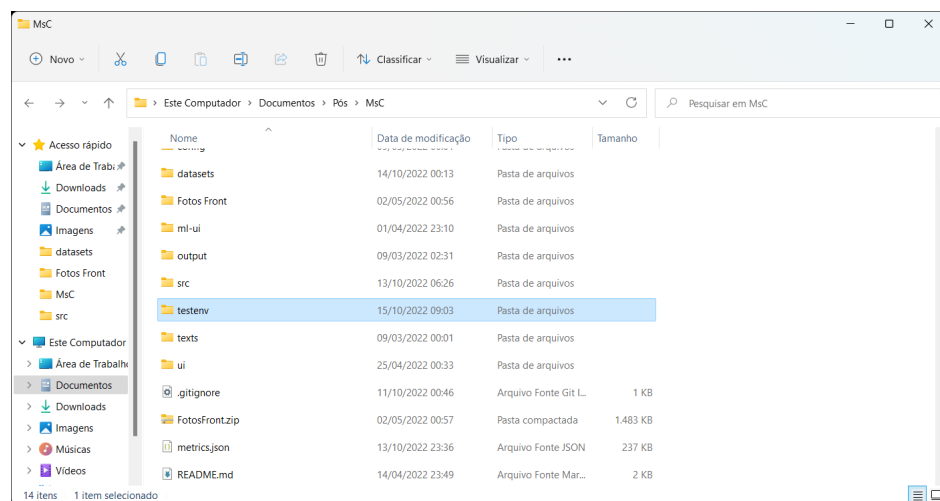
```
python3 -m venv ./(nome do ambiente)
```

Como exemplo, nesta documentação foi criado o documento **testenv**



```
MINGW64:/c:/Users/tenaz/OneDrive/Documents/Pós/MsC
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documents/Pós/MsC (master)
$ python3 -m venv ./testenv
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documents/Pós/MsC (master)
$
```

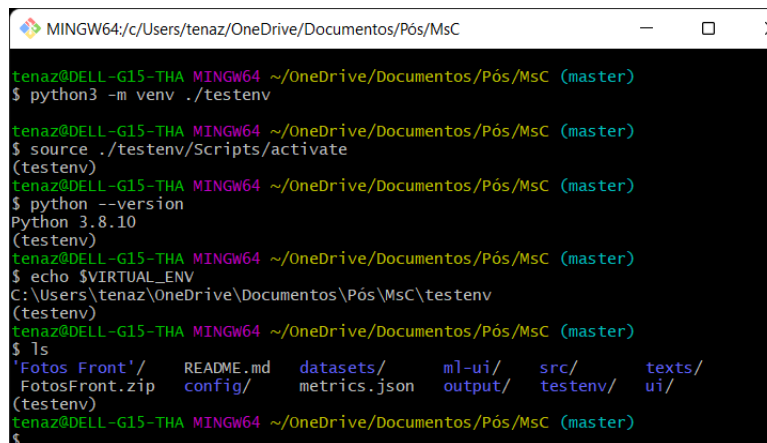
Após o término, aparecerá uma nova pasta de mesmo nome



Após criar o ambiente virtual, é preciso ativá-lo para utilizar

```
source ./(nome do ambiente)/Scripts/activate
```

Para verificar se o ambiente foi ativado, é possível verificar, ao digitar qualquer comando no bash, que o nome do ambiente virtual aparece logo abaixo.



```
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ python3 -m venv ./testenv

tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ source ./testenv/Scripts/activate
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ python --version
Python 3.8.10
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ echo $VIRTUAL_ENV
C:\Users\tenaz\OneDrive\Documentos\Pós\MsC\testenv
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ ls
'Fotos Front'/'  README.md  datasets/  ml-ui/  src/  texts/
FotosFront.zip  config/    metrics.json  output/  testenv/  ui/
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$
```

Para desativar o ambiente virtual, é preciso digitar o comando

```
deactivate
```

Para verificar se o ambiente foi desativado, é possível verificar, ao digitar qualquer comando no bash, que o nome do ambiente virtual não irá mais aparecer até ser ativado novamente.

No caso do Node.js não é necessário realizar tais etapas, pois a instalação das bibliotecas nesta documentação é realizada de maneira local

3.3 Instalação das bibliotecas

3.3.1 Python

Com o ambiente virtual criado e ativado, é possível utilizar o arquivo **src/requirements.txt** para instalar todas as bibliotecas necessárias através do comando

```
pip install -r ./src/requirements.txt
```

3.3.2 Node.js

Para o Node.js, como o arquivo **package.json** está dentro da pasta **ml-ui**, é possível acessar essa pasta e digitar o comando

```
npm install
```

4 Execução do sistema

4.1 Engenharia de dados

Dentro da pasta **MsC** e com o ambiente virtual criado e ativado, para rodar a etapa de Engenharia de dados basta digitar o seguinte comando

```
python -m src.data_engineering.data_engineering_start -  
-data (Opção)
```

No momento, há 3 opções disponíveis:

- **GERMAN_CREDIT:** Manipula o German Credit Dataset, cujo arquivo está na localização **datasets/german.data**, para utilização no Pipeline.
- **LENDINGCLUB:** Baixa e manipula o Lendingclub Dataset para utilização no Pipeline.
- **METRICS:** Obtém o maior valor, menor valor e a média de cada métrica para cada Pipeline já executado.

4.2 Pipeline de IA

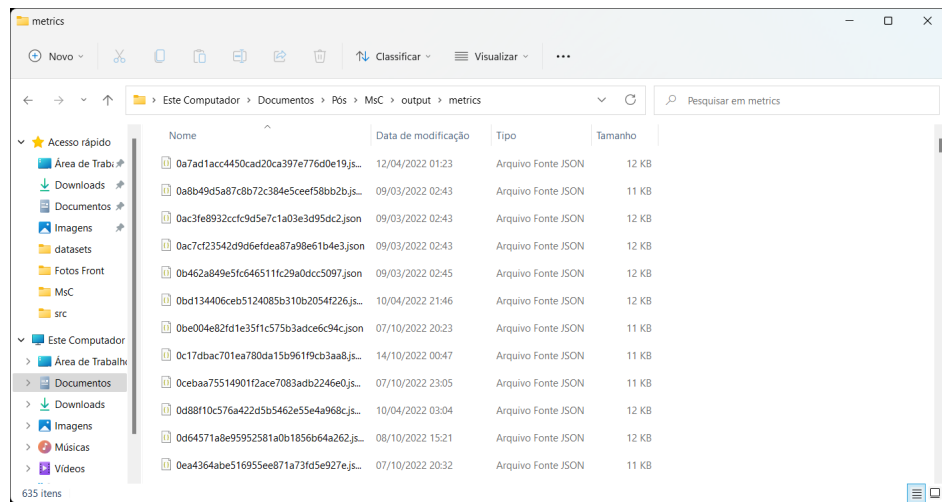
Dentro da pasta **MsC** e com o ambiente virtual criado e ativado, para rodar todos os Pipelines possíveis basta digitar o seguinte comando

```
python -m src.pipeline.pipeline_start --dataset (Opção)
```

No momento, há 4 opções disponíveis:

- **ADULT_INCOME_SEX:** Executa os Pipelines para o Adult Income Dataset, cujo arquivo está na localização **datasets/adult.csv**, utilizando Sexo (Masculino/Feminino) como atributo protegido.
- **GERMAN_CREDIT_FOREIGN:** Executa os Pipelines para o German Credit Dataset, cujo arquivo é manipulado na etapa anterior, utilizando Nacionalidade (Alemão/Estrangeiro) como atributo protegido.
- **GERMAN_CREDIT_AGE:** Executa os Pipelines para o German Credit Dataset, cujo arquivo é manipulado na etapa anterior, utilizando Idade (-25 anos/25 ou + anos) como atributo protegido.
- **LENDINGCLUB_INCOME:** Executa os Pipelines para o Lendingclub Dataset, cujo arquivo é manipulado na etapa anterior, utilizando Renda (-1 salário mínimo/1 ou + salários mínimos) como atributo protegido.

Após a execução, é possível ver a geração das métricas dentro da pasta **output/metrics**, necessárias para a execução da próxima etapa.



4.3 Autonomia do Pipeline

Dentro da pasta **MsC**, com o ambiente virtual criado e ativado e com pelo menos um Pipeline executado, é possível verificar como a etapa de autonomia funciona com o seguinte comando

```
python -m src.mapek.mapek_start
```

Nesta etapa, ele escolhe o Pipeline que apresentou as melhores métricas, porém a filtragem por conjunto de dados está desenvolvida até o momento apenas na próxima etapa. Como ele roda ininterruptamente, é preciso interromper sua execução.

4.4 Interface

4.4.1 Backend

Dentro da pasta **MsC**, com o ambiente virtual criado e ativado e com pelo menos um Pipeline executado, é possível rodar o Backend da interface com o seguinte comando

```
python -m src.api.flask_start
```

Ele vai iniciar um servidor na porta 8080, necessário para rodar as requisições que o Frontend vai solicitar

4.4.2 Frontend

Dentro da pasta **MsC/ml-ui**, é possível rodar o Frontend da interface com o seguinte comando

```
MINGW64/c/Users/tenaz/OneDrive/Documentos/Pós/MsC
(testenv)
tenaz@DELL-G15-THA MINGW64 ~/OneDrive/Documentos/Pós/MsC (master)
$ python -m src.api.flask_start
2022-10-15 10:27:17.437341: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll not found
2022-10-15 10:27:17.437371: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
* Serving Flask app 'flask_start' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Running on http://127.0.0.1:8080
Press CTRL+C to quit
```

npm start

Ele vai iniciar o navegador acessando um servidor na porta 3000, e deverá iniciar a tela no menu de Análise

