接入说明v2

接入demo内容说明

- `实验流程_情绪_svm` 情绪实验流程使用svm算法模块
- `实验流程_情绪_msct` 情绪实验流程使用msct算法模块
- `实验流程_情绪_模板` 情绪实验流程空白模板
- `实验流程_XX_模板` 某个典型流程空白模板
- `README.pdf *当前文档`

`实验流程_情绪_*` 都为示例，作为接入参考
`实验流程_XX_模板` `XX` 代表某个典型流程模板，如: 分类、连续预测、隐变量分析 等等.

模块接入步骤

0. 阅读接入说明

1. 创建实验流程

1. 完整查看接入说明，参考示例实验
2. 创建新的实验流程:
    1. 复制 `实验流程_XX_模板`
    2. 重命名为 `实验流程_XX_{算法实现}`, 比如 `实验流程_连续预测_Kalman`

2. 实现实验流程

以下操作目录为在上一步中创建的新实验流程目录下 `实验流程_XX_{算法实现}`，并使用 `实验流程_情绪_svm/msct` 举例

目录说明

查看&修改流程定义 `experiment.yaml`

了解需要填空的内容，并确定需要实现哪些模块

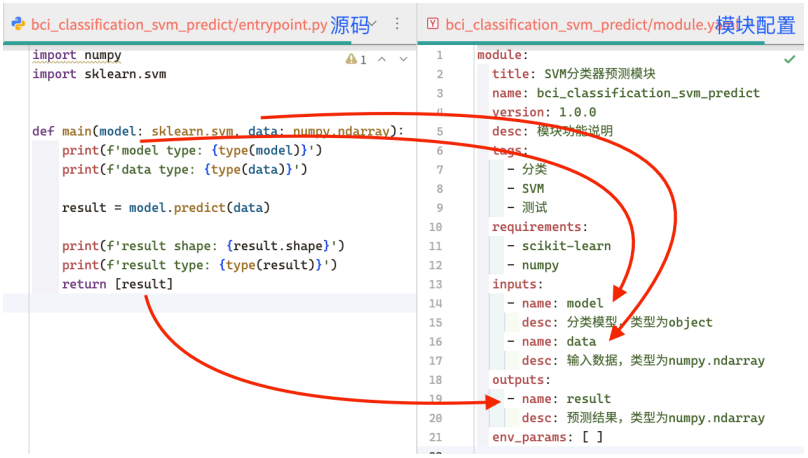
实验流程_情绪_SVM(参考)/experiment.yaml

```yaml
workflow:
  nodes:
    - name: 生成标签
      accepted_into_platform: true # 新增模块默认为真，计算平台将接收并导入。范例模块默认为假，不建议修改配置或源码，如需修改请说明原因。
      outputs:
        "0":
          - 划分数据集:1
    - name: 划分数据集
      kind: normal
      display_name: 划分数据集
      description:
      envs: { }
      module: bci_emotion_split_data
      accepted_into_platform: true # 新增模块默认为真，计算平台将接收并导入。范例模块默认为假，不建议修改配置或源码，如需修改请说明原因。
      outputs:
        "0":
          - 训练SVM模型:0
        "1":
          - 训练SVM模型:1
        "2":
          - SVM预测:1
        "3":
          - 计算准确率:0
          - 计算F1分数:0


    ### 需要被替换的模块
    - name: 训练SVM模型
      kind: normal
      display_name: 训练SVM模型
      description:
      envs:
        PARAM_SVC_KERNEL: "linear"
      module: bci_emotion_svm_fit
      accepted_into_platform: true # 新增模块默认为真，计算平台将接收并导入。范例模块默认为假，不建议修改配置或源码，如需修改请说明原因。
      outputs:
        "0":
          - SVM预测:0
    - name: SVM预测
      kind: normal
      display_name: SVM预测
      description:
      envs: { }
      module: bci_emotion_svm_pred
      accepted_into_platform: true # 新增模块默认为真，计算平台将接收并导入。范例模块默认为假，不建议修改配置或源码，如需修改请说明原因。
      outputs:
        "0":
          - 计算准确率:1
          - 计算F1分数:1


    # ###
    - name: 计算准确率
      kind: normal
      display_name: 计算准确率
      description:
      envs: { }
      module: bci_emotion_accuracy_score
      accepted_into_platform: true # 新增模块默认为真，计算平台将接收并导入。范例模块默认为假，不建议修改配置或源码，如需修改请说明原因。

    - name: 计算F1分数
```

```yaml
# 需要实现的模块
- name: 训练MSCT模型
  kind: normal
  display_name: 训练MSCT模型
  description:
  envs:
    INPUT_PARAM_TRAIN_EPOCH: "50"
    INPUT_PARAM_BATCH_SIZE: "64"
    INPUT_PARAM_LEARNING_RATE: "0.0005"
    INPUT_PARAM_WEIGHT_DECAY: "0.005"
    INPUT_PARAM_FREQUENCY: "250"
    INPUT_PARAM_TIME_FILTER_LENS: "[90, 60, 30, 15, 5]"
    INPUT_PARAM_TRANSFORMER_DEPTH: "4"
    INPUT_PARAM_NUM_CLASSES: "9"
  module: bci_emotion_MSCT_fit # 复制目录modules/bci_module_templat
  accepted_into_platform: true # 新增模块默认为真，计算平台将接收并导入。
  outputs:
    "0":
      - MSCT预测:0
# 需要实现的模块
- name: MSCT预测
  kind: normal
  display_name: MSCT预测
  description:
  envs: { }
  module: bci_emotion_MSCT_predict # 复制目录modules/bci_module_temp
  accepted_into_platform: true # 新增模块默认为真，计算平台将接收并导入。
  outputs:
    "0":
      - 计算准确率:1
      - 计算F1分数:1
```

以上为情绪实验从SVM实现调整为MSCT实验

设计&实现模块

> ## 模块输入输出配置与源码关系
>
> 配置与源码对应关系如图
>
> 
>
>
> 要求如下
>
> - 源码输入数据必须写明对象类型，如: `np.ndarray` 、 `string` 、 `CUSTOM_CLASS` 等等
> - 源码中输出必须为 `list` , 如 `return [A,B,C]` ,如果为空 `return []`
> - **配置**中inputs、outputs中的desc，需格式填写 `{这是什么}，类型为{类型名字}`
> - **配置**中如果输入输出为空 `inputs: [ ]` `outputs: [ ]`

禁止如图所示，对输入为 `None` 进行处理

```
def main(neural_data: numpy.ndarray,
    if neural_data is None:
        data, t = load_data()
        neural_data = data
        target = t
```

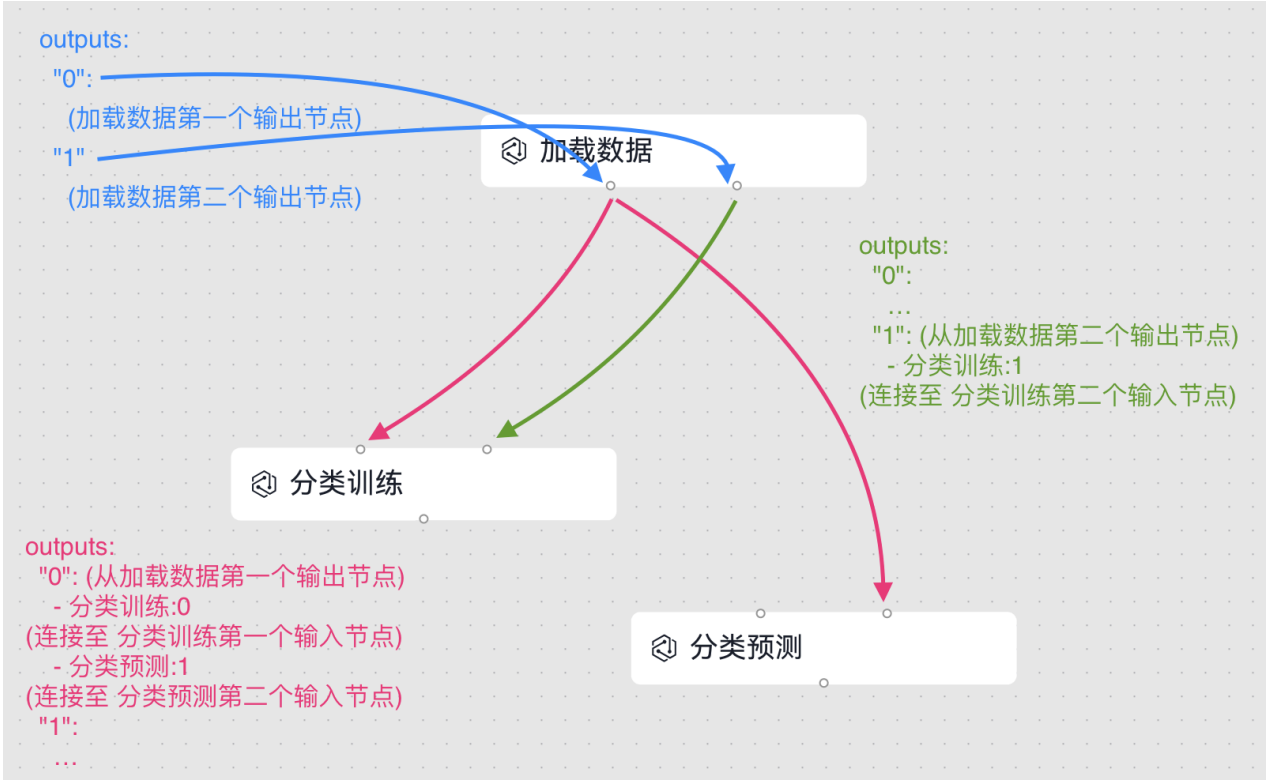
如需单独测试该模块，请使用如下方式

```
if __name__ == "main":
    env_c = int(os.environ.get("INPUT_PARAM_C"))
    env_dfs = os.environ.get("INPUT_PARAM_DECISION_FUNCTION_SHAPE")

    neural_data = load_some_data()
    target = load_some_data_2()

    result = main(neural_data, target)
```

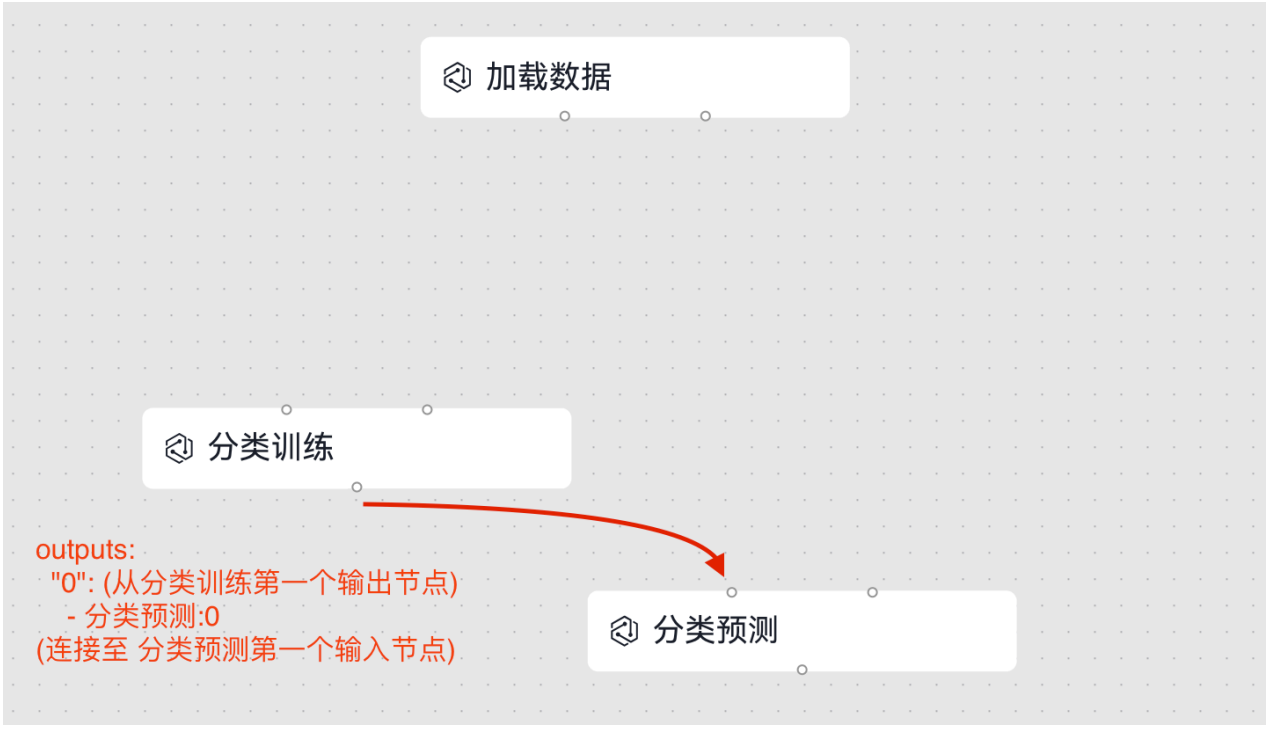
流程定义与输入输出

> ## 实验流程输入输出关系
>
> ```
> version: 1
> workflow:
>   name: 典型流程_分类_实验流程
>   description: ""
>   nodes:
>     - name: 加载数据
>       # 省略
>       outputs:
>         "0":
>           - 分类训练:0
>           - 分类预测:1
>         "1":
>           - 分类训练:1
>     - name: 分类训练
>       # 省略
>       outputs:
>         "0":
>           - 分类预测:0
>     - name: 分类预测
>       # 省略
> ```

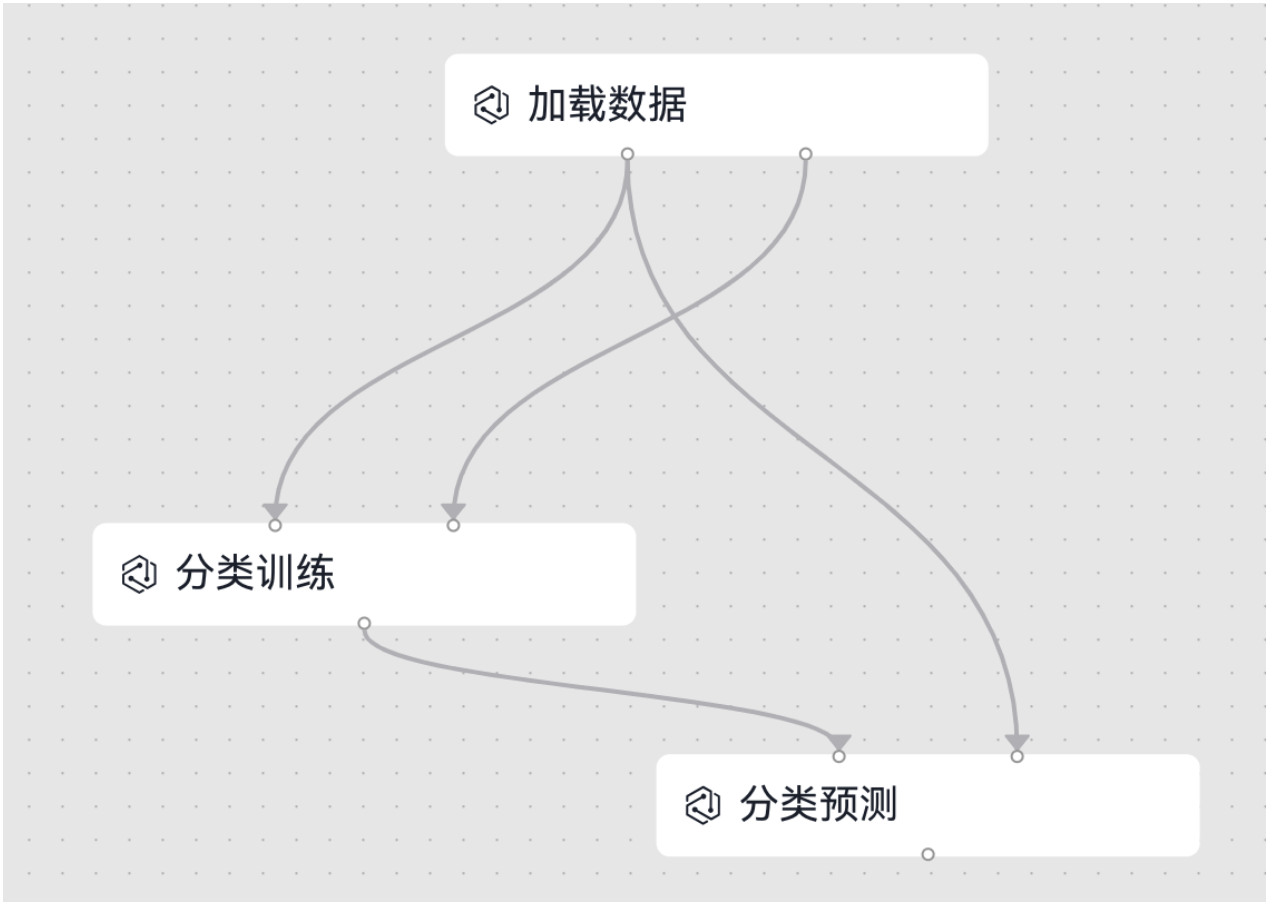
先看 **加载数据** 模块的输出配置，如下图所示



再看 **分类训练** 模块的输出配置，如下图所示



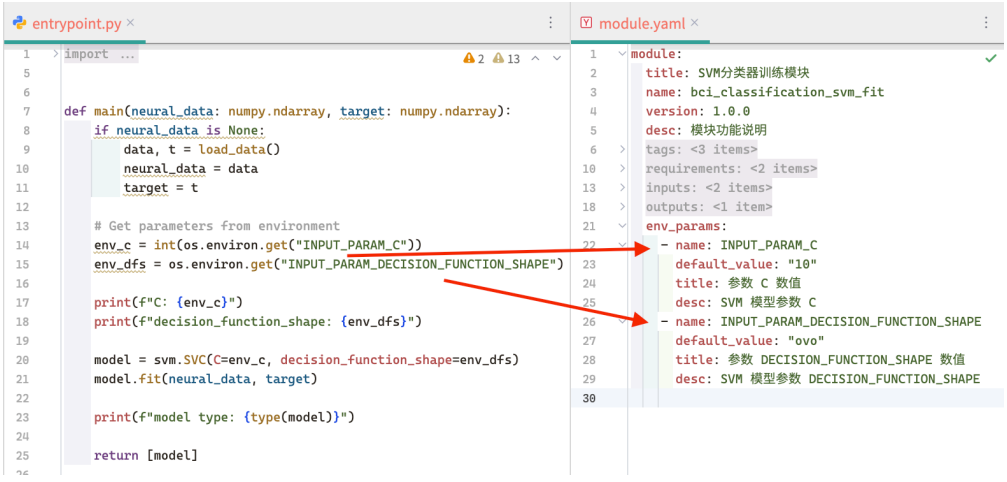
最终配置结果如下图



模块参数

模块参数与源码关系

如图



- 模块运行时，参数将会以环境变量的方式传入
- 环境变量获取为str
- 可以在get中加入默认值 `os.environ.get("INPUT_PARAM_C", "10")`
- 如果需要其他类型请手动转换
- 环境变量名必须是大写 `INPUT_PARAM_XXXX`

流程配置中的模块参数

运行调试流程

已完成实验流程定义`experiment.yaml` 如何配置?
通过命令行，运行实验流程: `python dcp-cli.py run`

提交流程zip

打包ZIP并提交: `python dcp-cli.py pack` *请勿手动打包，打包时会进行检查并剔除不必要的文件*

导出流程环境依赖

导出用于运行实验流程的python环境依赖
`pip freeze >` 实验流程*XX*算法实现A.txt`

3. 流程打包汇总

最终提交

- 实验流程_XX.zip
  - 实验流程_XX_算法实现A.zip
    - experiment.yaml
    - modules
      - bci_module_template
      - {组织}_{流程类型}_{算法实现A1}
        - module.yaml
        - entrypoint.py
        - {其他代码文件}
        - {不要包含测试数据文件}
      - {组织}_{流程类型}_{算法实现A2}
        - ...

- `实验流程_XX_算法实现A.txt`
- `测试数据_XX_算法实现A/`
  - file_1
  - file_2
  - ...
- `实验流程_XX_算法实现B.zip`
- `实验流程_XX_算法实现B.txt`
- `测试数据_XX_算法实现B/`
- `实验流程_XX_算法实现C.zip`
- `实验流程_XX_算法实现C.txt`
- `测试数据_XX_算法实现C/`

以下流程模版设计，模块接入无需查看

设计典型流程空白模板

如果接入demo中已包含你需要接入的典型流程空白模块(实验流程_{某典型流程}_模板)，则返回模块接入

为当前需要接入的典型流程，设计输入输出，可参考 `分类流程` 的三个示例

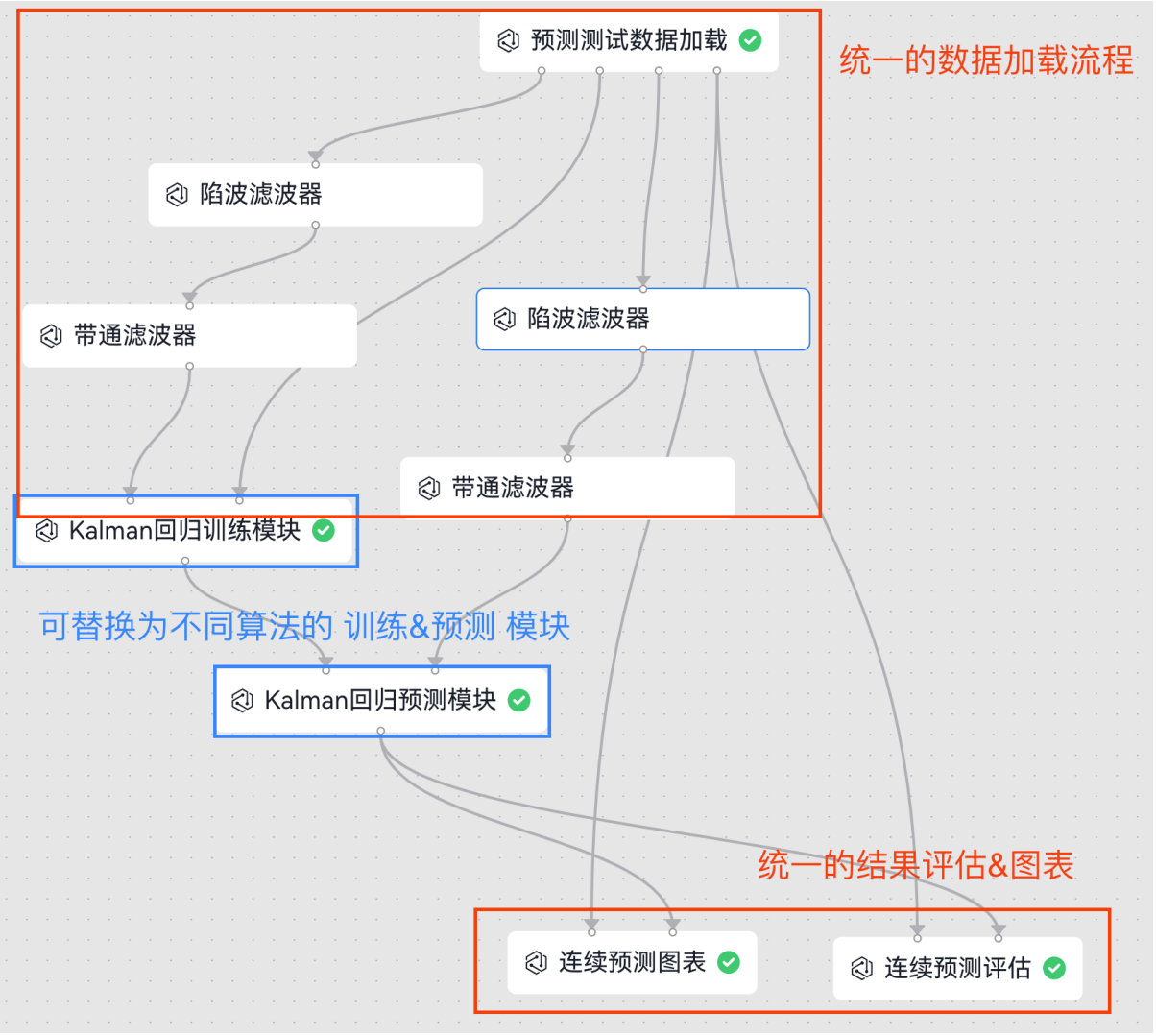
对比 `分类_svm` 和 `分类_rnn` 中 `experiment.yaml` 区别，结合 流程节点间输入输出 说明，修改空白模板中的 `experiment.yaml`
比如当前需要接入 `连续预测` 就修改 `实验流程_XX_模板` 为 `实验流程_连续预测_模板`

该模板中要求

- 统一数据加载模块（如何创建模块?）
- 统一流程节点间输入输出关系 (如何配置实验流程?)
- 流程中节点配置（除数据加载、数据预处理、结果评估、评估图表等）留空，由实现算法模块的人员填写

如图所示 统一数据加载&处理、统一的结果评估&图表
其中的训练&预测模块，应该是可以替换的（流程节点配置中应该留空的部分），需要被替换为不同

算法的训练&预测模块



简单`experiment.yaml`举例，按注释部分进行修改，该分类流程就从SVM实现被替换为Kalman实现

```yaml
version: 1
workflow:
  name: 典型流程_分类_实验流程
  description: ""
  nodes:
    - name: 加载数据
      kind: normal
      display_name: 典型流程_分类_加载数据
      description:
      envs:
        INPUT_MAT_PATH: 'C:\Users\xxx\Downloads\NeuralData1.mat'
        INPUT_LABEL_TXT_PATH: 'C:\Users\xxx\Downloads\label1.txt'
      module: bci_classification_load_sample_data
      accepted_into_platform: false
      outputs:
        "0":
          - 分类训练:0
          - 分类预测:1
        "1":
          - 分类训练:1
    - name: 分类训练
```

```
      kind: normal
      display_name: 分类流程_训练_SVM ## 可替换 如 分类流程_训练_Kalman
      description:
      envs: {} ## 模块参数按实际需求填写
      module: bci_classification_svm_fit  ## 可替换 如
bci_classification_kalman_fit
      accepted_into_platform: true
      outputs:
        "0":
          - 分类预测:0
    - name: 分类预测
      kind: normal
      display_name: 分类流程_预测_SVM  ## 可替换 如 分类流程_预测_Kalman
      description:
      envs: { }  ## 模块参数按实际需求填写
      module: bci_classification_svm_predict  ## 可替换 如
bci_classification_kalman_predict
      accepted_into_platform: true
```

最后打包为zip进行分发，比如 `连续预测.zip`，包含

```
- 实验流程_分类_svm
- 实验流程_分类_rnn
- 实验流程_分类_模板
- READEME.pdf
- 实验流程_连续预测_模板
```

此处举例 后续文档依旧使用 `实验流程_XX_模板`

实验流程配置

实验流程配置说明

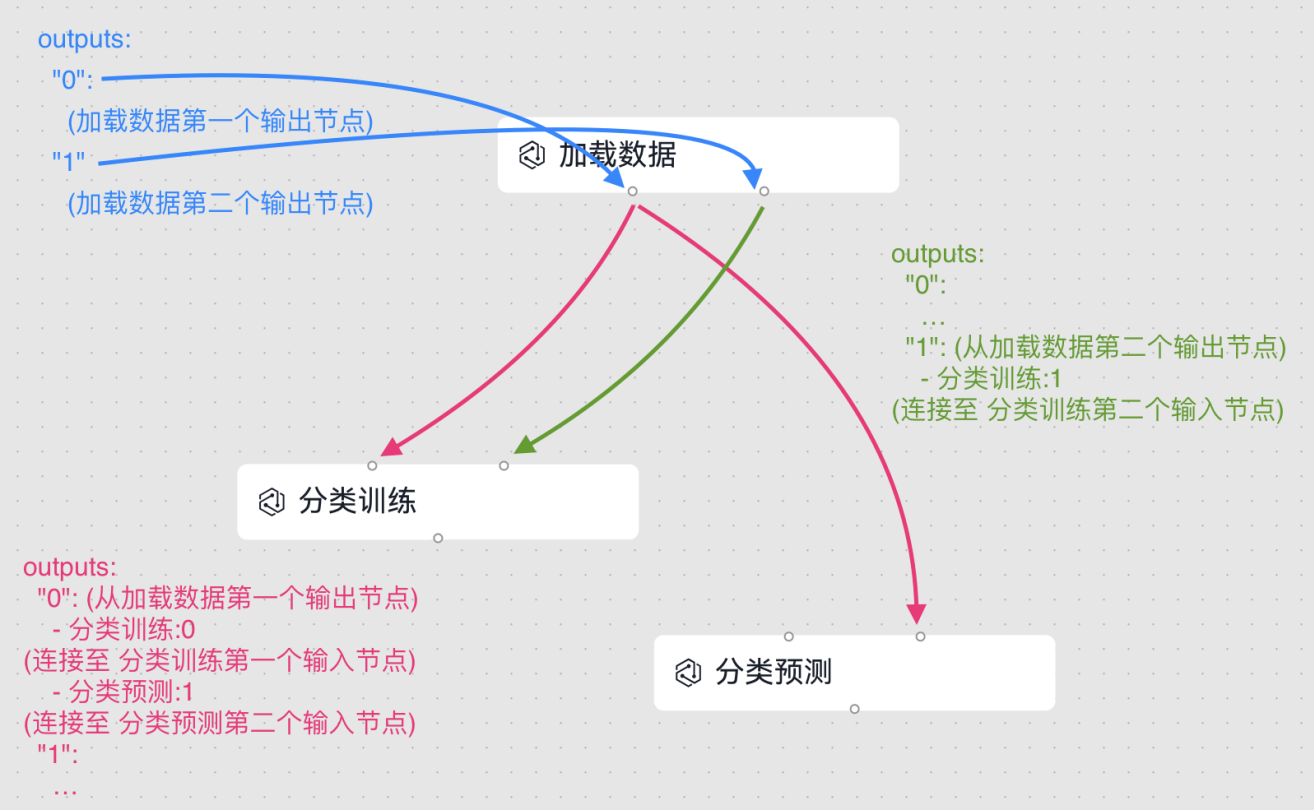
```yaml
version: 1
workflow:
  name: 实验流程名称
  description: 实验流程描述
  nodes:  # 实验流程节点列表
    - name: 流程节点名(将用于输入输出关联)
      kind: normal # 流程节点类型 常规python脚本为normal
      display_name: 流程节点显示名称(平台)
      description: 流程节点说明
      envs:  # 环境变量Key: 环境变量Value
        A: 123
        B: 456
      module: bci_classification_load_sample_data  # 节点执行模块名
      accepted_into_platform: false # 是否导入模块至实验计算平台
      outputs:
        "0":
          - bci_classification_svm_fit_NTU:0
```

```
            - bci_classification_svm_predict_ZGY:1
          "1":
            - bci_classification_svm_fit_NTU:1
# 省略
```

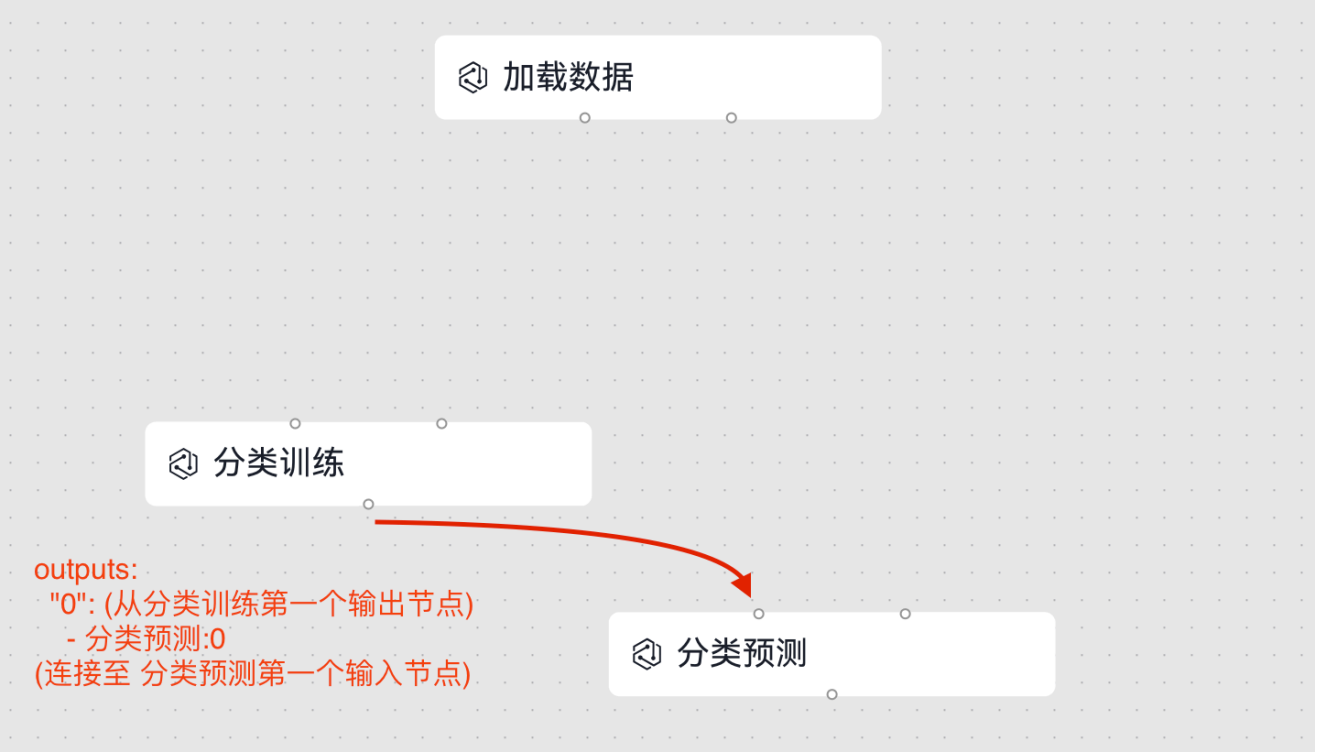
实验流程输入输出关系

```
version: 1
workflow:
  name: 典型流程_分类_实验流程
  description: ""
  nodes:
    - name: 加载数据
      # 省略
      outputs:
        "0":
          - 分类训练:0
          - 分类预测:1
        "1":
          - 分类训练:1
    - name: 分类训练
      # 省略
      outputs:
        "0":
          - 分类预测:0
    - name: 分类预测
      # 省略
```

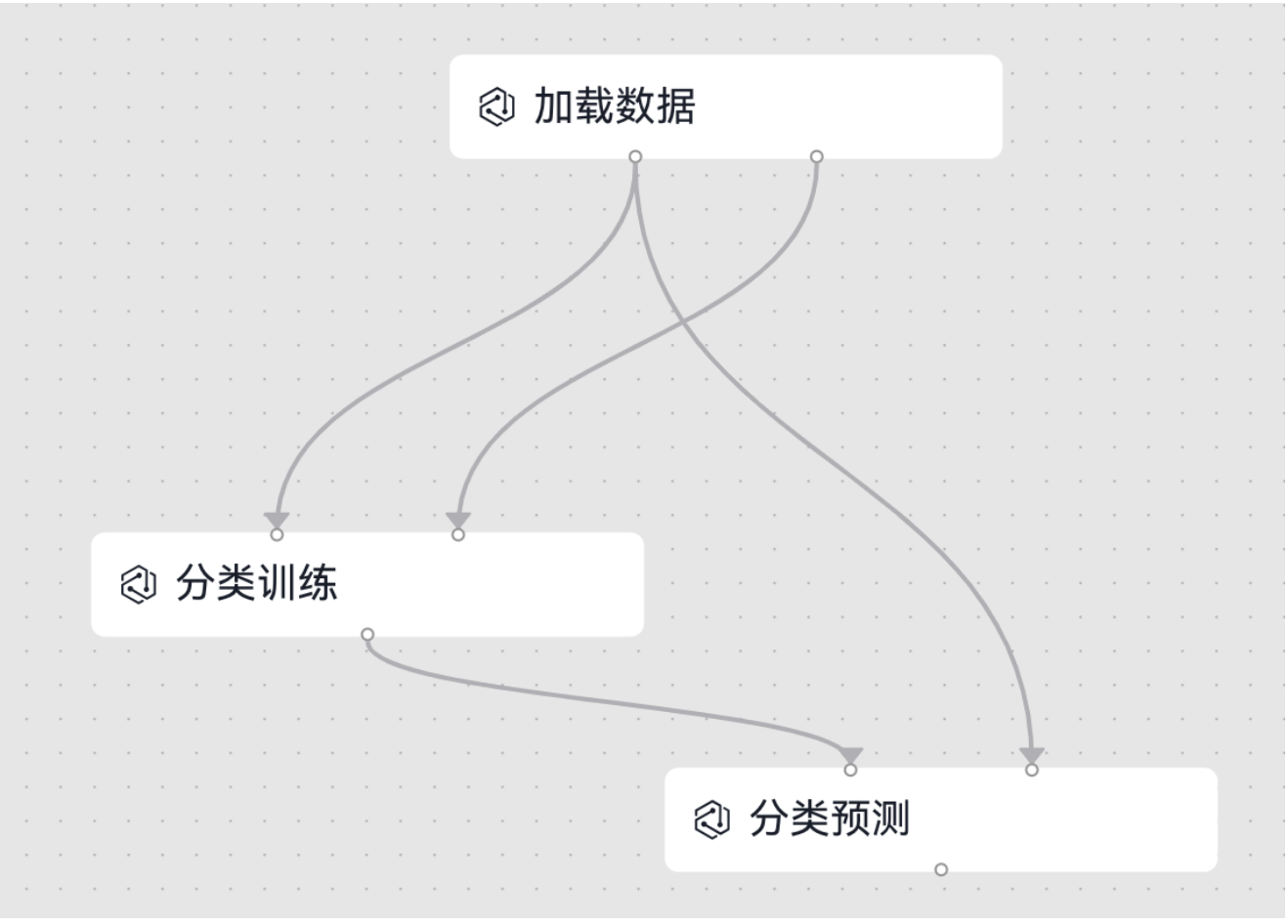
先看 **加载数据** 模块的输出配置，如下图所示



再看 **分类训练** 模块的输出配置，如下图所示



最终配置结果如下图

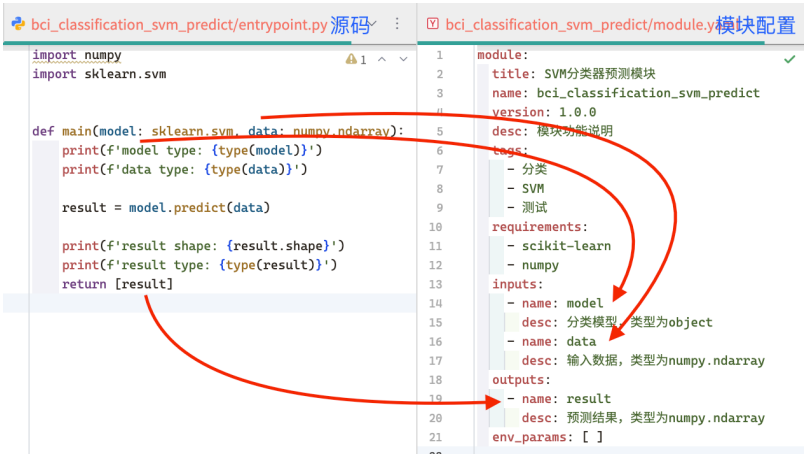


流程配置中的模块参数

模块输入输出配置

模块输入输出配置与源码关系

配置与源码对应关系如图

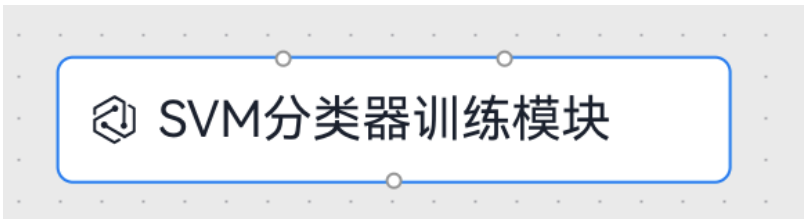


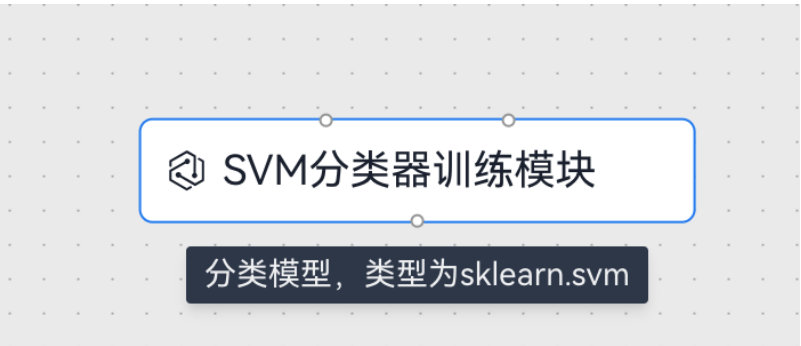
要求如下

- 源码输入数据必须写明对象类型，如: `np.ndarray` 、 `string` 、 `CUSTOM_CLASS` 等等
- 源码中输出必须为 `list` , 如 `return [A,B,C]` ,如果为空 `return []`
- **配置**中inputs、outputs中的desc，需格式填写 `{这是什么}，类型为{类型名字}`
- **配置**中如果输入输出为空 `inputs: [ ]` `outputs: [ ]`

模块输入输出配置与平台页面关系

请认真填写输入/输出，平台中表现如下图(提供上下游数据连接节点，并显示描述)





模块配置

模块配置说明

```yaml
module:
  title: SVM分类器训练模块  # 模块显示名称
  name: bci_classification_svm_fit  # 模块名称 需要与文件夹同名
../bci_classification_svm_fit/
  version: 1.0.0  # 模块版本 默认1.0.0
  desc: 模块功能说明  # 模块简要说明，将会显示在模块列表中
  tags:  # 模块tags
    - 分类
    - SVM
    - 训练
  requirements:  # 模块pypi依赖，requirements.txt 逐行填入即可
    - scikit-learn
    - numpy
  inputs:  # 模块输入定义
    - name: neural_data
      desc: 神经数据，类型为numpy.ndarray
    - name: target
      desc: 标签数据，类型为numpy.ndarray
  outputs:  # 模块输出定义
    - name: model
      desc: 分类模型，类型为sklearn.svm
  env_params:  # 模块参数定义
    - name: INPUT_PARAM_C
      default_value: "10"
      title: 参数 C 数值
      desc: SVM 模型参数 C
    - name: INPUT_PARAM_DECISION_FUNCTION_SHAPE
      default_value: "ovo"
      title: 参数 DECISION_FUNCTION_SHAPE 数值
      desc: SVM 模型参数 DECISION_FUNCTION_SHAPE
```

模块外部依赖安装

常规pypi安装

```yaml
module:
  # ...
```

```yaml
  requirements:
    - scikit-learn
    - numpy
  # ...
```

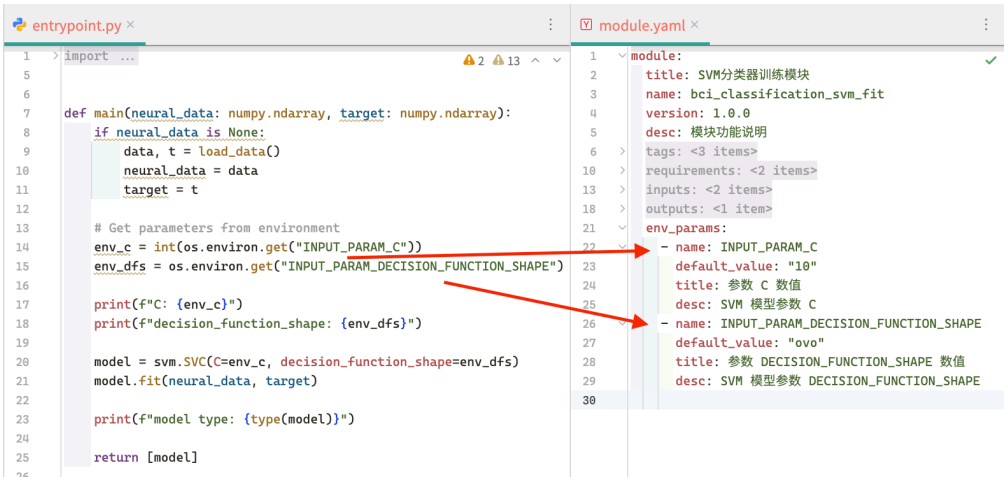
自建代码库打包为whl，放到模块目录下

```yaml
module:
  # ...
  requirements:
    - some_pkg.whl
```

- 实验流程_XX_A
  - modules
    - 算法A
      - `some_pkg.whl`

模块参数与源码关系

如图



- 模块运行时，参数将会以环境变量的方式传入
- 环境变量获取为str
- 可以在get中加入默认值 `os.environ.get("INPUT_PARAM_C", "10")`
- 如果需要其他类型请手动转换
- 环境变量名必须是大写 `INPUT_PARAM_XXXX`

模块参数与平台页面

```yaml
env_params:
  - name: INPUT_PARAM_C
    default_value: "10"
    title: 参数 C 数值
    desc: SVM 模型参数 C
  - name: INPUT_PARAM_DECISION_FUNCTION_SHAPE
    default_value: "ovo"
```

```
    title: 参数 DECISION_FUNCTION_SHAPE 数值
    desc: SVM 模型参数 DECISION_FUNCTION_SHAPE
```

请认真填写 `title` `desc` `default_value` 将用于平台页面， 如图

title: `参数C数值` 、 `参数 DECISION_FUNCTION_SHAPE 数值`
desc: `SVM模型参数 C` 、 `SVM模型参数 DECISION_FUNCTION_SHAPE`
default_value: `10` 、 `ovo`