

FairPEK - Documentação

Manutenção

January 19, 2023

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.
- **Workflow 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 Workflow (Componente MAPE-K):** Etapa que executa um componente para automatizar todas as etapas do Workflow, 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 Arquitetura do Workflow e Framework

O Workflow usa uma arquitetura chamada de Pipe-and-Filter, onde Pipes, que transportam os dados, são ligados por Filters, que realizam as manipulações

desses dados. Pipes e Filters se encadeiam para formar uma sequência de operações, caracterizando todo o Pipeline. Para auxiliar no desenvolvimento, um pequeno framework foi criado, para facilitar as operações necessárias entre Pipes e Filters.

2.1 Estrutura

Pipes herdam a classe **BasePipe**. Essa classe possui o atributo **value**, que caracteriza os dados transformados, em formato de um dicionário Python

```
1 class FairnessPipe(BasePipe):
2     privileged_group = []
3     unprivileged_group = []
4
5     label_names = []
6     protected_attribute_names = []
7
8     optim_options = {}
9
10    def __init__(self):
11        self.value = {
12            'privileged_group': self.privileged_group,
13            'unprivileged_group': self.unprivileged_group,
14            'label_names': self.label_names,
15            'protected_attribute_names': self.
16            protected_attribute_names,
17            'optim_options': self.optim_options
18        }
```

Filters herdam a classe **BaseFilter**. Essa classe possui dois Pipes (input e output) e um método chamado **execute**, que é onde as operações de transformação do Pipe de input são executadas para serem colocadas no Pipe de output

```
1 from sklearn.model_selection import train_test_split
2
3 from src.pipeline.pipe_filter.pipe import BaseFilter
4
5
6 class TrainTestSplit(BaseFilter):
7     test_size = 0.2
8
9     def __init__(self, test_size=0.2):
10        self.test_size = test_size
11
12    def execute(self):
13        df_x = self.input['df_x']
14        df_y = self.input['df_y']
15
16        x_train, x_test, y_train, y_test = train_test_split(df_x,
17        df_y, test_size=self.test_size, random_state=42)
18
19        self.output = {
20            'x_train': x_train,
21            'x_test': x_test,
22            'y_train': y_train,
```

```

22         'y_test': y_test,
23         'checksum': self.input['checksum']
24     }

```

2.2 Operações

No Framework, estão presentes as seguintes operações:

2.3 Ligação de Pipe com Filter

Para juntar um Pipe com um Filter, basta realizar o comando `>=`, caracterizando o transporte a um Filter.

```

1 Pipe1() >= Filter1()

```

Se o Pipe estiver carregando os dados corretos, o Filter será automaticamente executado.

2.4 Ligação de Filter com Pipe

Para juntar um Filter com um Pipe, basta realizar o comando `==`, caracterizando o transporte a um Pipe.

```

1 Filter1() == Pipe1()

```

Se o Filter estiver corretamente implementado, o Pipe será caracterizado como a saída desse mesmo Filter.

2.5 Seleção parcial de dados presentes no Pipe

Para selecionar apenas alguns atributos presentes no Pipe, basta adicionar colchetes e colocar os campos desejados.

```

1 pipe1['campo1', 'campo2', 'campo3']

```

2.6 Junção de Pipes

Para juntar os dados de dois pipes em um só, basta realizar o comando `+`, caracterizando uma junção de Pipes.

```

1 pipe1 + pipe2

```

3 Incrementos no Workflow

Aqui estão dois exemplos de como é possível realizar a manutenção do Workflow

3.1 Adicionando um novo Conjunto de Dados

3.1.1 Engenharia de dados

1. Geralmente conjuntos de dados não vem filtrados, e é preciso um trabalho de Engenharia de dados para realizar experimentos com melhores resultados. Neste sistema, a parte de Engenharia de dados se encontra na pasta **src/data_engineering**.
2. Os métodos onde os processamentos são realizados ficam no arquivo **data_engineering.py**. Para adicionar um novo processamento, basta adicionar um novo método neste arquivo.
3. Importar o novo método no arquivo **data_engineering_start.py**.
4. Adicionar uma nova opção no parâmetro **choices** presente no método **parser.add_argument**

```
1 parser.add_argument("--data", help="Selecao do gerador do  
conjunto de dados tratado",  
2 choices=['GERMAN_CREDIT', 'LENDINGCLUB  
, 'METRICS'])
```

5. Adicionar uma nova condição com a opção adicionada

3.1.2 Workflow

1. Neste sistema, a parte do Workflow se encontra na pasta **src/pipeline**. Dentro dela, os Pipes que armazenam os conjuntos de dados se encontram na pasta **processors/preprocessors/data**. Dentro dela, no arquivo **dataset.py**, criar uma classe que herda a classe **Dataset**, preenchendo os atributos **dataset_path** com o caminho do arquivo.

```
1 class LendingclubDataset(Dataset):  
2     dataset_path = 'datasets/lendingclub_dataset.csv'  
3  
4     def __init__(self):  
5         super().__init__()
```

2. Criar um novo arquivo.
3. Criar uma classe que herda a classe **FairnessPipe**, preenchendo os atributos **privileged_group**, **unprivileged_group**, **label_names**, **protected_attribute_names** e **optim_options**.

```
1 class LendingclubIncomeFairnessPipe(FairnessPipe):  
2     privileged_group = [{'annual_inc': 1}]  
3     unprivileged_group = [{'annual_inc': 0}]  
4  
5     label_names = ['loan_status']  
6     protected_attribute_names = ['annual_inc']  
7  
8     optim_options = {
```

```

9         "distortion_fun": get_distortion_german,
10        "epsilon": 0.05,
11        "clist": [0.99, 1.99, 2.99],
12        "dlist": [.1, 0.05, 0]
13    }
14
15    def __init__(self):
16        super().__init__()

```

4. Criar uma classe que herda a classe **FairnessPreprocessor**, preenchendo o método **dataset_preprocess** e retornando um DataFrame Pandas de dados de entrada e um DataFrame de dados de saída.

```

1 class LendingclubIncomePreprocessor(FairnessPreprocessor):
2     def dataset_preprocess(self, df):
3         df.info()
4
5         SAMPLE_PERCENTAGE = 100
6         df_sample_nok = df[df['loan_status'] == 'Charged Off']
7         df_sample_nok = df_sample_nok.sample(frac=SAMPLE_PERCENTAGE/100)
8         df_sample_ok = df[df['loan_status'] == 'Fully Paid'].
9         df_sample_ok = df_sample_ok.sample(frac=SAMPLE_PERCENTAGE / 100)
10        df_sample = pd.concat([df_sample_ok, df_sample_nok])
11
12        df_x = df_sample.drop('loan_status', axis=1)
13        df_y = pd.DataFrame(df_sample.loan_status)
14
15        return df_x, df_y

```

5. Na pasta **src/pipeline/processors** e dentro do arquivo **enums.py**, colocar novas opções nas classes **Datasets** e **Preprocessors**.

```

1 class Datasets(ExtendedEnum):
2     ADULT_INCOME = 1
3     GERMAN_CREDIT = 2
4     LENDINGCLUB = 3
5
6
7 class Preprocessors(ExtendedEnum):
8     SEX = 1
9     AGE = 2
10    FOREIGN = 3
11    INCOME = 4

```

6. Na pasta **src/pipeline** e dentro do arquivo **validation.py**, atualizar a variável **existant_preprocessors** com as novas opções colocadas no item anterior.

```

1     existent_preprocessors = \
2         (dataset == Datasets.ADULT_INCOME and preprocessor
3          == Preprocessors.SEX) or \
4         (dataset == Datasets.GERMAN_CREDIT and
5          preprocessor == Preprocessors.AGE) or \
6         (dataset == Datasets.LENDINGCLUB and
7          preprocessor == Preprocessors.FOREIGN) or \

```

```

5         (dataset == Datasets.LENDINGCLUB and preprocessor
        == Preprocessors.INCOME)

```

7. Na pasta **src/pipeline** e dentro do arquivo **pipeline.py**, encontrar o método **select_data_preprocessor**, atualizar a variável **options** com as novas opções e classes implementadas nos itens anteriores.

```

1     def select_data_preprocessor(self, dataset, preprocessor):
2         choice = [dataset, preprocessor]
3         options = [
4             ([Datasets.ADULT_INCOME, Preprocessors.SEX], (
5                 AdultDataset(), AdultSexPreprocessor(),
6                 AdultSexFairnessPipe()))),
7             ([Datasets.GERMAN_CREDIT, Preprocessors.AGE], (
8                 GermanDataset(), GermanAgePreprocessor(),
9                 GermanAgeFairnessPipe()))),
10            ([Datasets.GERMAN_CREDIT, Preprocessors.FOREIGN],
11            (GermanDataset(), GermanForeignPreprocessor(),
12            GermanForeignFairnessPipe()))),
13            ([Datasets.LENDINGCLUB, Preprocessors.INCOME], (
14                LendingclubDataset(), LendingclubIncomePreprocessor(),
15                LendingclubIncomeFairnessPipe()))),
16        ]
17
18        for option, pipe_filter in options:
19            if choice == option:
20                dataset_pipe, data_preprocessor_filter,
21                fairness_pipe = pipe_filter
22                preprocessed_data_pipe = dataset_pipe >=
23                data_preprocessor_filter == self.new_pipe()
24                break
25
26        return preprocessed_data_pipe, fairness_pipe

```

8. Na pasta **src/pipeline** e dentro do arquivo **pipeline_start.py**, adicionar uma nova opção no parâmetro **choices** presente no método **parser.add_argument**.

```

1     parser.add_argument("--dataset", help="Conjunto de dados
2         tratado com atributo protegido",
3         choices=['ADULT_INCOME_SEX',
4                 'GERMAN_CREDIT_FOREIGN', '
5                 GERMAN_CREDIT_AGE',
6                 'LENDINGCLUB_INCOME'])

```

9. No mesmo arquivo, adicionar uma nova condição com a opção adicionada.

```

1     if args.dataset == 'ADULT_INCOME_SEX':
2         datasets.append((Datasets.ADULT_INCOME, Preprocessors.
3             SEX))
4     elif args.dataset == 'ADULT_INCOME_FOREIGN':
5         datasets.append((Datasets.GERMAN_CREDIT, Preprocessors
6             .FOREIGN))
7     elif args.dataset == 'GERMAN_CREDIT_AGE':
8         datasets.append((Datasets.GERMAN_CREDIT, Preprocessors
9             .AGE))
10    elif args.dataset == 'LENDINGCLUB_INCOME':

```

```

8     datasets.append((Datasets.LENDINGCLUB, Preprocessors.
    INCOME))]
```

3.1.3 Interface (Backend)

1. Neste sistema, a parte do Backend se encontra na pasta **src/api**. Dentro dela, no arquivo **repo/pipeline.py**, adicionar a opção no método **get_dataset**.

```

1     def get_dataset(self, dataset):
2         indexes = {
3             'Datasets.ADULT_INCOME': Datasets.ADULT_INCOME,
4             'Datasets.GERMAN_CREDIT': Datasets.GERMAN_CREDIT,
5             'Datasets.LENDINGCLUB': Datasets.LENDINGCLUB,
6         }
7
8         return next(filter(lambda a: a[0] == dataset, indexes.
    items()))[1]
```

2. Na pasta **src/api** e dentro do arquivo **repo/pipeline.py**, adicionar a opção no método **get_preprocessor**.

```

1     def get_preprocessor(self, preprocessor):
2         indexes = {
3             'Preprocessors.SEX': Preprocessors.SEX,
4             'Preprocessors.AGE': Preprocessors.AGE,
5             'Preprocessors.FOREIGN': Preprocessors.FOREIGN
6         }
7
8         return next(filter(lambda a: a[0] == preprocessor,
    indexes.items()))[1]
```

3.1.4 Interface (Frontend)

1. Neste sistema, a parte do Frontend se encontra na pasta **ml-ui/src**. Dentro dela, no arquivo **Auto-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Workflow no componente Select onde estão as outras opções de conjunto de dados.

```

1 <Select
2   sx={{fontSize: '14px'}}
3   value={dataset}
4   onChange={handleDatasetChange}
5   displayEmpty
6   inputProps={{ 'aria-label': 'Without label' }}
7 >
8   <MenuItem value={'Datasets.ADULT_INCOME'}>Adult Income
    Dataset</MenuItem>
9   <MenuItem value={'Datasets.GERMAN_CREDIT'}>German Credit
    Dataset</MenuItem>
10  <MenuItem value={'Datasets.LENDINGCLUB'}>Lendingclub Dataset
    </MenuItem>
11 </Select>
```

2. Na pasta **ml-ui/src** e dentro do arquivo **Manual-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Workflow no componente Select onde estão as outras opções de conjunto de dados.

```

1 <Select
2   sx={{fontSize: '14px'}}
3   value={dataset}
4   onChange={handleDatasetChange}
5   displayEmpty
6   inputProps={{ 'aria-label': 'Without label' }}
7 >
8   <MenuItem value={'Datasets.ADULT_INCOME'}>Adult Income
9     Dataset</MenuItem>
10  <MenuItem value={'Datasets.GERMAN_CREDIT'}>German Credit
11  <MenuItem value={'Datasets.LENDINGCLUB'}>Lendingclub Dataset
12  </MenuItem>
13 </Select>

```

3. Na pasta **ml-ui/src** e dentro do arquivo **Manual-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Workflow no componente Select onde estão as outras opções de pré-processadores.

```

1 <FormControl sx={{ m: 1, width: 300, marginLeft: '35px' }}>
2   {dataset === 'Datasets.ADULT_INCOME' ?
3     <Select
4       sx={{fontSize: '14px'}}
5       value={protectedAtt}
6       onChange={handleProtectedAttChange}
7       displayEmpty
8       inputProps={{ 'aria-label': 'Without label' }}
9     >
10      <MenuItem value={'Preprocessors.SEX'}>Sexo (Masculino/
11        Feminino)</MenuItem>
12    </Select>
13    : dataset === 'Datasets.ADULT_INCOME' ?
14      <Select
15        sx={{fontSize: '14px'}}
16        value={protectedAtt}
17        onChange={handleProtectedAttChange}
18        displayEmpty
19        inputProps={{ 'aria-label': 'Without label' }}
20      >
21        <MenuItem value={'Preprocessors.AGE'}>Idade (-25 anos
22          /+25 anos)</MenuItem>
23        <MenuItem value={'Preprocessors.FOREIGN'}>Nacionalidade
24          (Local/Estrangeiro)</MenuItem>
25      </Select>
26    :
27      <Select
28        sx={{fontSize: '14px'}}
29        value={protectedAtt}
30        onChange={handleProtectedAttChange}
31        displayEmpty
32        inputProps={{ 'aria-label': 'Without label' }}
33      >

```



```

31     <MenuItem value={'Preprocessors.INCOME'}>Renda (-1
        Salario Minimo/1+ Salarios Minimos)</MenuItem>
32   </Select>
33 }
34 <FormHelperText>Atributo protegido para medir justicia</
    FormHelperText>
35 </FormControl>

```

4. Na pasta **ml-ui/src** e dentro do arquivo **Manual-Pipeline-Menu.js** adicionar a condição para a opção colocada na etapa de Workflow no método **handleDatasetChange**.

```

1   const handleDatasetChange = (event) => {
2     setDataset(event.target.value);
3
4     if (event.target.value === 'Datasets.ADULT_INCOME') {
5       setProtectedAtt('Preprocessors.SEX');
6     } else if (event.target.value === 'Datasets.GERMAN_CREDIT') {
7       setProtectedAtt('Preprocessors.AGE');
8     } else {
9       setProtectedAtt('Preprocessors.INCOME');
10    }
11  }

```

3.2 Adicionando um novo Algoritmo

3.2.1 Workflow

1. Neste sistema, a parte do Workflow se encontra na pasta **src/pipeline**. Dentro dela, os Filters que executam os algoritmos ficam dentro da pasta **processors**. Dentro dela, no arquivo **enums.py**, colocar novas opções na classe **Algorithms**.

```

1  class Algorithms:
2      LOGISTIC_REGRESSION = 1
3      RANDOM_FOREST = 2
4      GRADIENT_BOOST = 3
5      SUPPORT_VECTOR_MACHINES = 4
6      LINEAR_REGRESSION = 901
7      DECISION_TREE = 902
8      KERNEL_RIDGE = 903

```

A classe **Algorithms** serve para este exemplo em questão, mas para outros tipos de algoritmos as classes **UnbiasDataAlgorithms**, **UnbiasInProcAlgorithms** ou **UnbiasPostProcAlgorithms** podem ser mais adequadas.

2. Na pasta **src/pipeline** e dentro do arquivo **validation.py**, adicionar uma nova condição para a variável **existant_algorithms** no método **validate_params**.

```

1      existant_algorithms = \
2      (algorithm == Algorithms.LOGISTIC_REGRESSION and
unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
NOTHING) or \
3      (...)
4      (algorithm == UnbiasInProcAlgorithms.
GRID_SEARCH_REDUCTION and unbias_data_algorithm ==
UnbiasDataAlgorithms.NOTHING and unbias_postproc_algorithm
== UnbiasPostProcAlgorithms.NOTHING) or \
5      (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
NOTHING) or \
6      (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
EQUALIZED_ODDS) or \
7      (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
CALIBRATED_EQUALIZED_ODDS) or \
8      (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.NOTHING and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
REJECT_OPTION_CLASSIFICATION) or \
9      (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.REWEIGHING
and unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
NOTHING) or \
10     (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.
DISPARATE_IMPACT_REMOVER and unbias_postproc_algorithm ==
UnbiasPostProcAlgorithms.NOTHING) or \
11     (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.
OPTIMIZED_PREPROCESSING and unbias_postproc_algorithm ==
UnbiasPostProcAlgorithms.NOTHING) or \
12     (algorithm == Algorithms.NOVA_OPCAO and
unbias_data_algorithm == UnbiasDataAlgorithms.
LEARNING_FAIR_REPRESENTATIONS and
unbias_postproc_algorithm == UnbiasPostProcAlgorithms.
NOTHING)

```

3. Na pasta **src/pipeline** e dentro do arquivo **pipeline.py**, colocar as novas opções colocadas no item anterior no método **find_algorithm**.

```

1      def find_algorithm(self, algorithm):
2          indexes = {
3              'Algorithms.LOGISTIC_REGRESSION': 1,
4              'Algorithms.RANDOM_FOREST': 2,
5              'Algorithms.GRADIENT_BOOST': 3,
6              'Algorithms.SUPPORT_VECTOR_MACHINES': 4,
7              'Algorithms.LINEAR_REGRESSION': 901,
8              'Algorithms.DECISION_TREE': 902,
9              'Algorithms.KERNEL_RIDGE': 903,
10             'UnbiasInProcAlgorithms.PREJUDICE_REMOVER': 101,

```

```

11         'UnbiasInProcAlgorithms.ADVERSARIAL_DEBIASING':
12         102,
13         'UnbiasInProcAlgorithms.
14         EXPONENTIATED_GRADIENT_REDUCTION': 103,
15         'UnbiasInProcAlgorithms.RICH_SUBGROUP_FAIRNESS':
16         104,
17         'UnbiasInProcAlgorithms.GRID_SEARCH_REDUCTION':
18         105,
19         'UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER':
20         106,
21         'UnbiasInProcAlgorithms.ART_CLASSIFIER': 107
22     }
23
24     return next(filter(lambda a: a[1] == algorithm,
25                       indexes.items()))[0]

```

4. Criar um novo arquivo na pasta categorizada pelo algoritmo a ser implementado. Para o exemplo exemplo ilustrado abaixo, como o Gradient Boosting é um algoritmo de treinamento e não foi projetado para redução de viés, os Filters que executam os algoritmos se encontram na pasta **processors/inprocessors/inproc_algorithms**.

Seguem as pastas onde os respectivos Filters se encontram:

- **Algoritmo de treinamento sem redução de viés:** processors/inprocessors/inproc_algorithms
- **Algoritmo com redução de viés no dado (Pré-processamento):** processors/preprocessors/unbias_algorithms
- **Algoritmo com redução de viés no treinamento (Processamento):** processors/inprocessors/unbias_algorithms
- **Algoritmo com redução de viés no resultado (Pós-processamento):** processors/postprocessors

5. No arquivo criado, criar uma classe que herda a classe **BaseFilter**.

```

1 class GradientBoostFilter(BaseFilter):
2     weighed = False
3
4     def __init__(self, weighed=False):
5         self.weighed = weighed

```

6. Implementar nesta classe o método **execute**, atribuindo em **self.output** um dicionário Python com os atributos **y_pred** e **scores**.

```

1     def execute(self):
2         y_pred, scores = self.gradient_boost_weighed() if self
3         .weighed else self.gradient_boost()
4
5         self.output = {
6             'y_pred': y_pred,
7             'scores': scores
8         }

```

- Na pasta **src/pipeline** e dentro do arquivo **pipeline.py**, encontrar o método **process**, atualizar a variável **process_options** com as novas opções e classes implementadas nos itens anteriores.

```

1  def process(self, process_pipe, algorithm,
2      unbiased_data_algorithm):
3      weighed_algorithm = unbiased_data_algorithm ==
4      UnbiasDataAlgorithms.REWEIGHING or \
5      unbiased_data_algorithm ==
6      UnbiasDataAlgorithms.LEARNING_FAIR_REPRESENTATIONS
7
8      process_options = [
9          (Algorithms.LOGISTIC_REGRESSION,
10           LogisticRegressionFilter(weighed=weighed_algorithm)),
11          (Algorithms.RANDOM_FOREST, RandomForestFilter(
12           weighed=weighed_algorithm)),
13          (Algorithms.GRADIENT_BOOST, GradientBoostFilter(
14           weighed=weighed_algorithm)),
15          (Algorithms.SUPPORT_VECTOR_MACHINES, SVMFilter(
16           weighed=weighed_algorithm)),
17          (UnbiasInProcAlgorithms.PREJUDICE_REMOVER,
18           PrejudiceRemoverFilter()),
19          (UnbiasInProcAlgorithms.ADVERSARIAL_DEBIASING,
20           AdversarialDebiasingFilter()),
21          (UnbiasInProcAlgorithms.
22           EXPONENTIATED_GRADIENT_REDUCTION,
23           ExponentiatedGradientReductionFilter(algorithm=Algorithms.
24           GRADIENT_BOOST)),
25          (UnbiasInProcAlgorithms.RICH_SUBGROUP_FAIRNESS,
26           RichSubgroupFairnessFilter(algorithm=Algorithms.
27           DECISION_TREE)),
28          (UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER,
29           MetaFairClassifierFilter()),
30          (UnbiasInProcAlgorithms.GRID_SEARCH_REDUCTION,
31           GridSearchReductionFilter(algorithm=Algorithms.
32           RANDOM_FOREST))
33      ]
34
35      for option, filter in process_options:
36          if algorithm == option:
37              prediction_pipe = process_pipe >= filter ==
38              self.new_pipe()
39              break
40
41      return prediction_pipe

```

O método **process** serve para este exemplo em questão, mas para outros tipos de algoritmos os métodos **unbias_data_preprocessor** ou **data_postprocess** podem ser mais adequados.

- Na pasta **src/pipeline** dentro do arquivo **pipeline_start.py**, adicionar as opções (adaptadas a opção corrente) na variável **process_options**.

```

1  process_options = [
2      (Algorithms.NOVA_OPcao, UnbiasDataAlgorithms.NOTHING,
3      UnbiasPostProcAlgorithms.NOTHING),

```

```

3      (Algorithms.NOVA_OPCAO, UnbiasDataAlgorithms.NOTHING,
      UnbiasPostProcAlgorithms.EQUALIZED_ODDS),
4      (Algorithms.NOVA_OPCAO, UnbiasDataAlgorithms.NOTHING,
5        UnbiasPostProcAlgorithms.CALIBRATED_EQUALIZED_ODDS),
6      (Algorithms.NOVA_OPCAO, UnbiasDataAlgorithms.NOTHING,
7        UnbiasPostProcAlgorithms.REJECT_OPTION_CLASSIFICATION
      ),
8      (Algorithms.NOVA_OPCAO, UnbiasDataAlgorithms.
      REWEIGHING, UnbiasPostProcAlgorithms.NOTHING),
9      (Algorithms.NOVA_OPCAO, UnbiasDataAlgorithms.
      DISPARATE_IMPACT_REMOVER,
10      UnbiasPostProcAlgorithms.NOTHING),
11      # (Algorithms.NOVA_OPCAO, UnbiasDataAlgorithms.
      OPTIMIZED_PREPROCESSING, UnbiasPostProcAlgorithms.NOTHING)
12      ,
13      (Algorithms.NOVA_OPCAO, UnbiasDataAlgorithms.
      LEARNING_FAIR_REPRESENTATIONS,
14      UnbiasPostProcAlgorithms.NOTHING)
      ]

```

3.2.2 MAPE-K

1. Neste sistema, a parte do Backend se encontra na pasta **src/mapek**. Dentro dela, no arquivo **ml/planner.py**, colocar as novas opções colocadas no item anterior no método **find_inproc_algorithm**.

```

1      def find_inproc_algorithm(self, algorithm):
2          indexes = {
3              'Algorithms.LOGISTIC_REGRESSION': 1,
4              'Algorithms.RANDOM_FOREST': 2,
5              'Algorithms.GRADIENT_BOOST': 3,
6              'Algorithms.SUPPORT_VECTOR_MACHINES': 4,
7              'Algorithms.LINEAR_REGRESSION': 901,
8              'Algorithms.DECISION_TREE': 902,
9              'Algorithms.KERNEL_RIDGE': 903,
10             'UnbiasInProcAlgorithms.PREJUDICE_REMOVER': 101,
11             'UnbiasInProcAlgorithms.ADVERSARIAL_DEBIASING':
12             102,
13             'UnbiasInProcAlgorithms.
14             EXPONENTIATED_GRADIENT_REDUCTION': 103,
15             'UnbiasInProcAlgorithms.RICH_SUBGROUP_FAIRNESS':
16             104,
17             'UnbiasInProcAlgorithms.GRID_SEARCH_REDUCTION':
18             105,
19             'UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER':
20             106,
21             'UnbiasInProcAlgorithms.ART_CLASSIFIER': 107
22         }
23
24         return next(filter(lambda a: a[0] == algorithm,
25                             indexes.items()))[1]

```

3.2.3 Interface (Backend)

1. Neste sistema, a parte do Backend se encontra na pasta **src/api**. Dentro dela, no arquivo **repo/pipeline.py**, colocar as novas opções colocadas no item anterior no método **get_inproc_algorithm**.

```
1 def get_inproc_algorithm(self, algorithm):
2     indexes = {
3         'Algorithms.LOGISTIC_REGRESSION': 1,
4         'Algorithms.RANDOM_FOREST': 2,
5         'Algorithms.GRADIENT_BOOST': 3,
6         'Algorithms.SUPPORT_VECTOR_MACHINES': 4,
7         'Algorithms.LINEAR_REGRESSION': 901,
8         'Algorithms.DECISION_TREE': 902,
9         'Algorithms.KERNEL_RIDGE': 903,
10        'UnbiasInProcAlgorithms.PREJUDICE_REMOVER': 101,
11        'UnbiasInProcAlgorithms.ADVERSARIAL_DEBIASING':
12        102,
13        'UnbiasInProcAlgorithms.
14        EXPONENTIATED_GRADIENT_REDUCTION': 103,
15        'UnbiasInProcAlgorithms.RICH_SUBGROUP_FAIRNESS':
16        104,
17        'UnbiasInProcAlgorithms.GRID_SEARCH_REDUCTION':
18        105,
19        'UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER':
20        106,
21        'UnbiasInProcAlgorithms.ART_CLASSIFIER': 107
22    }
23
24    return next(filter(lambda a: a[0] == algorithm,
25                      indexes.items()))[1]
```

3.2.4 Interface (Frontend)

1. Neste sistema, a parte do Frontend se encontra na pasta **ml-ui/src**. Dentro dela, no arquivo **Manual-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Workflow no componente Select onde estão as outras opções de algoritmos.

```
1 <Select
2   sx={{fontSize: '14px'}}
3   value={trainAlgorithm}
4   onChange={handleTrainAlgorithmChange}
5   displayEmpty
6   inputProps={{ 'aria-label': 'Without label' }}
7 >
8   <MenuItem value={'Algorithms.LOGISTIC_REGRESSION'}>Logistic
9     Regression</MenuItem>
10  <MenuItem value={'Algorithms.RANDOM_FOREST'}>Random Forest</
11    MenuItem>
12  <MenuItem value={'Algorithms.GRADIENT_BOOST'}>Gradient Boost
13    </MenuItem>
14  <MenuItem value={'Algorithms.SUPPORT_VECTOR_MACHINES'}>
15    Support Vector Machines</MenuItem>
16 </Select>
```

2. Na pasta **ml-ui/src** e dentro do arquivo **Planning-Menu.js**, adicionar as opções (adaptadas a opção corrente) na variável **validAlgorithms**.

```

1  {
2      options: ["Algorithms.NOVA_OPcao", "UnbiasDataAlgorithms.
3      NOTHING", "UnbiasPostProcAlgorithms.NOTHING"],
4      labels: ["Nome da nova opcao", "Sem metodo", "Sem metodo"],
5      selected: true
6  },
7  {
8      options: ["Algorithms.NOVA_OPcao", "UnbiasDataAlgorithms.
9      NOTHING", "UnbiasPostProcAlgorithms.EQUALIZED_ODDS"],
10     labels: ["Nome da nova opcao", "Sem metodo", "Equalized Odds"
11     ],
12     selected: true
13 },
14 {
15     options: ["Algorithms.NOVA_OPcao", "UnbiasDataAlgorithms.
16     NOTHING", "UnbiasPostProcAlgorithms.CALIBRATED_EQUALIZED_ODDS"
17     ],
18     labels: ["Nome da nova opcao", "Sem metodo", "Calibrated
19     Equalized Odds"],
20     selected: true
21 },
22 {
23     options: ["Algorithms.NOVA_OPcao", "UnbiasDataAlgorithms.
24     NOTHING", "UnbiasPostProcAlgorithms.
25     REJECT_OPTION_CLASSIFICATION"],
26     labels: ["Nome da nova opcao", "Sem metodo", "Reject Option
27     Classification"],
28     selected: true
29 },
30 {
31     options: ["Algorithms.NOVA_OPcao", "UnbiasDataAlgorithms.
32     REWEIGHING", "UnbiasPostProcAlgorithms.NOTHING"],
33     labels: ["Nome da nova opcao", "Reweighing", "Sem metodo"],
34     selected: true
35 },
36 {
37     options: ["Algorithms.NOVA_OPcao", "UnbiasDataAlgorithms.
38     DISPARATE_IMPACT_REMOVER", "UnbiasPostProcAlgorithms.NOTHING"],
39     labels: ["Nome da nova opcao", "Disparate Impact Remover", "
40     Sem metodo"],
41     selected: true
42 },
43 {
44     options: ["Algorithms.NOVA_OPcao", "UnbiasDataAlgorithms.
45     OPTIMIZED_PREPROCESSING", "UnbiasPostProcAlgorithms.NOTHING"],
46     labels: ["Nome da nova opcao", "Optimized Preprocessing", "
47     Sem metodo"],
48     selected: true
49 },
50 {
51     options: ["Algorithms.NOVA_OPcao", "UnbiasDataAlgorithms.
52     LEARNING_FAIR_REPRESENTATIONS", "UnbiasPostProcAlgorithms.
53     NOTHING"],
54     labels: ["Nome da nova opcao", "Learning Fair Representations

```

```
39     ", "Sem metodo"],  
40     selected: true  
    }
```