

PythonとRapidMinerの JupyterNotebook統合

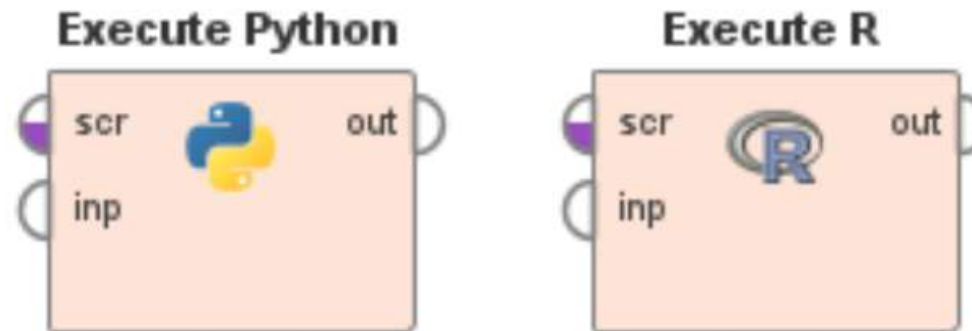
株式会社KSKアナリティクス



1.従来使われてきたPython/Rとの統合

はじめに

- RapidMinerはRとPythonとの統合に力を入れており、RapidMiner StudioでネイティブオペレーターとExecute RかPythonのオペレーターを組み合わせることで統合できます。



1.従来使われてきたPython/Rとの統合

RapidMinerとの統合メリット

- コーディングのみの機械学習にも良さがありますが、組み合わせることで新たな力を発揮します。

コーディングのみの良さ

- ・最先端のオープンソースである
- ・フリーである
- ・専門家レベルの柔軟性がある

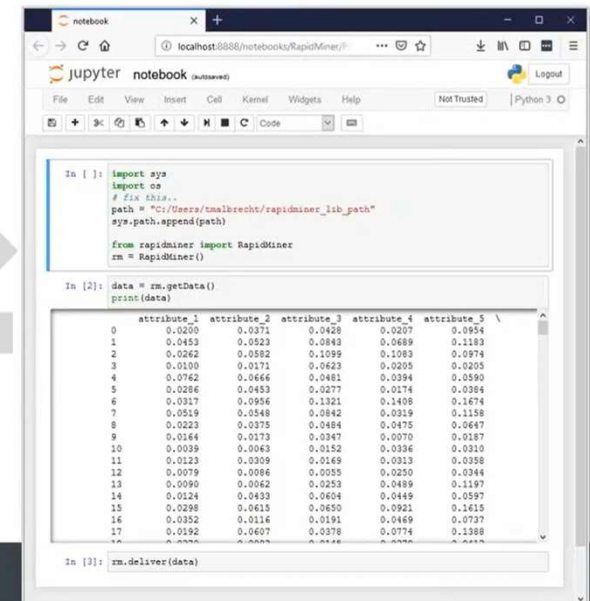
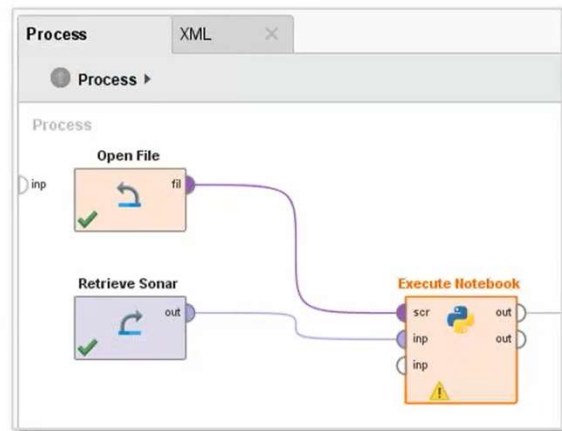
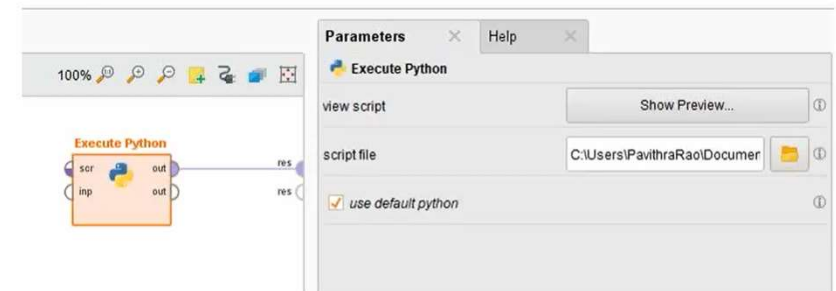
✿ 組み合わせの良さ

- ・試作が早い
- ・手作業の自動化、低スキル化
- ・思考の人と行動の人のコラボ
- ・エンタープライズレベルへのスケール化
- ・標準化できる

2.どのように使用するか

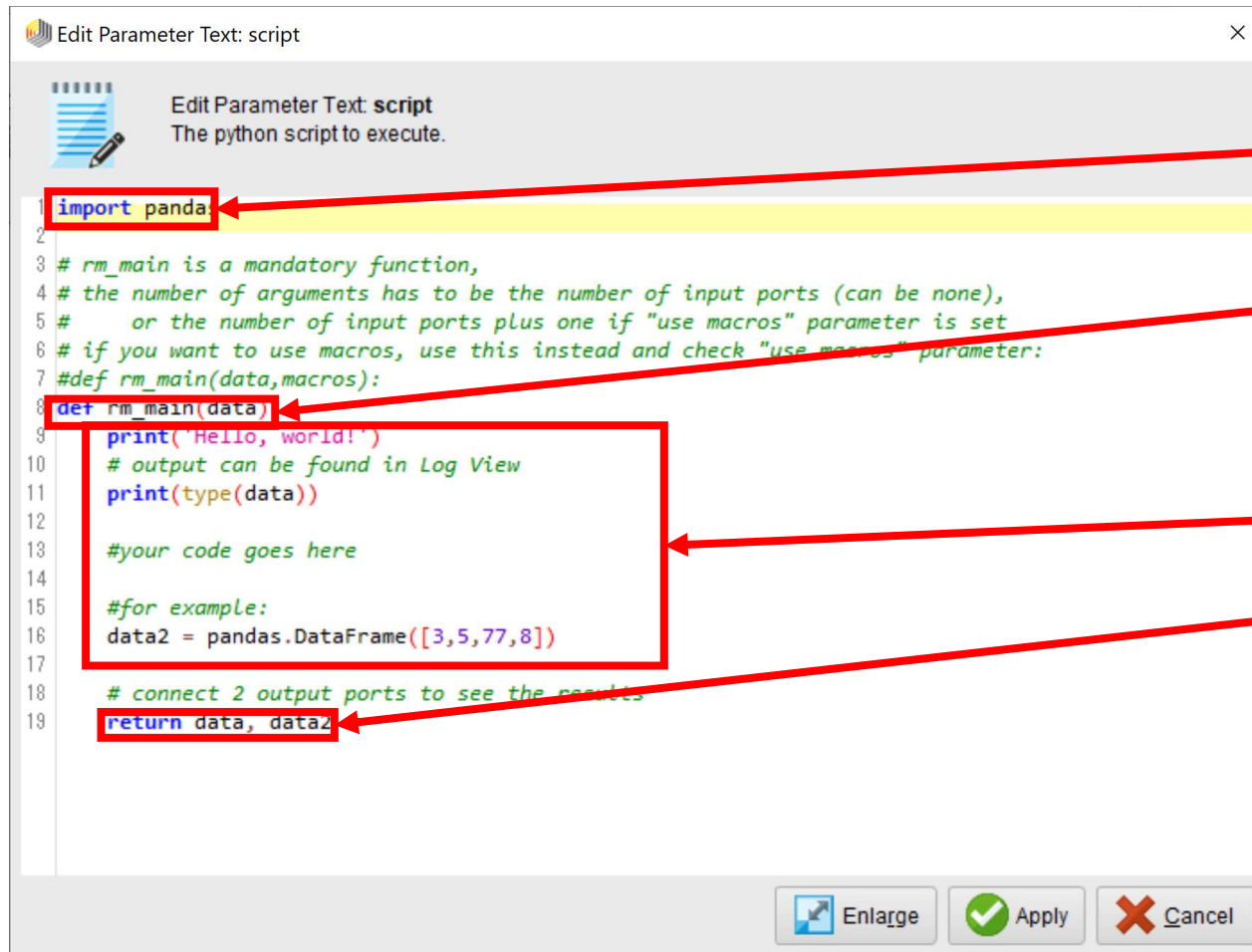
Studioでの利用

- 「Execute Python/R」の「Show Preview」をクリックして、ポップアップにスクリプトを貼り付けます。
または
- アクセス可能な場所に保存されているスクリプトファイル（.pyまたはipynb）「Open File」で呼び出します。



2. どのように使用するか

Studioでの利用



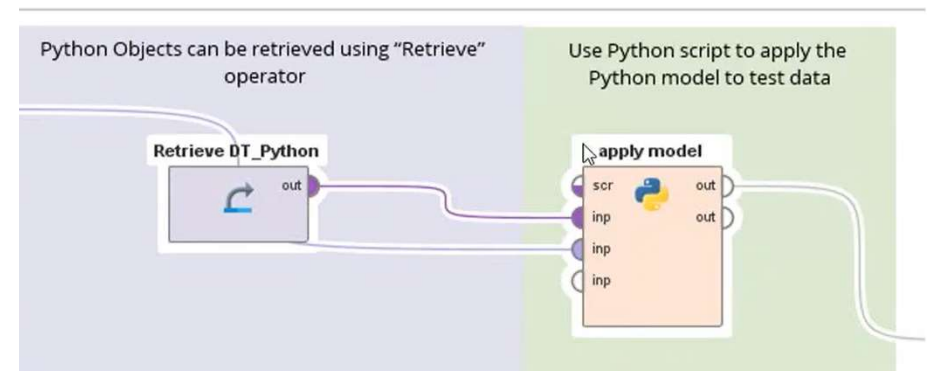
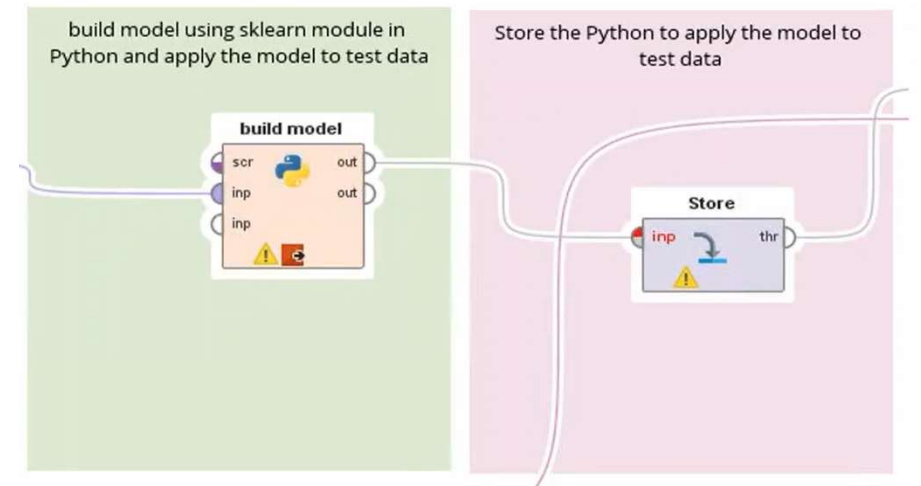
```
1 import pandas
2
3 # rm_main is a mandatory function,
4 # the number of arguments has to be the number of input ports (can be none),
5 # or the number of input ports plus one if "use macros" parameter is set
6 # if you want to use macros, use this instead and check "use macros" parameter:
7 #def rm_main(data, macros):
8 def rm_main(data):
9     print('Hello, world!')
10    # output can be found in Log View
11    print(type(data))
12
13    #your code goes here
14
15    #for example:
16    data2 = pandas.DataFrame([3,5,77,8])
17
18    # connect 2 output ports to see the results
19    return data, data2
```

- ライブラリをインポートする
- DataTables / Pythonオブジェクトを入れる
- メインコード
- DataTables / Pythonオブジェクトをリターンする

2. どのように使用するか

使用にあたって

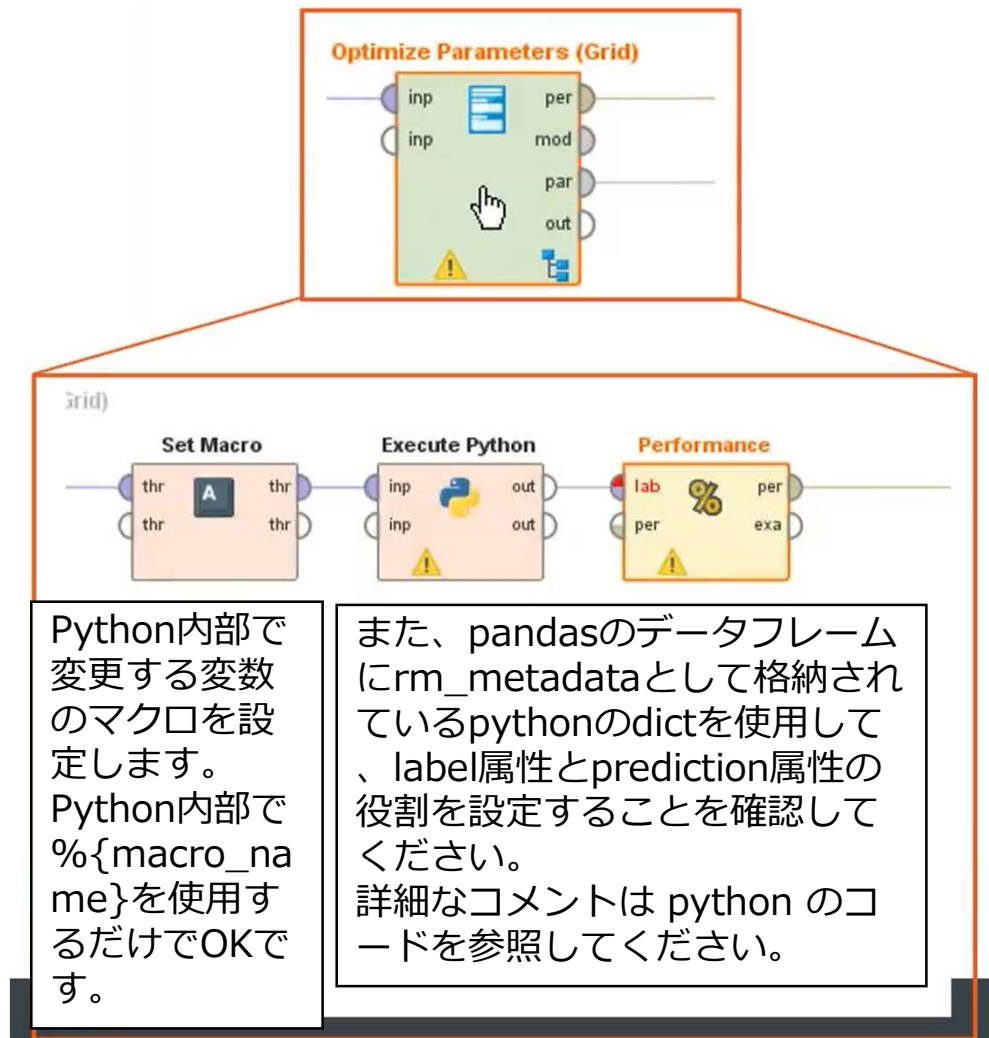
- RapidMinerテーブルは自動的にNumpy配列に変換されます。
- Pythonオブジェクトは「Store」オペレータを使用して保存することができます
例：Pythonで作ったモデルを保存
- Pythonオブジェクトは、「Retrieve」オペレータを使用して取得することができます
例：以前に作成したモデルを取得
- Pythonオブジェクトは他のPythonスクリプトオペレータに接続できます。
例：モデルを作成し、別のPythonスクリプトに渡して、独立したテストデータセットで評価する



3. Pythonモデルの統合

Pythonモデルの最適化

- Pythonモデルのパラメータは通常のRapidminerモデルと同様に、「Optimize Parameter」オペレータで最適化ができます。
- 「Optimize Parameters」のサブプロセス内に「Set Macro」と「Execute Python」と「Performance」オペレーターを配置します。



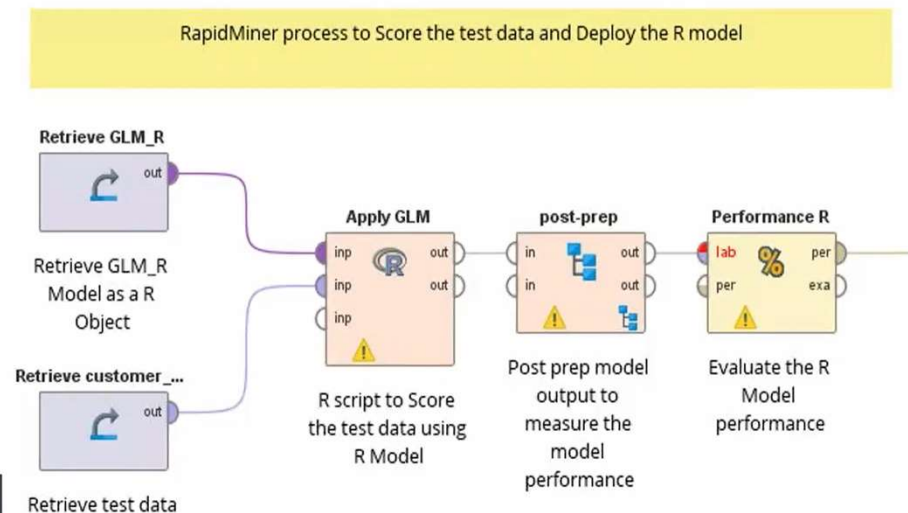
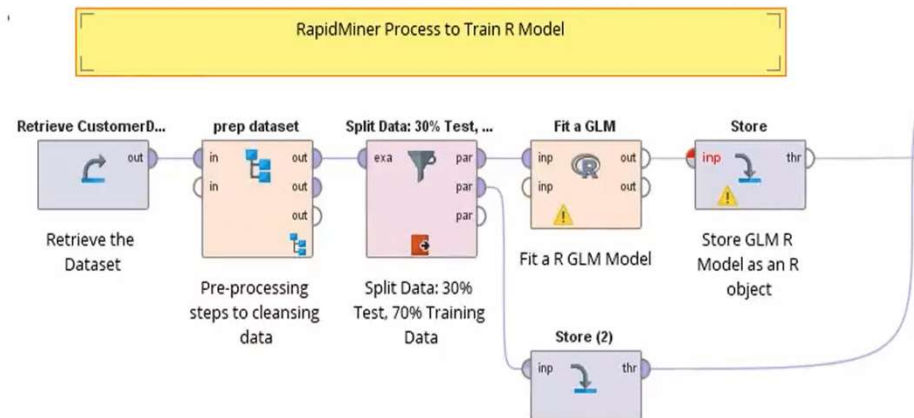
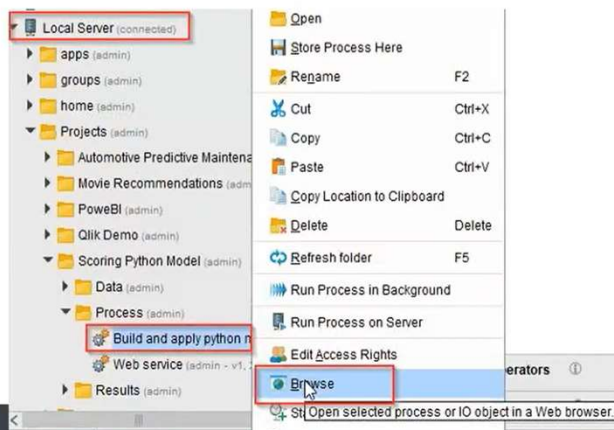
3. Pythonモデルの統合

Webサービスとしてデプロイ

- Python / Rモデルをトレーニングし、モデルにテストデータを適用するプロセスを作成します
- このRapidMinerプロセスをRapidMiner AIHubリポジトリに保存してください

プロセスをWebサービスとして出力する手順

1. ウェブサービスとしてホストされるべき RapidMinerプロセスを選択します。
2. 右クリック→ ブラウズ (Browse)



3. Pythonモデルの統合

Webサービスとしてデプロイ

(前ページより続き)

3. サーバーUIには、プロセスをWebサービスとして出力するオプションがあります。
4. 必要に応じてデフォルト設定を変更します。
5. プロセスで設定されたマクロまたはパラメータはすべて自動的にコピーされます。
6. Webサービスを送信
7. 必要に応じてWebサービスのテストを実行します。

Local Service

Data source: /Projects/Scoring Python Model/Process/Web service

Output format: Table

MIME Type: text/html

Cache input: ☐

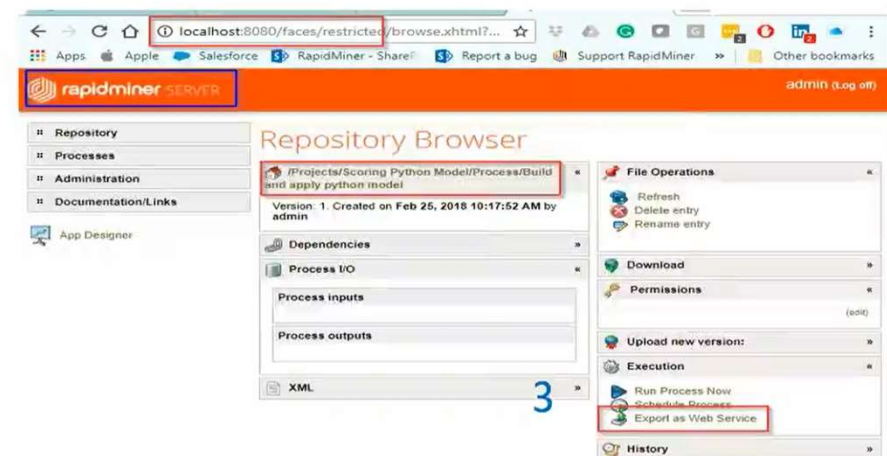
Parameter binding

URL query parameter	Target (macro/operator parameter)	Mandatory
Cust_ID	Cust_ID	☒

Format parameters

column_widths	1000
fragment_only	☐
raw_html	☐
css_location	
tooltip_attribute	

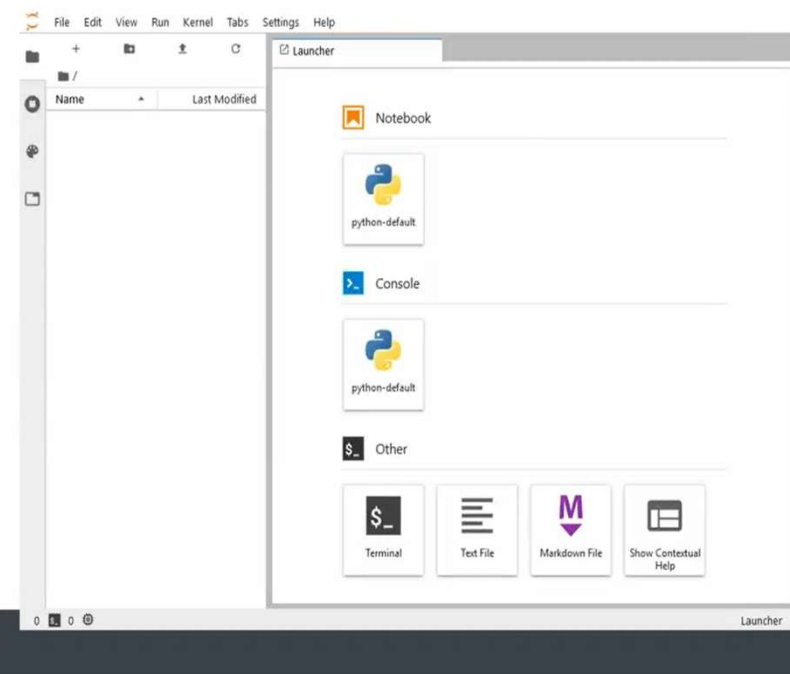
Submit



4. RapidMinerとJupyterHubの組み合わせ

JupyterHubとの連携

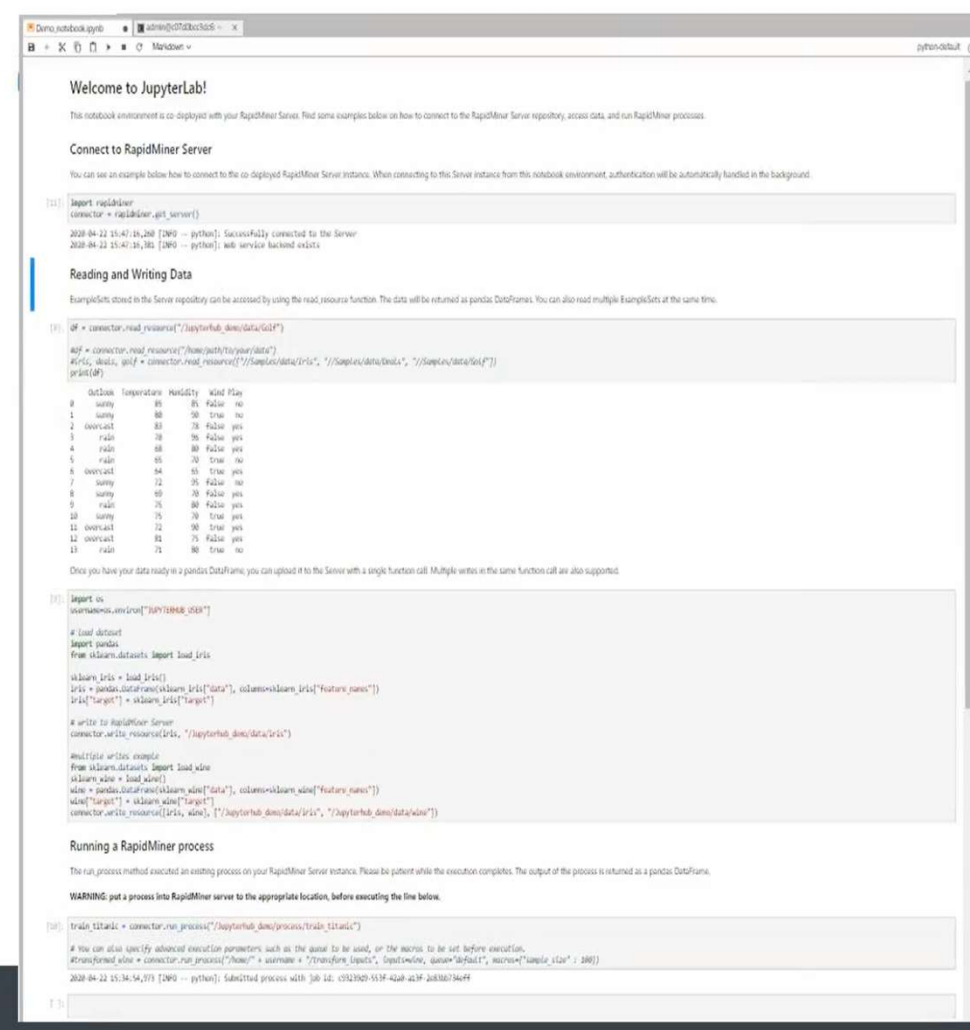
- JupyterHubはRapidMinerプラットフォームに直接組み込まれるようになりました。
 - ユーザーは一元管理および統治された場所でPythonコーディングができます。
 - コードベースの作業は視覚的なワークフローベースのプロジェクトおよび自動化されたデータサイエンスと相互運用可能です。
- ※RapidMiner AIHubの導入が必要です。



4. RapidMinerとJupyterHubの組み合わせ

JupyterHubインターフェースの操作

- RapidMiner AIHubリポジトリのデータ、プロセス、結果などにアクセス出来ます
- JupyterNotebookからRapidMinerワークフローを実行出来ます。
- Notebookからプロセスにリアルタイムの値を渡して、これらのプロセスの実行方法を制御出来ます



```
import rapidminer
connector = rapidminer.get_server()

2020-04-22 15:47:15,208 [INFO - python]: Successfully connected to the Server
2020-04-22 15:47:15,305 [INFO - python]: web service backend exists

Reading and Writing Data
ExampleSets stored in the Server repository can be accessed by using the read_resource function. The data will be returned as pandas DataFrames. You can also read multiple ExampleSets at the same time.

[1]: connector.read_resource("/jupyterhub_demo/data/iris")

def connector_read_resource(homepath/to/your/data):
    # /iris, /demo, /url = connector.read_resource("/iris", "/Samples/data/iris", "/Samples/data/iris")
    print(df)

Outlook  Temperature  Humidity  Wind  Play
0  sunny           85         86  false  no
1  sunny           88         80  true   no
2  overcast        83         78  false  yes
3  rain            76         76  false  yes
4  rain            68         80  false  yes
5  rain            65         70  true   no
6  overcast        64         65  true   yes
7  sunny           72         65  false  no
8  sunny           69         70  false  yes
9  rain            75         80  false  yes
10 sunny           75         70  true   yes
11 overcast        72         90  true   yes
12 overcast        81         75  false  yes
13 rain            71         80  true   no

Once you have your data ready in a pandas DataFrame, you can upload it to the Server with a single function call. Multiple writes in the same function call are also supported.

[1]: import os
overname=os.getenv("JUPYTERHUB_USER")

# load dataset
import pandas
from sklearn.datasets import load_iris

sklearn_iris = load_iris()
iris = pandas.DataFrame(sklearn_iris["data"], columns=sklearn_iris["feature_names"])
iris["target"] = sklearn_iris["target"]

# write to RapidMiner Server
connector.write_resource(iris, "/jupyterhub_demo/data/iris")

# modify iris, example
from sklearn.datasets import load_wine
sklearn_wine = load_wine()
wine = pandas.DataFrame(sklearn_wine["data"], columns=sklearn_wine["feature_names"])
wine["target"] = sklearn_wine["target"]
connector.write_resource(iris, wine), ["/jupyterhub_demo/data/iris", "/jupyterhub_demo/data/wine"]

Running a RapidMiner process
The run_process method executed an existing process on your RapidMiner Server instance. Please be patient while the execution completes. The output of the process is returned as a pandas DataFrame.

WARNING: put a process into RapidMiner server to the appropriate location, before executing the line below.

[1]: train_title = connector.run_process("/jupyterhub_demo/process/train_title")

# You can also specify advanced execution parameters such as the user to be used, or the macros to be set before execution.
# transformed_wine = connector.run_process("/home/" + username + "/transform/inputs", inputname="demo/iris", macroval["sample_size" : 500])
2020-04-22 15:34:54,771 [INFO - python]: Submitted process with job id: c5f2920-543f-42ab-423f-2d8807364ff
```

4. RapidMinerとJupyterHubの組み合わせ

Python環境の管理

- 簡単にPython環境管理が出来ます

主な利点:

- 管理費用の削減
- 管理エラーのリスクの軽減
- Pythonコードを含むデプロイされたRapidMinerプロセスの完璧な実行を保証します。

The screenshot shows the 'RapidMiner Python Environment Manager' web interface. It has a sidebar with links to 'Python environments', 'WebUI Access control', and 'WebUI Certificates'. The main content area is titled 'Python environments' and explains that users can create and manage Python environments on all Job Agents. It lists three options: modifying the base environment, creating a new one, or exporting the current one. Below this is a form to upload an 'Environment yml file' with a 'Choose File' button and a 'Submit' button. At the bottom, there are sections for 'Registered environments' and 'Archived environments'. The 'Registered environments' section contains a table with one entry named 'base'.

Environment name	Uploaded at	Status	Actions
base	2020. 03. 05. 06:18:09	All 1 successful.	Download Details