

FairPEK - Documentação

Manutenção

November 7, 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 Arquitetura do Pipeline e Framework

O Pipeline 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,
23            'y_test': y_test,
24            'checksum': self.input['checksum']
25        }
```

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 Pipeline

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

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 Pipeline

1. Neste sistema, a parte do Pipeline se encontra na pasta **src/pipeline**. Dentro dela, os Pipes que armazenam os conjuntos de dados se encontram na pasta **processors/preprocessors/data**.
2. 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__()

```

3. Criar um novo arquivo.
4. 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": [1.1, 0.05, 0]
13    }
14
15    def __init__(self):
16        super().__init__()

```

5. 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_ok = df[df['loan_status'] == 'Fully Paid'].
8         df_sample = pd.concat([df_sample_ok, df_sample_nok])
9
10        df_x = df_sample.drop('loan_status', axis=1)
11        df_y = pd.DataFrame(df_sample.loan_status)
12
13        return df_x, df_y
```

6. No 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
```

7. No 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.GERMAN_CREDIT and
7         preprocessor == Preprocessors.FOREIGN) or \
8         (dataset == Datasets.LENDINGCLUB and preprocessor
9         == Preprocessors.INCOME)
```

8. No 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

```

9. No 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'])

```

10. Adicionar uma nova condição com a opção adicionada

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.
9            items()))[1]

```

2. No 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,
9                          indexes.items()))[1]

```

3.1.4 Interface (Frontend)

1. Neste sistema, a parte do Backend se encontra na pasta **ml-ui/src**. Dentro dela, no arquivo **Auto-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Pipeline 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         Dataset</MenuItem>
12     <MenuItem value={'Datasets.LENDINGCLUB'}>Lendingclub Dataset
13         </MenuItem>
14 </Select>

```

2. No arquivo **Manual-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Pipeline 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         Dataset</MenuItem>
12     <MenuItem value={'Datasets.LENDINGCLUB'}>Lendingclub Dataset
13         </MenuItem>
14 </Select>

```

3. No arquivo **Manual-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Pipeline 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    >
34      <MenuItem value={'Preprocessors.INCOME'}>Renda (-1
35      Salario Minimo/1+ Salarios Minimos)</MenuItem>
36    </Select>
37  }
38  <FormHelperText>Atributo protegido para medir justica</
39  FormHelperText>
40 </FormControl>

```

4. No arquivo **Manual-Pipeline-Menu.js** adicionar a condição para a opção colocada na etapa de Pipeline 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 Pipeline

1. 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
```

2. No 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]
```

3. Os Filters que armazenam os conjuntos de dados se encontram na pasta **processors/inprocessors**. Nesta pasta, criar um novo arquivo.
4. Os Filters que executam os algoritmos se encontram na pasta **processors/inprocessors/inproc_algorithms**. Nesta pasta, criar um novo arquivo.
5. 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
          .weighed else self.gradient_boost()
3
4         self.output = {
5             'y_pred': y_pred,
6             'scores': scores
7         }

```

7. 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,
          unbias_data_algorithm):
2         weighed_algorithm = unbias_data_algorithm ==
          UnbiasDataAlgorithms.REWEIGHING or \
3             unbias_data_algorithm ==
          UnbiasDataAlgorithms.LEARNING_FAIR_REPRESENTATIONS
4
5         process_options = [
6             (Algorithms.LOGISTIC_REGRESSION,
          LogisticRegressionFilter(weighed=weighed_algorithm)),
7             (Algorithms.RANDOM_FOREST, RandomForestFilter(
          weighed=weighed_algorithm)),
8             (Algorithms.GRADIENT_BOOST, GradientBoostFilter(
          weighed=weighed_algorithm)),
9             (Algorithms.SUPPORT_VECTOR_MACHINES, SVMFilter(
          weighed=weighed_algorithm)),
10            (UnbiasInProcAlgorithms.PREJUDICE_REMOVER,
          PrejudiceRemoverFilter()),
11            (UnbiasInProcAlgorithms.ADVERSARIAL_DEBIASING,
          AdversarialDebiasingFilter()),
12            (UnbiasInProcAlgorithms.
          EXPONENTIATED_GRADIENT_REDUCTION,
          ExponentiatedGradientReductionFilter(algorithm=Algorithms.
          GRADIENT_BOOST)),
13            (UnbiasInProcAlgorithms.RICH_SUBGROUP_FAIRNESS,
          RichSubgroupFairnessFilter(algorithm=Algorithms.
          DECISION_TREE)),
14            (UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER,
          MetaFairClassifierFilter()),
15            (UnbiasInProcAlgorithms.GRID_SEARCH_REDUCTION,
          GridSearchReductionFilter(algorithm=Algorithms.
          RANDOM_FOREST))
16        ]
17
18        for option, filter in process_options:
19            if algorithm == option:
20                prediction_pipe = process_pipe >= filter ==
          self.new_pipe()

```

```

21         break
22
23     return prediction_pipe

```

8. No arquivo **pipeline_start.py**, adicionar as opções (adaptadas a opção corrente) na variável **process_options**.

```

1     process_options = [
2         (Algorithms.NOVA_OP CAO , UnbiasDataAlgorithms.NOTHING ,
3         UnbiasPostProcAlgorithms.NOTHING),
4         (Algorithms.NOVA_OP CAO , UnbiasDataAlgorithms.NOTHING ,
5         UnbiasPostProcAlgorithms.EQUALIZED_ODDS),
6         (Algorithms.NOVA_OP CAO , UnbiasDataAlgorithms.NOTHING ,
7         UnbiasPostProcAlgorithms.CALIBRATED_EQUALIZED_ODDS),
8         (Algorithms.NOVA_OP CAO , UnbiasDataAlgorithms.NOTHING ,
9         UnbiasPostProcAlgorithms.REJECT_OPTION_CLASSIFICATION
10        ),
11        (Algorithms.NOVA_OP CAO , UnbiasDataAlgorithms.
12        REWEIGHING , UnbiasPostProcAlgorithms.NOTHING),
13        (Algorithms.NOVA_OP CAO , UnbiasDataAlgorithms.
14        DISPARATE_IMPACT_REMOVER ,
15        UnbiasPostProcAlgorithms.NOTHING),
16        # (Algorithms.NOVA_OP CAO , UnbiasDataAlgorithms.
17        OPTIMIZED_PREPROCESSING , UnbiasPostProcAlgorithms.NOTHING)
18    ,
19    (Algorithms.NOVA_OP CAO , UnbiasDataAlgorithms.
20    LEARNING_FAIR_REPRESENTATIONS ,
21    UnbiasPostProcAlgorithms.NOTHING)
22    ]

```

9. Adicionar uma nova condição com a opção adicionada

3.2.2 Interface (Backend)

1. 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,

```

```

15         'UnbiasInProcAlgorithms.META_FAIR_CLASSIFIER':
16         106,
17         'UnbiasInProcAlgorithms.ART_CLASSIFIER': 107
18     }
19     return next(filter(lambda a: a[0] == algorithm,
20                       indexes.items()))[1]

```

3.2.3 Interface (Frontend)

1. No arquivo **Manual-Pipeline-Menu.js**, adicionar a opção colocada na etapa de Pipeline 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. No 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 },

```

```

16 {
17     options: ["Algorithms.NOVA_OPCAO", "UnbiasDataAlgorithms.
18     NOTHING", "UnbiasPostProcAlgorithms.
19     REJECT_OPTION_CLASSIFICATION"],
20     labels: ["Nome da nova opcao", "Sem metodo", "Reject Option
21     Classification"],
22     selected: true
23 },
24 {
25     options: ["Algorithms.NOVA_OPCAO", "UnbiasDataAlgorithms.
26     REWEIGHING", "UnbiasPostProcAlgorithms.NOTHING"],
27     labels: ["Nome da nova opcao", "Reweighting", "Sem metodo"],
28     selected: true
29 },
30 {
31     options: ["Algorithms.NOVA_OPCAO", "UnbiasDataAlgorithms.
32     DISPARATE_IMPACT_REMOVER", "UnbiasPostProcAlgorithms.NOTHING"],
33     labels: ["Nome da nova opcao", "Disparate Impact Remover", "
34     Sem metodo"],
35     selected: true
36 },
37 {
38     options: ["Algorithms.NOVA_OPCAO", "UnbiasDataAlgorithms.
39     LEARNING_FAIR_REPRESENTATIONS", "UnbiasPostProcAlgorithms.
40     NOTHING"],
41     labels: ["Nome da nova opcao", "Learning Fair Representations
42     ", "Sem metodo"],
43     selected: true
44 }

```