INTERNAL

SAP Data Intelligence hands-on exercises

This document will guide you step-by-step through the process of training and implementing association rules to produce a recommendation analysis using SAP HANA ML in SAP DI.

THE BEST RUN SAP

www.sap.com/contactsap

THE BEST RUN

Table of Contents

DISCLAIMER

The information shared in this document is confidential and proprietary to SAP and may not be disclosed without the permission of SAP. All functionality presented here is subject to change and may be changed by SAP at any time for any reason without notice.

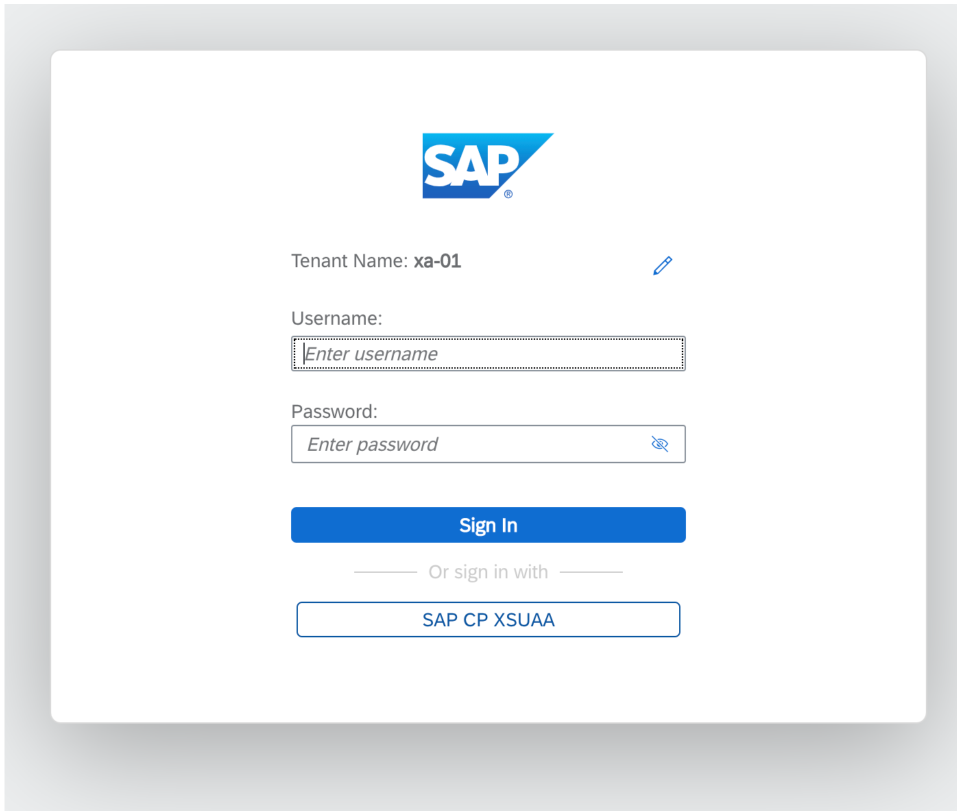
OBJECTIVE

The objective of this exercise is to give you an overview of how you can use the machine learning capabilities in SAP Data Intelligence. We will use the association rules APRIORI algorithm available in the SAP HANA ML PAL library.

SCENARIO

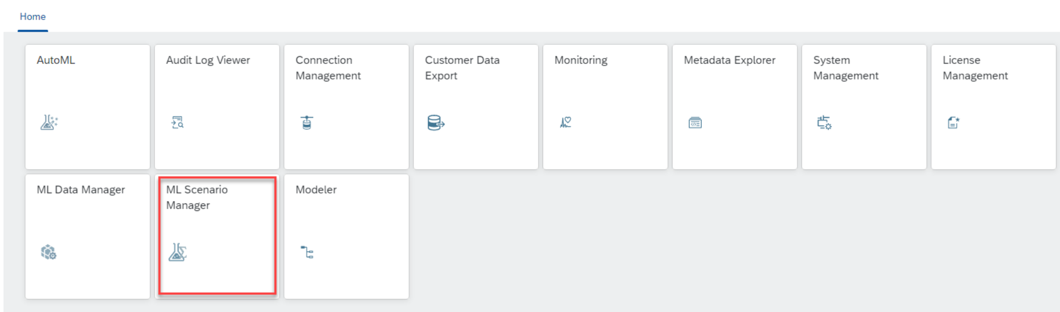
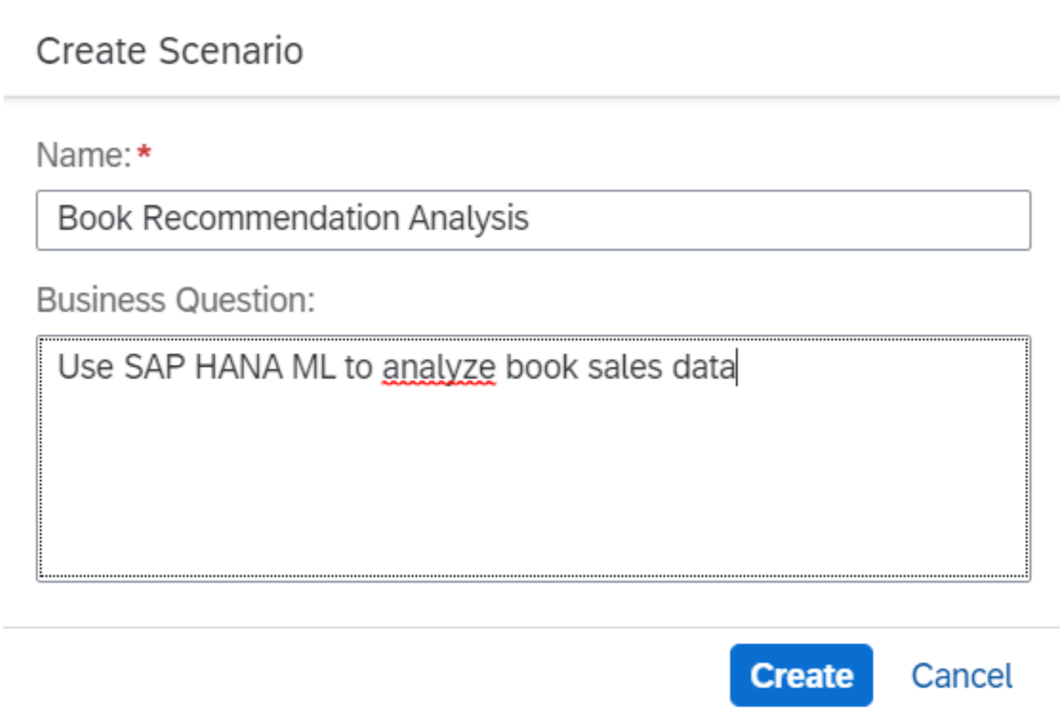
In this scenario, we want to build a book recommendation algorithm, using as input data source the history of book sales for a fictitious book shop having about 10 years of activity (2011 to 2021). The sales data are stored in a Data Warehouse Cloud instance. We will perform market-basket analysis on the historical combinations of books purchased and build association rules that will be used to make book recommendations.

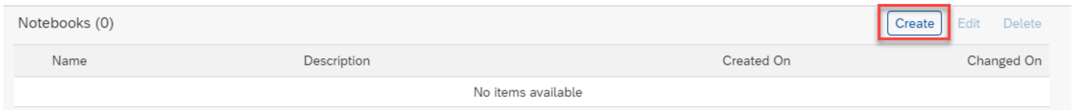
ENVIRONMENT ACCESS

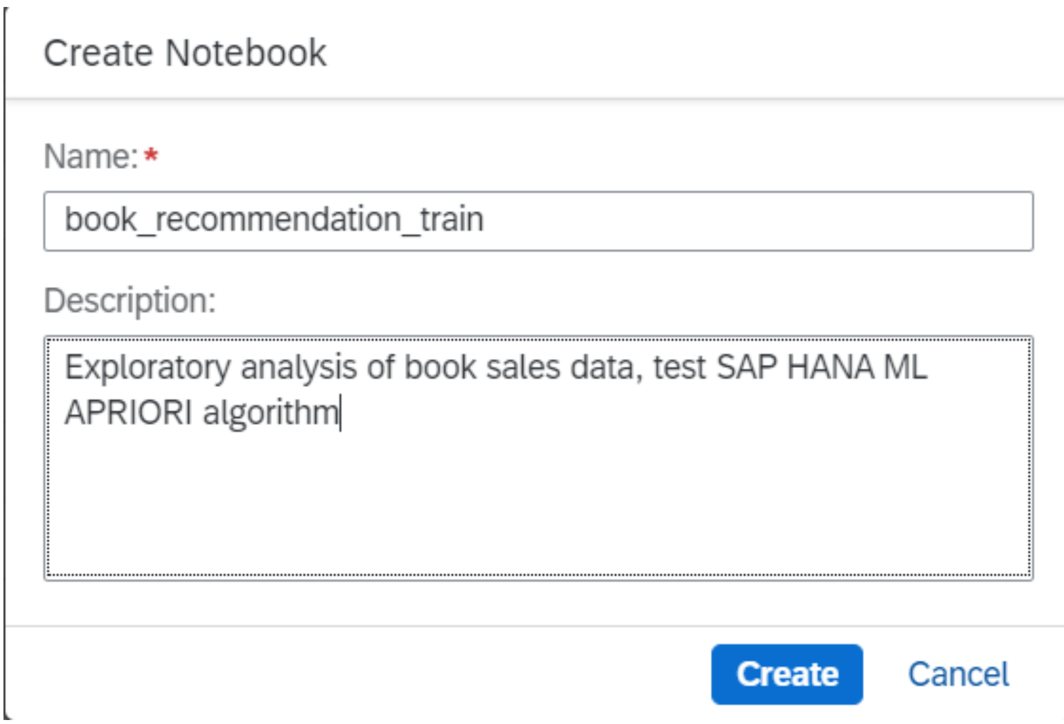
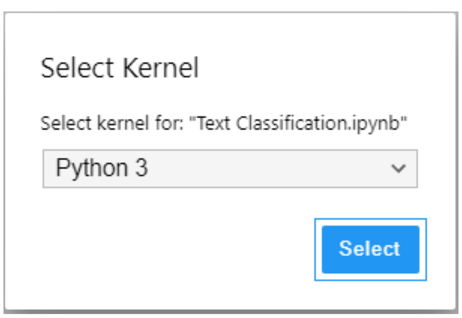
Explanation	Screenshot
In order to open the SAP Data Intelligence application, follow the URL and credentials provided to you by the instructors in the Teams channel you have been invited to. Usually it is **Microsoft Teams > General (Channel) > System Access (Tab) > SAP Data Intelligence Cloud (Section)**.	

STEP 1 – USE A JUPYTER NOTEBOOK

A Jupyter Notebook environment is used to explore the data, and to test an association algorithm on the book sales dataset.

Explanation	Screenshot
Click to open ML Scenario Manager.	
Click the Create button. Create a new scenario. Name the scenario "Book Recommendation Analysis <USER_ID>". Add your USER_ID to distinguish your scenario from the other ones. You see the empty scenario. First, you will use the Jupyter Notebooks to explore the data and to script the recommendation model in Python. Next, you will design pipelines to bring the code into production. Executions of these pipelines will create Machine Learning models (training phase), which are then deployed as REST-API for	

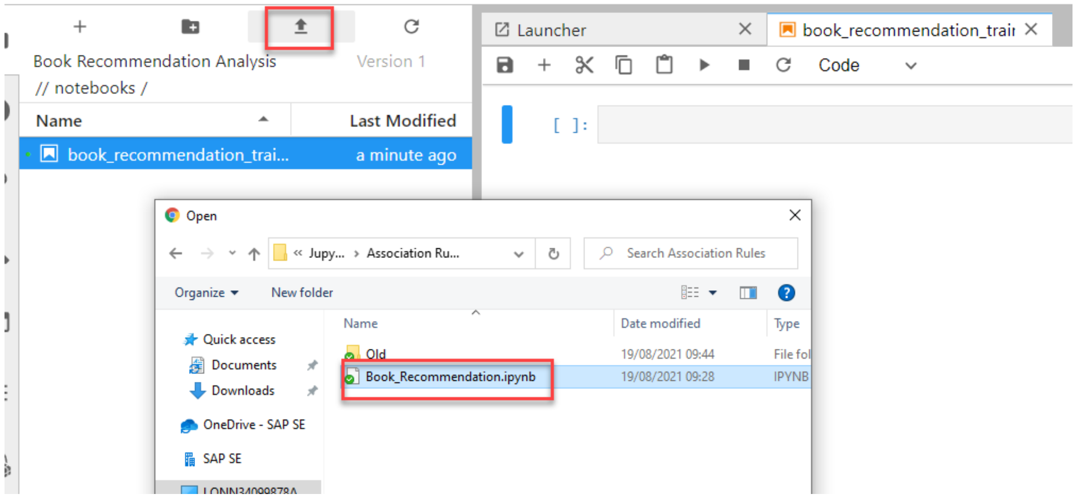
<table>
<tr><th>Explanation</th><th>Screenshot</th></tr>
<tr><td>inference (deployment).</td><td></td></tr>
<tr><td>In the Notebooks section, click Create to create a new notebook.</td><td>
</td></tr>
<tr><td>Name the notebook “book_recommendation_train”.

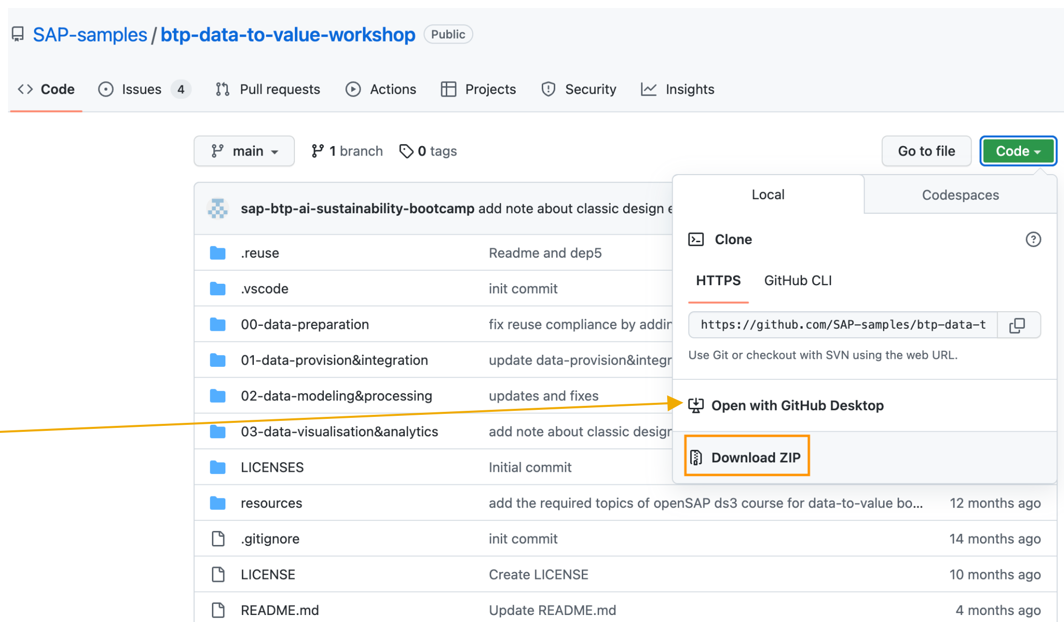
This notebook is used to conduct an Exploratory Data Analysis and run an association rules analysis.</td><td>
</td></tr>
<tr><td>Select the Python 3 Kernel option.</td><td>
</td></tr>
</table>

<table>
<tr><th>Explanation</th><th>Screenshot</th></tr>
<tr><td>
A Jupyter notebook with the Python code splitted into cells was already prepared for you and it is available on the official GitHub repository at this link: DV210_Exercise01_Book_Recommendation.ipynb
So there is no need to populate a new one from scratch.

Use the Upload Files function to upload the Jupyter notebook into the console.

Once it is uploaded, then double click on the notebook that is uploaded.

NB: In order to download the notebooks from GitHub, you need to go to the main page and download locally the entire repository in a zipped format.
</td><td>


NB: The notebook name in the screenshot is only an example. The notebook name is DV210_Exercise01_Book_Recommendation.ipynb.


</td></tr>
</table>

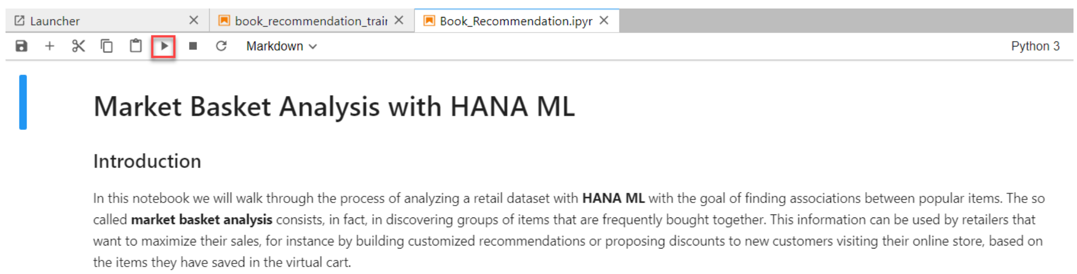
Explanation	Screenshot

The Python Code accesses the data in DWC and explores the data for this exercise.
The data are not loaded into SAP Data Intelligence, but remain in SAP HANA environment, so there is no data transfer.

```
df_hana = (conn.table('V_BL_Sales_Dataset', schema='DEV'))
df_hana.head(20).collect()
```

	order_ID	order_date	quantity	net_amount	Book_Title	Book_Description	Author_ID
0	10706	2017-02-14	1	31	Bedknob and broomstick	None	2016
1	71412	2014-11-08	1	41	Don't let the pigeon drive the bus!	None	2856
2	148404	2015-02-25	1	21	The Indian in the cupboard	Indians of North America--Juvenile fiction ; O...	151
3	92125	2018-11-01	1	10	The BFG	Giants--Juvenile fiction;Children's fiction ; ...	655
4	77517	2019-07-02	1	16	Tom Thorneval : dream merchant extraordinaire	None	21
5	85734	2021-01-09	1	26	Peter Pan	None	176
6	11752	2018-11-19	1	10	The BFG	Giants--Juvenile fiction;Children's fiction ; ...	655
7	117299	2021-06-02	1	27	Alice's adventures in Wonderland : and, Throug...	Children's stories, English ; Alice, (Fictitio...	481

Step through the code to check it works correctly. Highlight each cell in sequence and click the arrow button to run the selected cell and advance.Analyze the results of your data analysis and understand the association rules. Please, read carefully the comments added in the notebook. The meaning of each cell is explained there.



Pay attention at cell 2, you will need to insert the correct name of your DWC connection
When you created the DWC connection in DI you assigned a name in the form *DWC_<USER_ID>*. Please, use that one here.

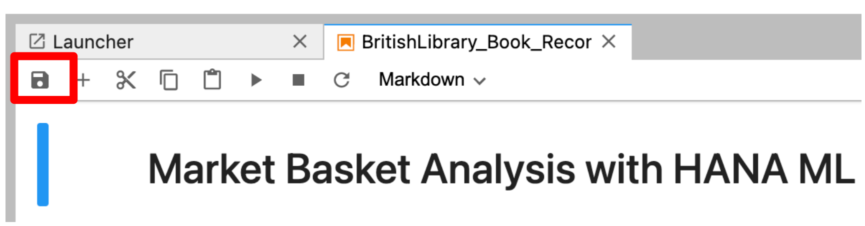
```
[2]: import hana_ml.dataframe as dataframe
     from notebook_hana_connector.notebook_hana_connector import NotebookConnectionContext
     conn = NotebookConnectionContext(connectionId = 'DWC_<XX>' )
```

<table>
<tr><th>Explanation</th><th>Screenshot</th></tr>
<tr><td>Pay attention also at cell 3, you will need to insert the name of the entity you want to use and the schema where it resides in DWC.

We will use the perspective view created during exercise DV200_Exercise05, that should be named All_BL_Sales_Order_Data if you followed the instructions. If it is not, please, use the technical name you assigned.

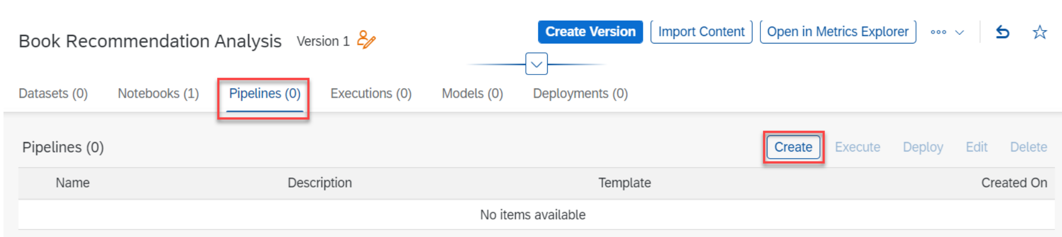
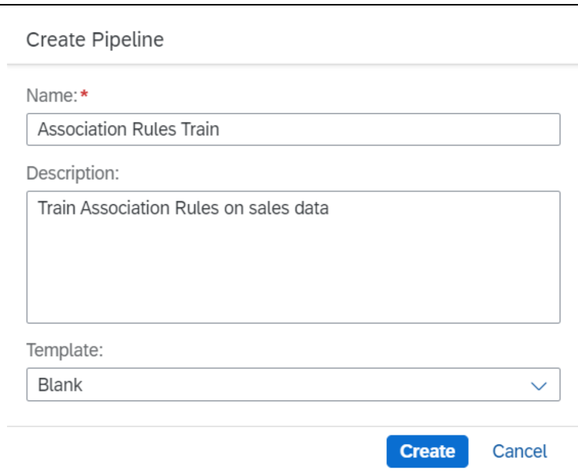
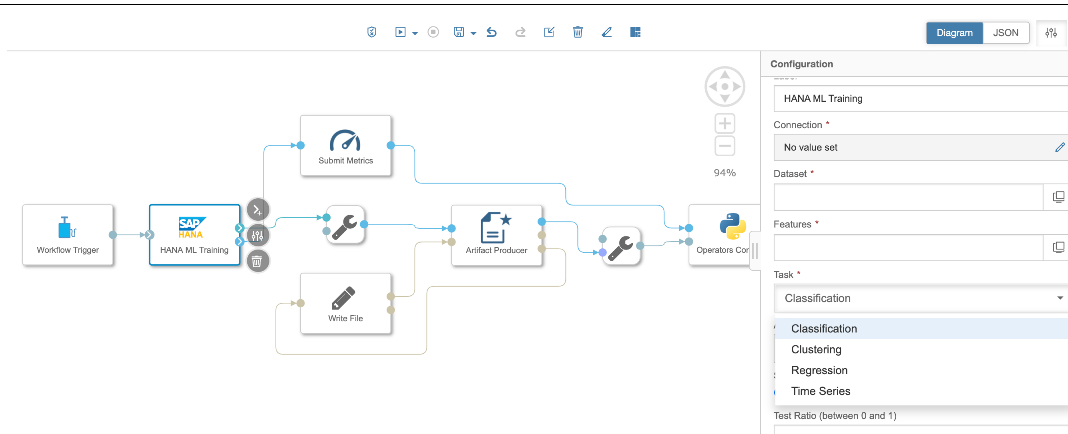
The schema were DWC created the view is named like your USER_ID.
All these aspects can be checked easily by means of the Metadata Explorer/Browse Connections functionality in DI.</td><td>

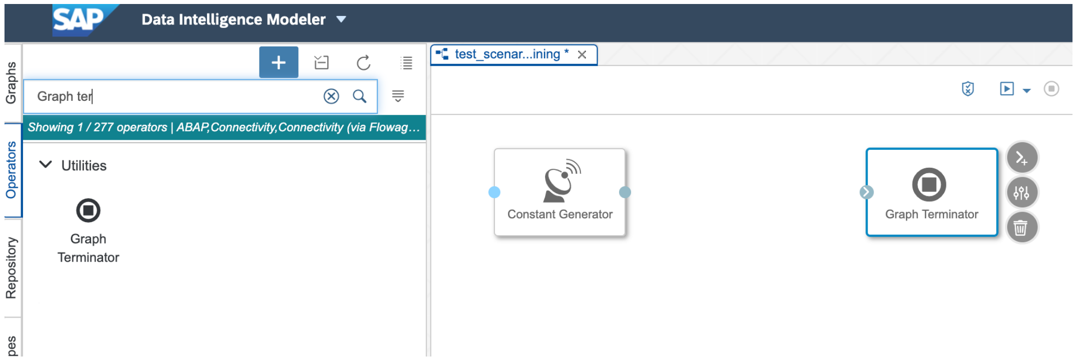
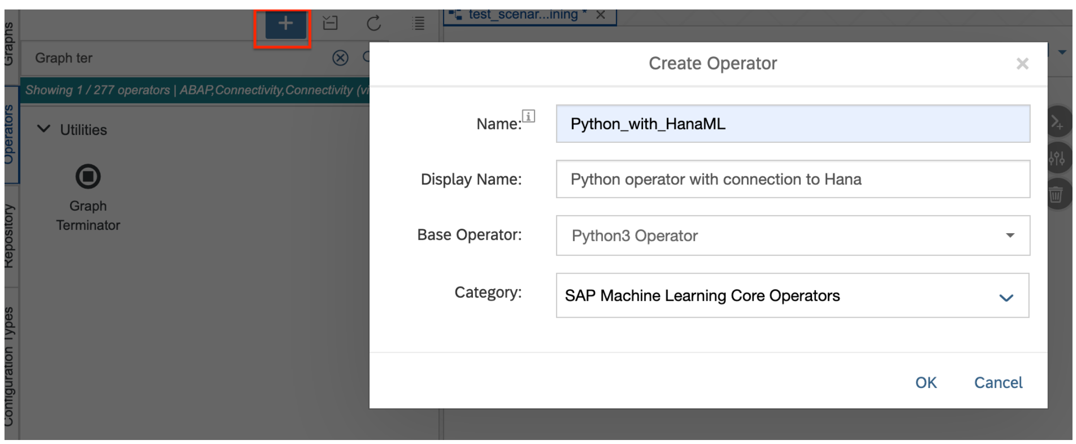
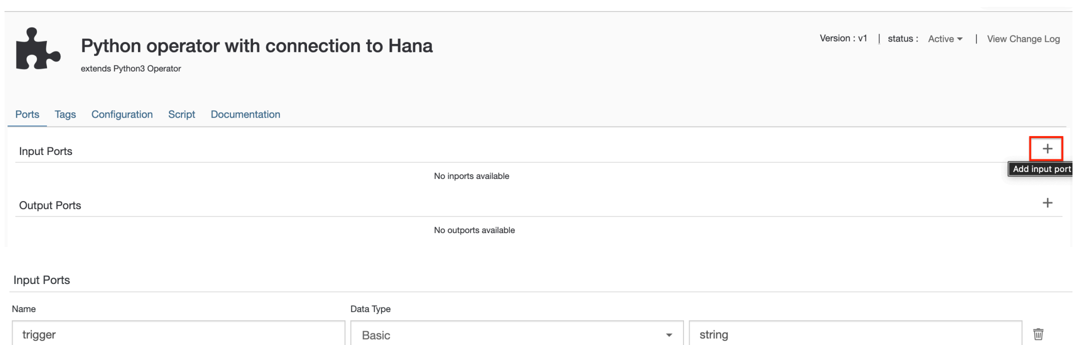
```python
df_hana = (conn.table('All_BL_Sales_Order_Data', schema='D2VUXXXX'))

df_hana.head(20).collect()
```

</td></tr>
<tr><td>Once you have stepped through to the end of the notebook, and there are no errors, save the notebook.
Please, note that the code in the notebook generates an association table during the model training named “APRIORI_BOOK_ASSOCIATION” that is persisted in DWC under the schema <USER_ID>#DI.</td><td>
</td></tr>
</table>

STEP 2 – BUILD MODEL PIPELINES

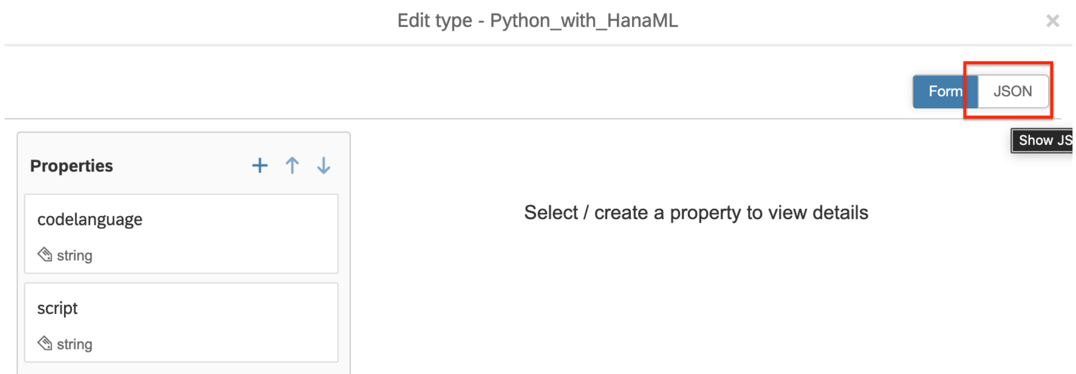
Model pipelines are used to operationalize the association rules model. Now that you have prepared the data, identified the best algorithm to use and which hyper-parameters work best, you can operationalize the model. We will build one pipeline to automate the model training, and a second pipeline to deploy the book recommendation engine.

Explanation	Screenshot
	CREATE A DI PIPELINE TO TRAIN THE ML MODEL
To create the graphical pipeline to retrain the association rules, go to your ML Scenario's main page, select the "Pipelines" tab and click Create.	
Name the pipeline "Association Rules Train" and select the "**Blank**" template to create a bespoke pipeline graph. Click Create.	
Notice that other non-blank templates are available in the menu. For instance, the **HANA ML Training** template (on the right), is very useful for the most common machine learning scenarios. Association rules are not cover by the HANA ML training operator, that's why	

Explanation	Screenshot
we will need a customized pipeline.	
In the blank pipeline, we'll start by adding two operators: 1. Constant Generator 2. Graph Terminator Select each, and double click or drag and drop to add to the console. These will be the start and the end point of our pipeline.	
Then, we will build a custom operator to run association rules in Hana ML. Click on the plus icon at top left of the screen and complete the dialog box as in the picture.	
Create an input port like in the picture: we want the operator to start running as soon as it gets a message from the constant generator operator. Please, pay attention to the name and type you assign to the input port. This name will	

<table>
<tr><th>Explanation</th><th>Screenshot</th></tr>
<tr><td>be used later in the training python code.</td><td></td></tr>
<tr><td>Go to the Configuration tab. We will configure the operator so that it can connect to a HANA instance connected to SAP Data Intelligence. Click on the pencil to get into the edit mode and change to JSON.

In the editor, copy-paste the code available in the GitHub repository at this link reported here.

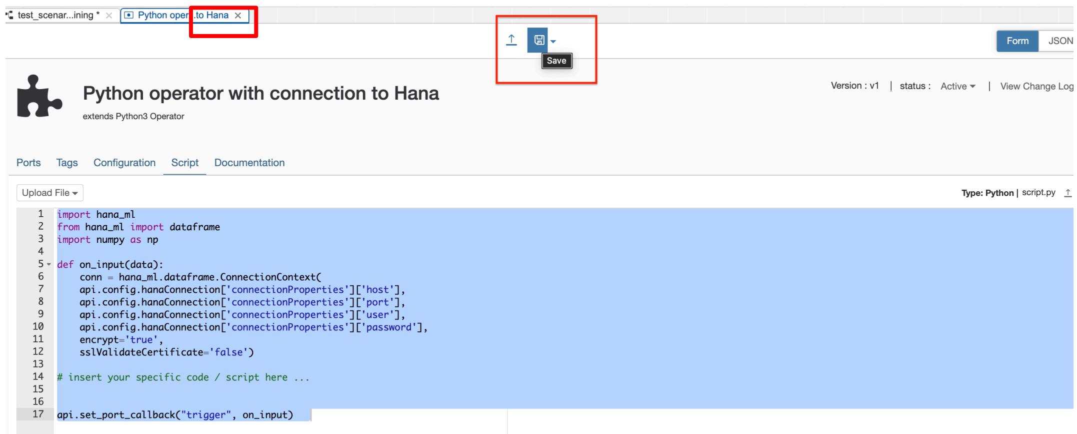
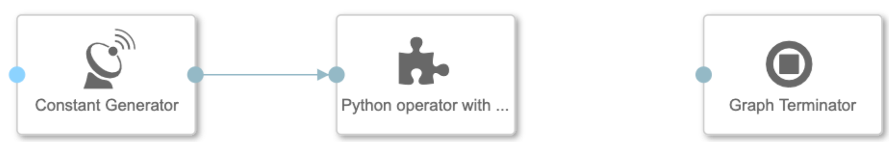
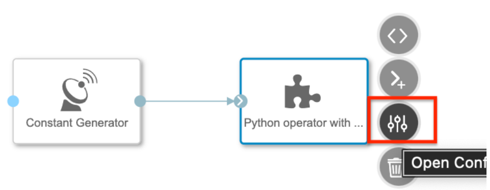
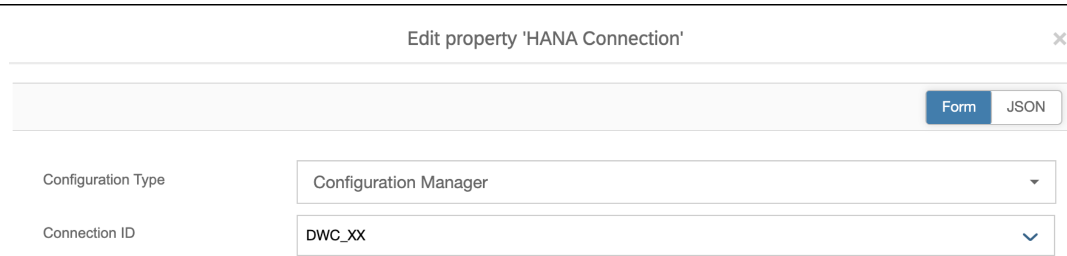
Please, pay attention to this configuration, otherwise later you won’t be able to connect to the HANA instance.</td><td>

https://github.com/SAP-samples/btp-data-to-value-workshop/blob/main/02-data-modeling%26processing/exercises/code_snippets/customPythonOperator.json</td></tr>
<tr><td>Go to the Script tab and copy paste the python code to the right.

NB: The Script tab contains the code the python operator will execute. What we are copying now is just a template code that will be customized later. You can see that only the lines needed for</td><td>

```
import hana_ml
from hana_ml import dataframe
import numpy as np

def on_input(data):
    conn = hana_ml.dataframe.ConnectionContext(
    api.config.hanaConnection['connectionProperties']['host'],
    api.config.hanaConnection['connectionProperties']['port'],
    api.config.hanaConnection['connectionProperties']['user'],
    api.config.hanaConnection['connectionProperties']['password'],
    encrypt='true',
    sslValidateCertificate='false')

# insert your specific code / script here ...
```

</td></tr>
</table>

connecting to HANA Cloud are defined here.	api.set_port_callback("trigger", on_input)
Save the custom operator and the close the window.	
You should be now again in the pipeline window. Look for your new custom operator in the menu and add it to the pipeline. Connect the constant Generator with the trigger port of the custom operator.	
Begin with configuring the HANA ML within the Python operator. Highlight it and click on the Configuration icon.	
Edit the **Hana Connection** field. Select Configuration Manager as Configuration Type and select your DWC connection in the Connection ID drop-down menu. Click save.	

Please, recall the DWC connection was set in exercise DV150_Exercise04 and the name you should have set is in the form DWC_<USER_ID>.

Coming back to the Python operator, highlight it and click on the script icon to modify the code the instance will execute.

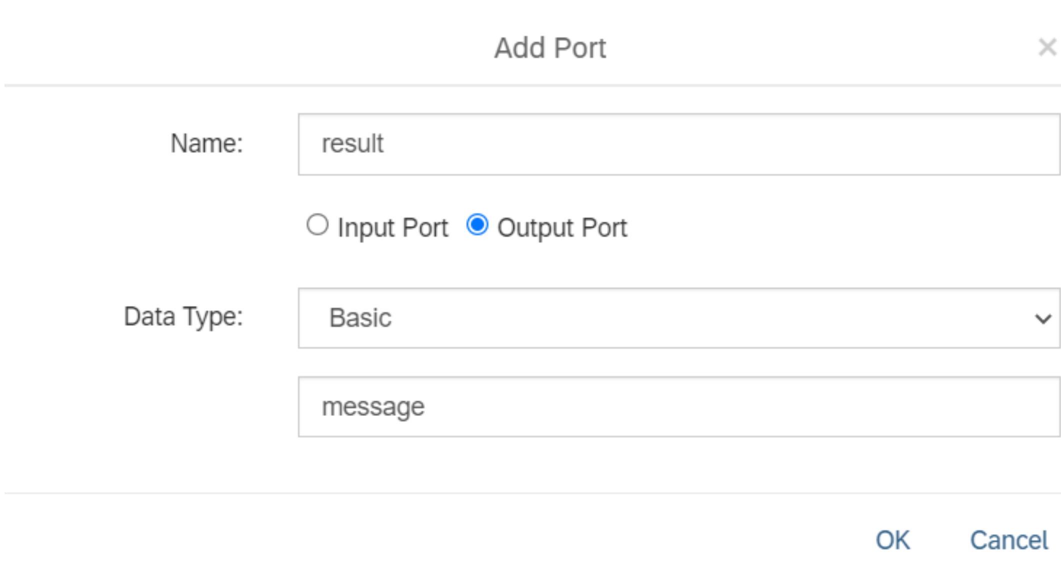
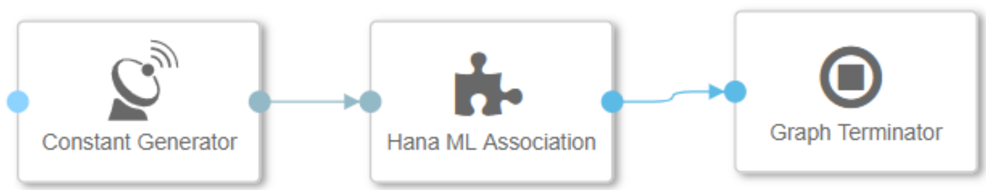
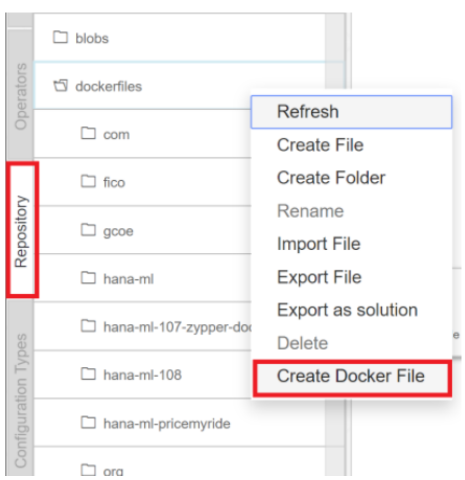
Customize the python template by adding the code to train the association algorithm. The complete script is available at the link reported here. Copy and paste the code in the script editor and save the operator again.

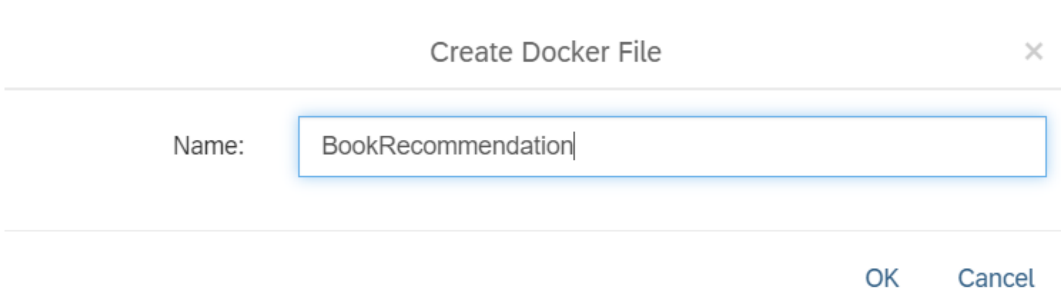
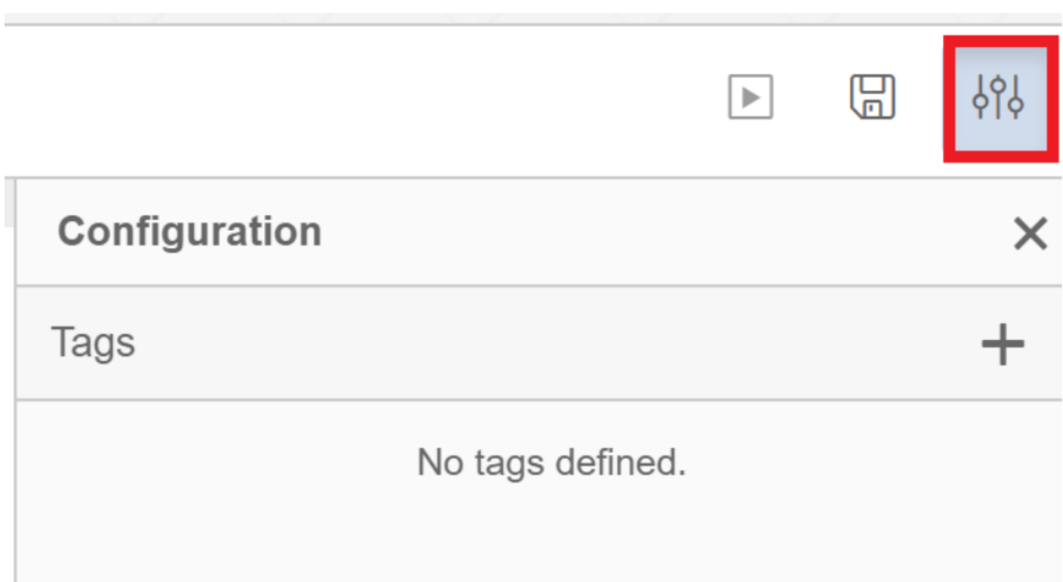
https://github.com/SAP-samples/btp-data-to-value-workshop/blob/main/02-data-modeling%26processing/exercises/code_snippets/dv210_train.py

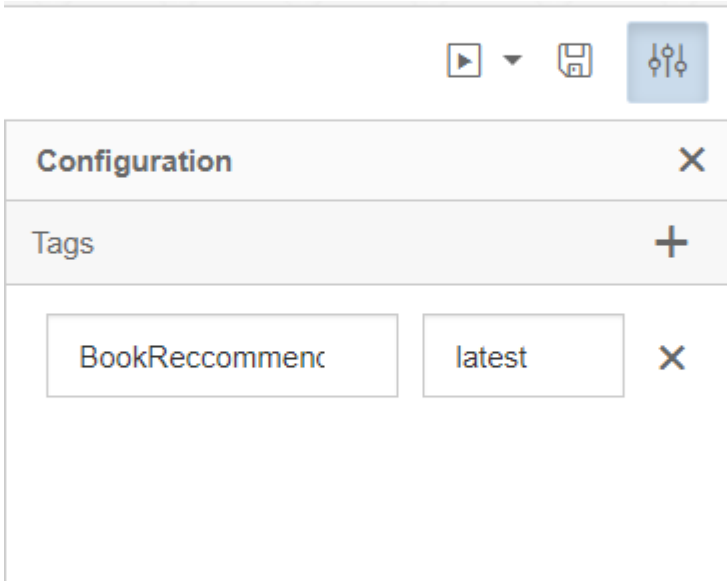
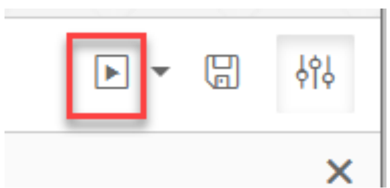
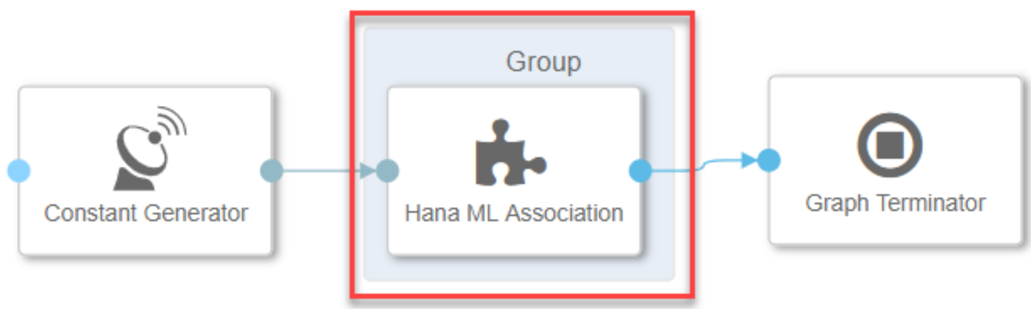
Please, notice that python is sensitive to indentation!

NB: In the script, you need to access the table you created in DV200_Exercise05 (that is the Perspective view). You need also to specify the schema where the view resides and that corresponds to your USER_ID.

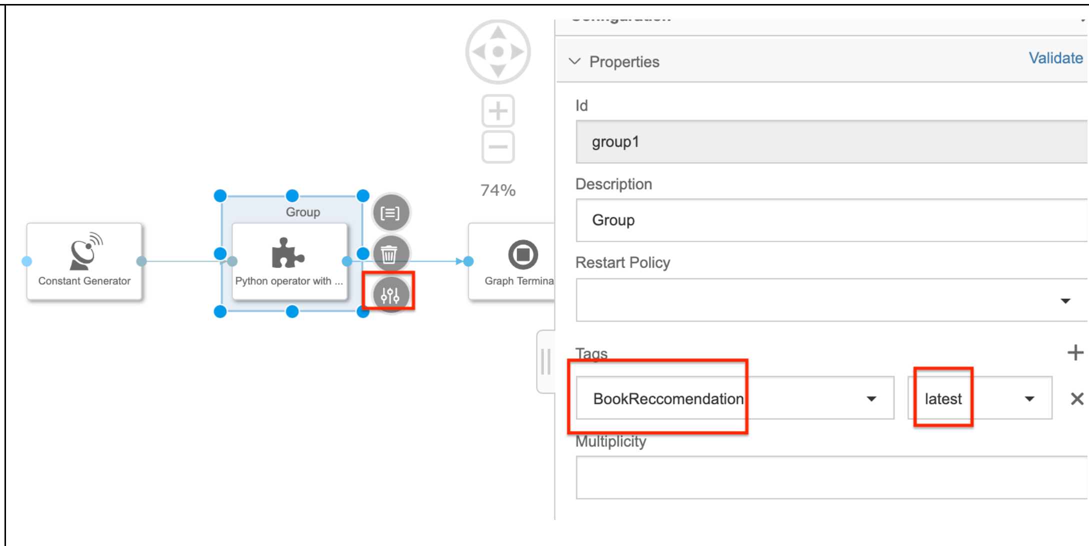
```
# insert your specific code / script here ...
    df_hana = (conn.table('PP_All_BL_Sales_Order_Data', schema='PA09700UXXX'))
    df_hana=df_hana.select('Order_ID','Book_ID')

    from hana_ml.algorithms.pal.association import Apriori
```

<table>
<tr>
<td>Return to your pipeline.
Right-click on the custom operator and select Add Port.
Add an output port.
Enter the information here and then click OK.</td>
<td>
</td>
</tr>
<tr>
<td>Link the output of the Constant Generator to the Trigger port of the Python operator, and the result of the Python operator to the Graph Terminator to create the pipeline.</td>
<td>
</td>
</tr>
<tr>
<td>You need to create a Docker image for the Python operator. This gives the flexibility to leverage virtually any Python library needed to execute your code. The Docker file specifies and installs the necessary libraries for your code.
You can find the docker files by clicking on the “Repository” tab on the left.</td>
<td>
</td>
</tr>
</table>

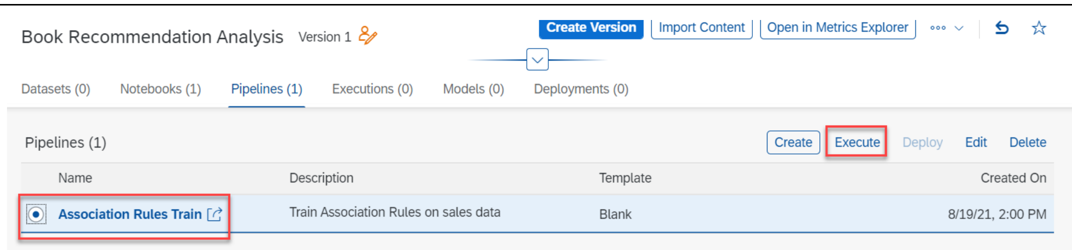
To create a new one right-click on the "**dockerfiles**" folder and select "Create Docker File".	
Name the file BookRecommendation.	
Enter this code into the Docker File window. This code leverages a base image that comes with SAP Data Intelligence and installs the necessary libraries on it.	FROM $com.sap.sles.base RUN pip install --user numpy RUN pip install --user pandas RUN pip install --user hana_ml
Open the Configuration panel for the Docker File with the icon on the top-right hand corner.	

Assign a tag to the Dockerfile	
Now save the Docker file and then click the "Build" icon to start building the Docker image. Wait a few minutes and you should receive a confirmation that the build completed successfully.	
Now you need to configure the custom operator, which trains the model, to use this Docker image. Click the tab to go back to your graphical pipeline.Right-click the custom operator and select "Group".	

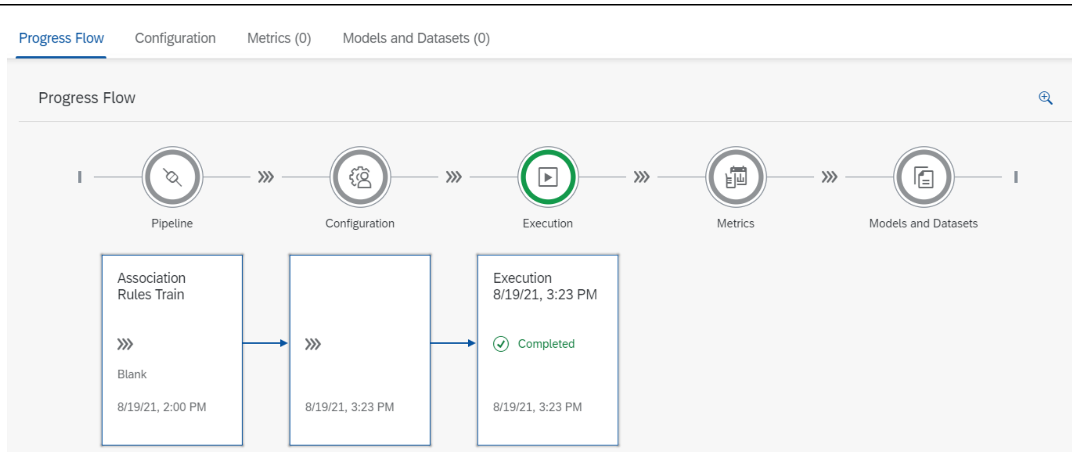
You specify which Docker image should be used. Select the group which surrounds the "Hana ML". In the group's Configuration select the docker tag. Save your pipeline.



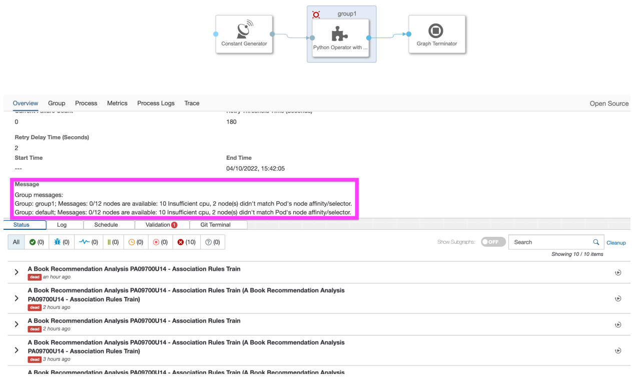
The pipeline is now complete and you can run it. Go back to the ML Scenario. Select the pipeline in the ML Scenario and click the "Execute" button on the right.



Wait a few seconds until the pipeline executes and completes.
In case of problems, you can check the logs from the pipeline modeler. By ckicking on the issue, you can access its description to get more details and debug your code.

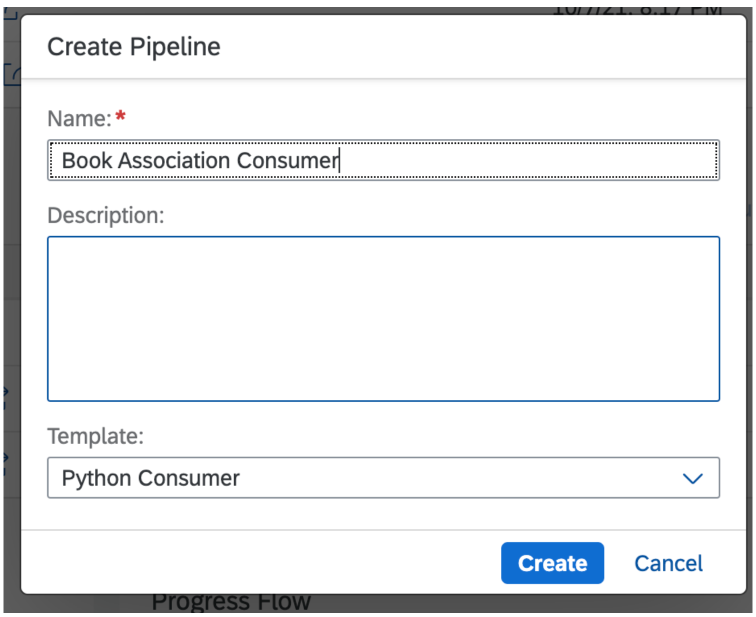


If the training completes successfully, then the code will write in DWC a new association table with the result of the training. The name of this table is specified in the python code (APRIORI_BOOK_ASSOCIATION_IDS to distinguish it from the one created when running the code within the Jupyter notebook). Please, note that the table resides under the schema <USER_ID>#DI.



CREATE A DI PIPELINE TO SERVE AND CONSUME THE ML MODEL

You will now use the model for real-time inference with REST-API.Go back to the main page of your ML Scenario and create a second pipeline. This pipeline will provide the REST-API to obtain predictions in real-time. Select the template “Python Consumer” to create a bespoke pipeline. Click Create.

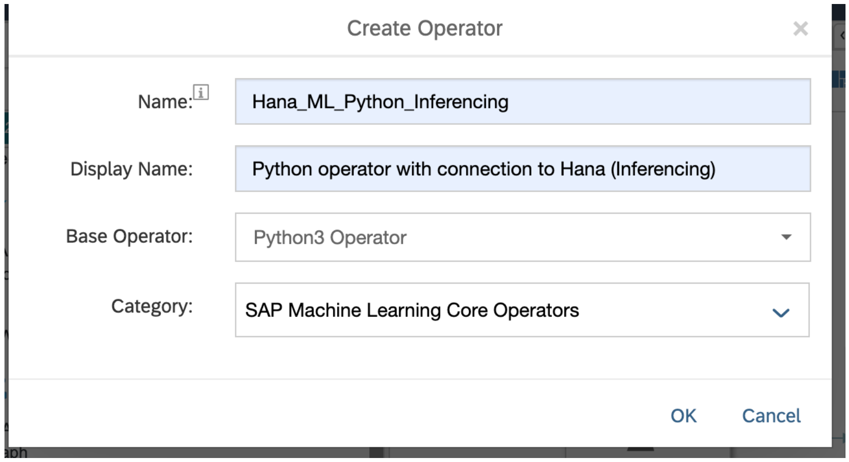


Take a moment to understand this template. The first portion of the graph, shown in red in the picture, has the function of reading a model artifact stored in a binary file and feeding it to the second portion of the graph. As a matter of fact, in machine learning scenarios, trained models such

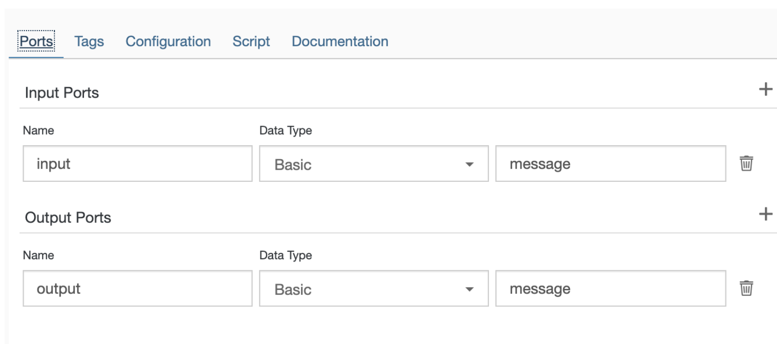
as regressors or classifiers, are usually stored in binary format. In our scenario, however, the results of Apriori are not stored in an artifact, but in a HANA table. e can delete the operators within the red box, we will not need them. The blue portion of the graph, instead, is what we need to expose the book association table to an open API service. The OpenAPI Servlow operator reads requests coming from the openAPI and submits them to the python operator. These requests contains the ID of books that are about to be chosen by a customer. The role of the python operator will be querying the book association table to come up with a list of recommendations that will be sent back to the open API service.	

Since we need to query a table stored in HANA, a simple python operator will not suffice. We will need a custom operator, similar to the one we built for the training pipeline, with the possibility to configure a HANA connection. Click on the plus button

to create a new operator. Configure the dialog box as shown here.

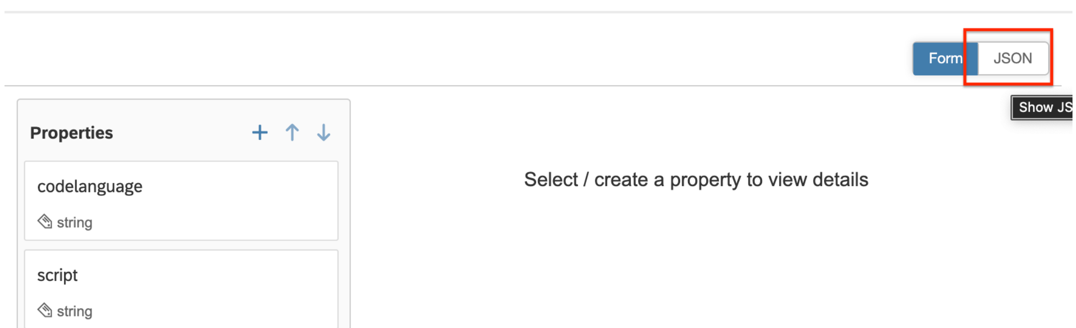


Add two ports as shown in the picture Please, pay attention to the name and type of the input ports. In particular, the names will be used in the python code of this new operator.



Go to the Configuration tab and repeat the same steps you did for the training custom operator.

NB: Also in this case you need to configure the operator to be able to connect to HANA Cloud. This is done by means of the same JSON file you have previously used for the training python operator.



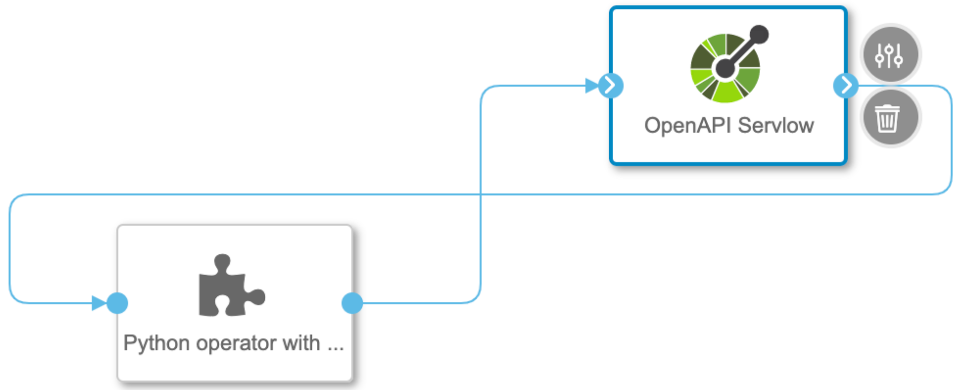
https://github.com/SAP-samples/btp-data-to-value-workshop/blob/main/02-data-modeling%26processing/exercises/code_snippets/customPythonOperator.json

In the Script session, enter the piece of code here
Save the operator and go back to your blank pipeline

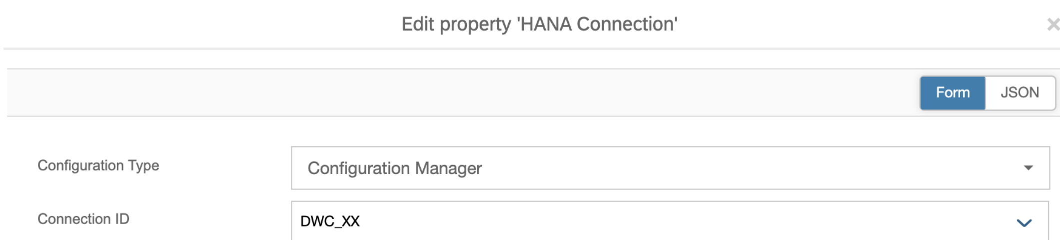
https://github.com/SAP-samples/btp-data-to-value-workshop/blob/main/02-data-modeling%26processing/exercises/code_snippets/dv210_consumer.py

---

Replace the Python Inference operator with the custom operator you just built.

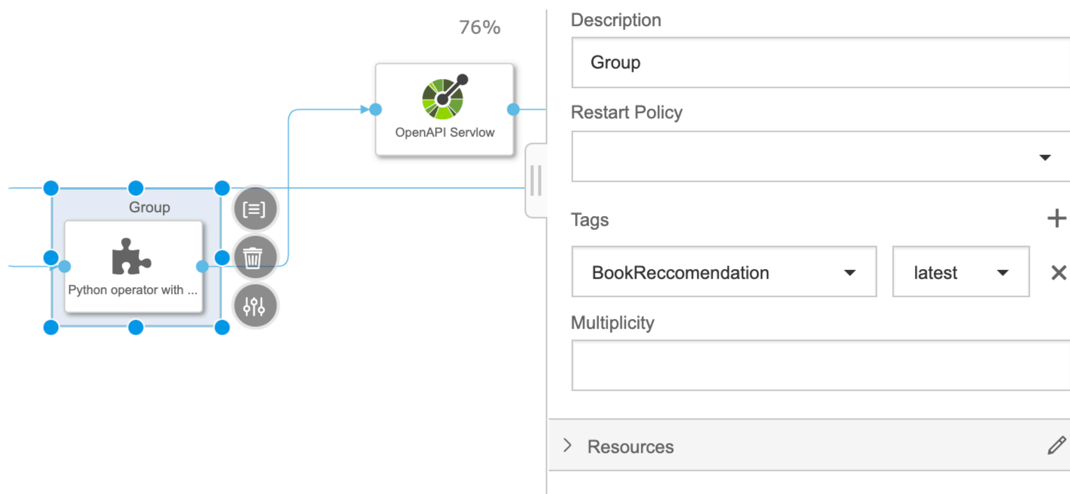
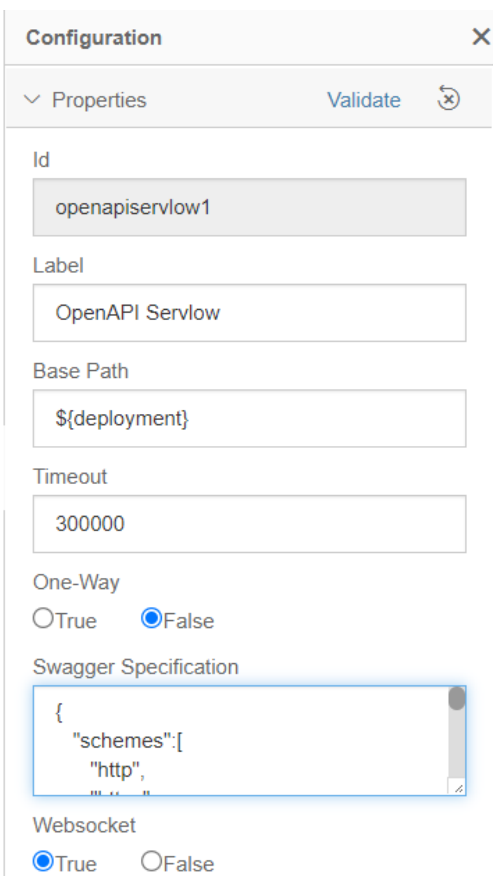


Open the configuration menu for the custom operator and select the DWC_<USER_ID> connection as already done for the training pipeline
NB: Please, keep in mind the naming convention for the DWC connection we used during exercise DV150_Exercise04.



In the nferencing Python operator, click the "Script" icon. Make sure you are using the table produced during the training phase with the correct schema (that is in the form <USER_ID>#DI). You can cross check by using the Metadata Explorer

```python
try:
    # Load your HANA_ML model here
    model = (conn.table('APRIORI_BOOK_ASSOCIATION_IDS', schema='PA09700UXXX#DI'))
    model_ready = True
    api.logger.info("Model Received & Ready")
except Exception as e:
        api.logger.error(e)
        error_message = "An error occurred while loading the model: " + str(e)
```

-> Browse Connections functionality. Close the editor window.	
The Docker image built for the training is also good for inferencing. Assign the Docker image. As before, right-click the custom operator and select "Group". Add the tag ave the changes.	
Click on OpenAPIServlow and have a look at the configuration	

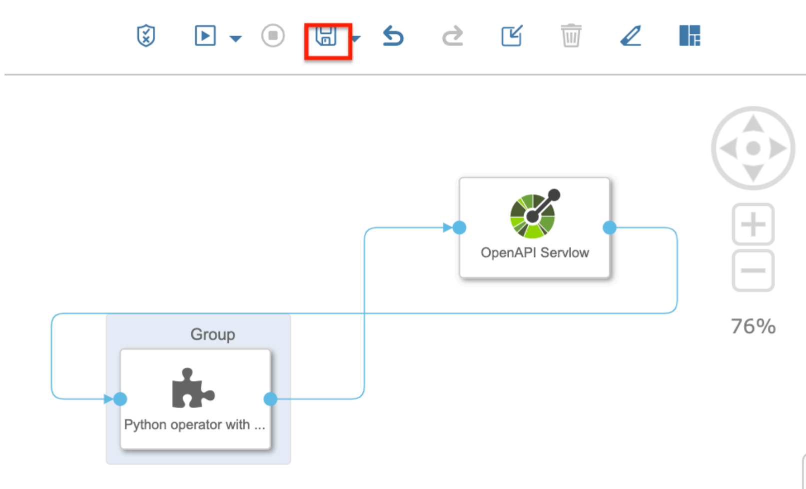
<table>
<tr>
<td></td>
<td>
Max Concurrency
32
Max Accepted
128
</td>
</tr>
<tr>
<td>
Notice in particular the content of the Swagger Specification,but don’t change anything here!

The code highlighted in yellow will define the URL will be exposed.
</td>
<td>
<pre>
{
  "schemes":[
    "http",
    "https"
  ],
  "swagger":"2.0",
  "info":{
    "description":"This is an example of using the OpenAPI Servlow to carry out
inference with an existing model.",
    "title":"OpenAPI demo",
    "termsOfService":"http://www.sap.com/vora/terms/",
    "contact":{

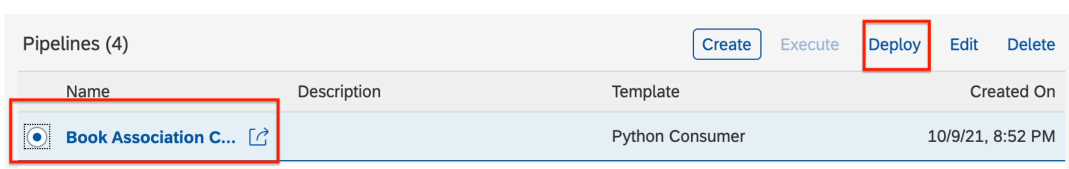
    },
    "license":{
      "name":"Apache 2.0",
      "url":"http://www.apache.org/licenses/LICENSE-2.0.html"
    },
    "version":"1.0.0"
  },
  "basePath":"/$deployment",
  "paths":{
    "/v1/uploadjson":{
      "post":{
        "description":"Upload data in json format",
        "consumes":[
          "application/json"
        ],
        "produces":[
          "application/json"
        ],
        "summary":"Upload JSON data to be used in the Python operator's script",
        "operationId":"upload",
        "parameters":[
          {
            "type":"object",
            "description":"json data",
            "name":"body",
            "in":"body",
            "required":true
          }
        ],
        "responses":{
</pre>
</td>
</tr>
</table>

```
                "200":{
                  "description":"Data uploaded"
                },
                "500":{
                  "description":"Error during upload of json"
                }
              }
            }
          }
        },
        "definitions":{
        },
        "securityDefinitions":{
          "UserSecurity":{
            "type":"basic"
          }
        }
      }
```

Save the changes applied to the consumer pipeline.



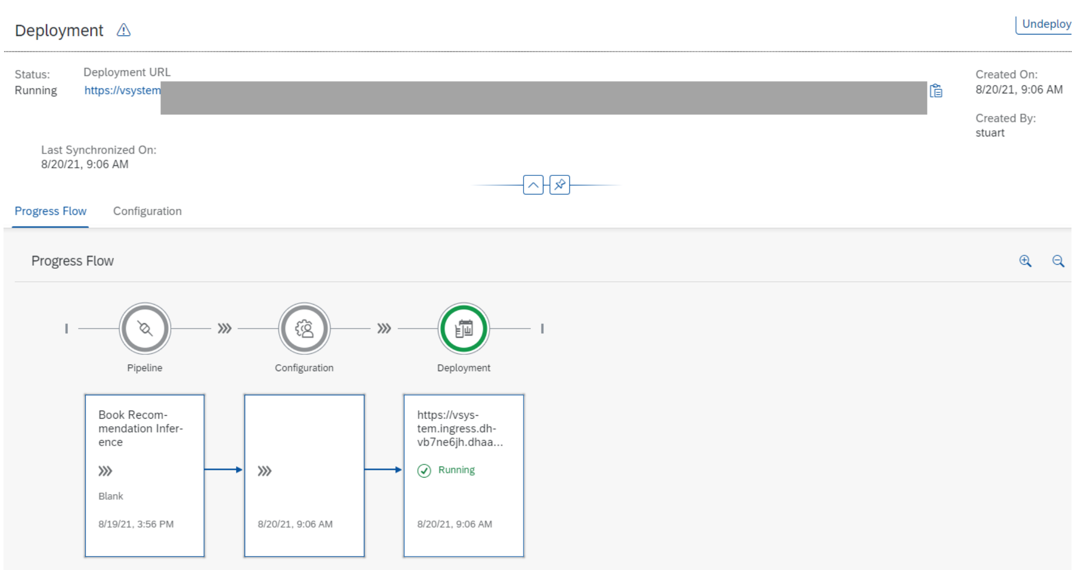
Go back to the ML Scenario. Select your pipeline and click “Deploy”.



After a few seconds the pipeline will start running.

Also in this case, any issues can be investigated within the pipeline modeler.

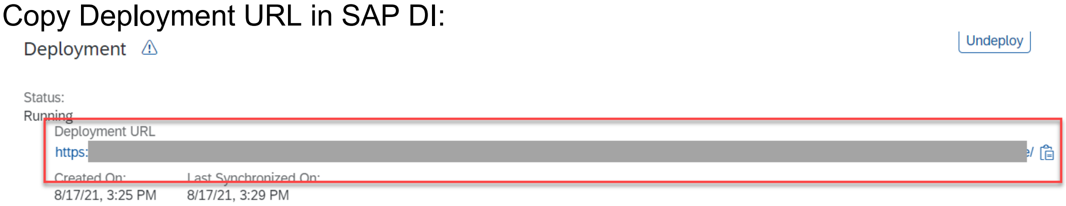
NB: Once the consumer pipeline is started, it will run until you stop it. If you want to make an inference with the ML model through the exposed URL, It is necessary the consumer is running. **If you don’t need it anymore, please stop it to free resources!**



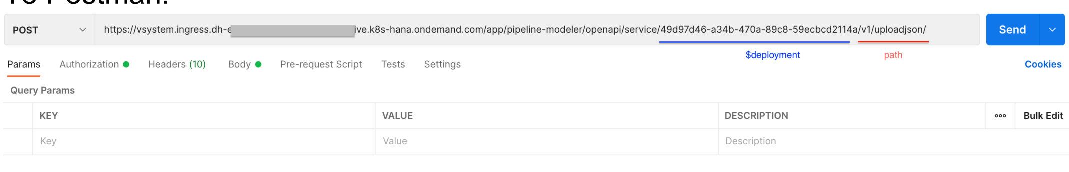
STEP 3 – USE YOUR ASSOCIATION RULES MODEL

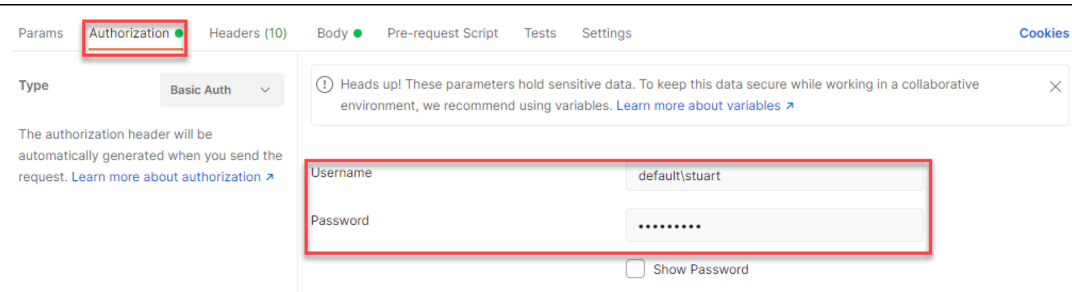
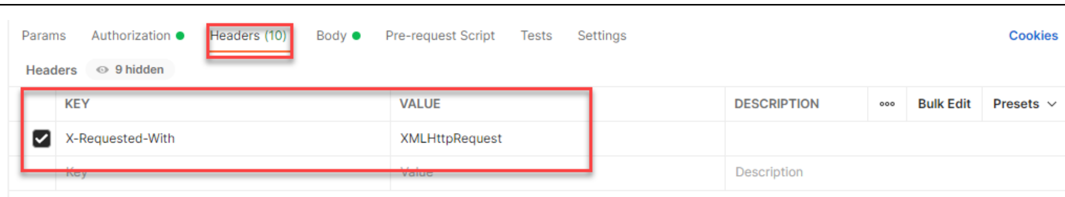
Now that you have deployed your model, you can use it for real-time book recommendations. For this, you are going to use the Postman application. Please, use the desktop version to avoid the limitations of the cloud version.

<table>
<tr><th>Explanation</th><th>Screenshot</th></tr>
<tr><td>Open Postman. Copy the deployment URL from SAP DI. Enter the Deployment URL as request URL. Extend the URL with v1/uploadjson/, the path specified in the OpenAPI servlow operator. Change the request type from “GET” to “POST”.</td><td>Copy Deployment URL in SAP DI:

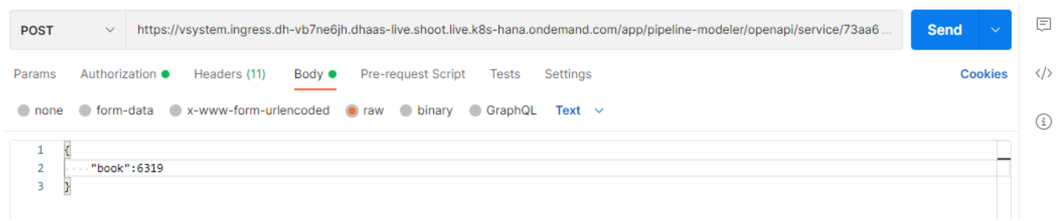


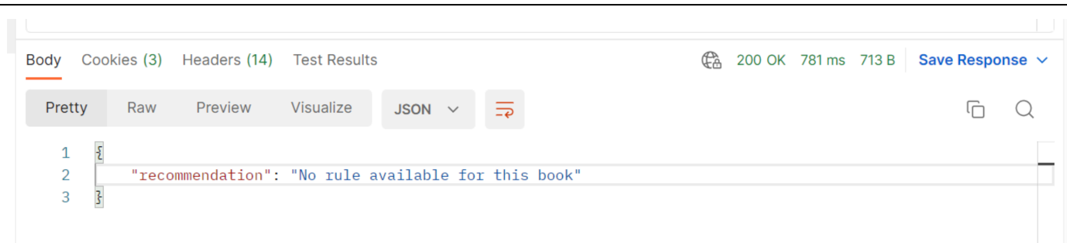
To Postman:


</td></tr>
<tr><td>Go to the “Authorization” tab. Select “Basic Auth” and enter your username and password for SAP Data Intelligence. The username starts with your tenant’s name, followed by a backslash and your actual username.

Please, go to the Teams channel General>System Access to check the name of your tenant.</td><td>
</td></tr>
<tr><td>Go to the “Headers” tab and enter the key “X-Requested-With” with value “XMLHttpRequest”.</td><td>
</td></tr>
</table>

Explanation	Screenshot

Finally, pass the input data to the REST-API.
Select the “Body” tab, choose “raw” and enter the syntax given here. Replace <Book_ID> with a book ID such as 6319.
 	
NB! The Book_ID must be contained in the list of antecedents, so you can find some valid examples to test from the output you created when you ran the Python code analysis in Jupyter.
	
	
	{
	   "book": <Book_ID>
	}

Press “Send” and you will see the book recommendation that comes from SAP Data Intelligence. Try the REST-API with different book IDs to see how the recommendations change.	
If you get a response like in this screenshot, this is not an error. The algorithm is just saying there are no recommendations for that book.	

Explanation	Screenshot
You have now completed the exercise.	

APPENDIX 1 – INTRODUCTION TO ASSOCIATION RULES

For a clearly presented tutorial on the concepts of association rules and the Apriori algorithm we use in this exercise, please see **https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html.**

APPENDIX 2 – APRIORI IN SAP HANA ML

Apriori is a classic predictive analysis algorithm for finding association rules used in association analysis. Association analysis uncovers the hidden patterns, correlations or casual structures among a set of items or objects. For example, association analysis enables you to understand what products and services customers tend to purchase at the same time. By analyzing the purchasing trends of your customers with association analysis, you can predict their future behavior.

Apriori is designed to operate on databases containing transactions. As is common in association rule mining, given a set of items, the algorithm attempts to find subsets which are common to at least a minimum number of the item sets. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time, a step known as candidate generation, and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found. Apriori uses breadth-first search and a tree structure to count candidate item sets efficiently. It generates candidate item sets of length k from item sets of length k-1, and then prunes the candidates which have an infrequent sub pattern. The candidate set contains all frequent k-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates.

The Apriori function in PAL uses vertical data format to store the transaction data in memory. The function can take VARCHAR/NVARCHAR or INTEGER transaction ID and item ID as input. It supports the output of confidence, support, and lift value, but does not limit the number of output rules.

Prerequisites:
- The input data does not contain null value.
- There are no duplicated items in each transaction.

Input Table

Table	Column	Data Type	Description
DATA	1st column	INTEGER, VARCHAR, or NVARCHAR	Transaction ID
	2nd column	INTEGER, VARCHAR, or NVARCHAR	Item ID

Parameter Table
Mandatory Parameters
The following parameters are mandatory and must be given a value.

Name	Data Type	Description
MIN_SUPPORT	DOUBLE	User-specified minimum support (actual value).
MIN_CONFIDENCE	DOUBLE	User-specified minimum confidence (actual value).

Optional Parameters
The following parameters are optional. If a parameter is not specified, PAL will use its default value.

Name	Data Type	Default Value	Description
MIN_LIFT	DOUBLE	0.0	User-specified minimum lift.
MAX_CONSEQUENT	INTEGER	100	Maximum length of dependent items.
MAXITEMLENGTH	INTEGER	5	Total length of leading items and dependent items in the output.
UBIQUITOUS	DOUBLE	1.0	Ignores items whose support values are greater than the UBIQUITOUS value during the frequent items mining phase.
IS_USE_PREFIX_TREE	INTEGER	0	Indicates whether to use the prefix tree, which can save memory. • 0: Does not use the prefix tree. • 1: Uses the prefix tree.
LHS_RESTRICT	VARCHAR	No default value	Specifies that some items are only allowed on the left-hand side of the association rules.
RHS_RESTRICT	VARCHAR	No default value	Specifies that some items are only allowed on the right-hand side of the association rules.

Name	Data Type	Default Value	Description

LHS_IS_COMPLEMENTARY_RHS	INTEGER	0	If you use RHS_RESTRICT to restrict some items to the right-hand side of the association rules, you can set this parameter to 1 to restrict the complementary items to the left-hand side.
			
			For example, if you have 1000 items (i1, i2, ..., i1000) and want to restrict i1 and i2 to the right-hand side, and i3, i4, ..., i1000 to the left-hand side, you can set the parameters similar to the following:
			
			`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i1');`
			
			`INSERT INTO PAL_CONTROL_TBL VALUES ('RHS_RESTRICT', NULL, NULL, 'i2');`
			
			`INSERT INTO PAL_CONTROL_TBL VALUES ('LHS_IS_COMPLEMENTARY_RHS', 1, NULL, NULL);`

RHS_IS_COMPLEMENTARY_LHS	INTEGER	0	If you use LHS_RESTRICT to restrict some items to the left-hand side of the association rules, you can set this parameter to 1 to restrict the complementary items to the right-hand side.

Name	Data Type	Default Value	Description
THREAD_RATIO	DOUBLE	0	Specifies the ratio of total number of threads that can be used by this function. The value range is from 0 to 1, where 0 means only using 1 thread, and 1 means using at most all the currently available threads. Values outside the range will be ignored and this function heuristically determines the number of threads to use.
TIMEOUT	INTEGER	3600	Specifies the maximum run time in seconds. The algorithm will stop running when the specified timeout is reached.
PMML_EXPORT	INTEGER	0	• 0: Does not export Apriori model in PMML. • 1: Exports Apriori model in PMML in single row. • 2: Exports Apriori model in PMML in several rows, and the minimum length of each row is 5000 characters.

Output Tables

Table	Column	Data Type	Column Name	Description
RESULT	1st column	NVARCHAR(1000)	ANTECEDENT	Leading items
	2nd column	NVARCHAR(1000)	CONSEQUENT	Dependent items
	3rd column	DOUBLE	SUPPORT	Support value
	4th column	DOUBLE	CONFIDENCE	Confidence value
	5th column	DOUBLE	LIFT	Lift value
MODEL	1st column	INTEGER	ROW_INDEX	ID
	2nd column	NVARCHAR(5000)	MODEL_CONTENT	Apriori model in PMML format

We know that the connection is set, we can access the sales order table in the form of a hana dataframe and start to prepare the data for the association analysis.

```
df_hana = (conn.table('SAP_CAPIRE_BOOKSHOP_ORDERITEMS', schema='AC3287U01'))
df_hana.head(20).collect()
```

This gives an output:

	ORDER_ID	ORDER_DATE	BOOK_ID	TITLE
0	115	2012-09-07	5	Harry Potter and the Prisoner of Azkaban (Harr...
1	237	2015-05-02	5	Harry Potter and the Prisoner of Azkaban (Harr...
2	285	2011-06-11	5	Harry Potter and the Prisoner of Azkaban (Harr...
3	312	2015-09-17	5	Harry Potter and the Prisoner of Azkaban (Harr...
4	321	2015-01-21	5	Harry Potter and the Prisoner of Azkaban (Harr...
5	394	2014-12-29	5	Harry Potter and the Prisoner of Azkaban (Harr...
6	553	2015-08-06	5	Harry Potter and the Prisoner of Azkaban (Harr...
7	674	2016-09-01	5	Harry Potter and the Prisoner of Azkaban (Harr...
8	711	2012-11-03	5	Harry Potter and the Prisoner of Azkaban (Harr...
9	749	2016-05-18	5	Harry Potter and the Prisoner of Azkaban (Harr...
10	916	2015-08-19	5	Harry Potter and the Prisoner of Azkaban (Harr...
11	968	2014-04-28	5	Harry Potter and the Prisoner of Azkaban (Harr...
12	1438	2015-12-29	5	Harry Potter and the Prisoner of Azkaban (Harr...
13	152[illegible]	2015-1[illegible]-07	5	Harry Potter and the Prisoner of Azkaban (Harr...

To run the book association analysis, we will need just the order id and the book id, so we have filtered these two columns using the select method.

```
df_hana=df_hana.select('ORDER_ID','BOOK_ID')
df_hana.head(5).collect()
```

This gives an output:

	ORDER_ID	BOOK_ID
0	115	5
1	237	5
2	285	5
3	312	5
4	321	5

```
import numpy as np
print('Number of purchased books: ', df_hana.shape[0])
n_transactions=len(np.unique(df_hana.collect()['ORDER_ID']))
print( 'Number of purchase orders: ',n_transactions)
```

This gives an output:

```
Number of purchased books:  146293
Number of purchase orders:  84607
```

As we can see above, the book sales records contain the list of books sold by Mr. Cricket in the last few years. “Harry Potter and the Prisoner of Azkaban” had a pretty good success. The table is in long (transactional) format, meaning that each row contains only one book, and books that were sold in the same transaction are recorded in multiple rows having the same order ID.

The sales history contains 84607 transactions, for a total of 146293 books.

Build the Association Rules Model
For this analysis, we will use the Apriori association algorithm. We will not enter here in the details of how the algorithm works, but you can check out the following resources to learn more:
- [HANA ML Python APIs.- Association Analysis Algorithms] (https://blogs.sap.com/2019/09/03/association-algorithms-hana-ml-apis/)
- [Association Rules and the Apriori Algorithm: A Tutorial] (https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html)
- [Association Discovery — the Apriori Algorithm] (https://pub.towardsai.net/association-discovery-the-apriori-algorithm-28c1e71e0f04)
- [SAP HANA PAL documentation] (https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.04/en-US/7a073d66173a4c1589ef5fbe5bb3120f.html)

In the cell below, we import the Apriori algorithm from HANA ML and we fit it to our sales dataset. The algorithm will crunch historical sales records in search of good book associations rules. Notice that we set a few parameters while calling the algorithm (e.g. min_support and min_confidence). We use this analysis in Jupyter Notebooks to find suitable values for these thresholds, often repeating the tests a number of times to identify the best range for the parameters. Once we have established the best parameter values, we can then easily use these to productionize the models in the SAP DI pipeline. We will come back to these and explain them later.

```
from hana_ml.algorithms.pal.association import Apriori

min_support=0.0005
min_confidence=0.05

ap = Apriori(min_support=min_support,
        min_confidence=min_confidence,
        max_len = 2,
        )

ap.fit(data=df_hana)
```

To look at the output:

```
rules_df = ap.result_.collect().sort_values('LIFT',ascending=False)
rules_df
```

The output is:

	ANTECEDENT	CONSEQUENT	SUPPORT	CONFIDENCE	LIFT
73	6545536	6393047	0.000508	0.573333	606.350167
72	6393047	6545536	0.000508	0.537500	606.350167
75	280277	277191	0.000898	0.655172	518.057686
74	277191	280277	0.000898	0.710280	518.057686
70	6389704	6393047	0.000508	0.462366	488.992070
...	...	...	...	...	...
3	11737313	11594337	0.000579	0.114486	6.540389
11	233818	47281	0.000520	0.063128	5.987718
2	11594337	11387515	0.002127	0.121540	2.750225
4	11737313	11387515	0.000508	0.100467	2.273398
7	30119	5	0.000556	0.054335	1.082955

76 rows × 5 columns

The Apriori algorithm found 76 book associations. Let's replace the book ID with the book title, so that we can have a better understanding of the results.

```
import string

books = (conn.table('SAP_CAPIRE_BOOKSHOP_BOOKS', schema='AC3287U01')).collect()
books=books.set_index('ID')

rules_df['ANTECEDENT']=rules_df['ANTECEDENT'].apply(lambda x:  books.TITLE[int(x)])
rules_df['CONSEQUENT'] =rules_df['CONSEQUENT'].apply(lambda x: books.TITLE[int(x)])
rules_df
```

The output is:

	ANTECEDENT	CONSEQUENT	SUPPORT	CONFIDENCE	LIFT
73	The Emperor's Code (The 39 Clues, #8)	The Viper's Nest (39 Clues, #7)	0.000508	0.573333	606.350167
72	The Viper's Nest (39 Clues, #7)	The Emperor's Code (The 39 Clues, #8)	0.000508	0.537500	606.350167
75	Lucinda's Secret (The Spiderwick Chronicles, #3)	The Ironwood Tree (The Spiderwick Chronicles, #4)	0.000898	0.655172	518.057686
74	The Ironwood Tree (The Spiderwick Chronicles, #4)	Lucinda's Secret (The Spiderwick Chronicles, #3)	0.000898	0.710280	518.057686
70	In Too Deep (The 39 Clues, #6)	The Viper's Nest (39 Clues, #7)	0.000508	0.462366	488.992070
...	...	...	...	...	...
3	Three Times Lucky (Mo & Dale Mysteries, #1)	The One and Only Ivan	0.000579	0.114486	6.540389
11	Island of the Blue Dolphins (Island of the Blu...	Number the Stars	0.000520	0.063128	5.987718
2	The One and Only Ivan	Wonder (Wonder #1)	0.002127	0.121540	2.750225
4	Three Times Lucky (Mo & Dale Mysteries, #1)	Wonder (Wonder #1)	0.000508	0.100467	2.273398
7	Where the Sidewalk Ends	Harry Potter and the Prisoner of Azkaban (Harr...	0.000556	0.054335	1.082955

76 rows × 5 columns

What a surprise! The first lines seem to indicate that volumes belonging to the same series are often purchased together.

How can we interpret these results in more detail?

The result table shows a list of antecedent-consequent pairs: customers that bought the antecedent book (**A**) have often bought the corresponding consequent (**C**) in the same purchase. The antecedent and consequent columns contain always just one book each because we set the maximum length of the sequence (max_len parameter) to 2. This has been done just for sake of simplicity. Otherwise, more complex sequences made of combinations of multiple books would be also possible.

For each association rule, some statistics are also available:

SUPPORT - The support indicates how frequent the book association is. This is why we set the minimum_support parameter to a value of 0.05%, meaning that we are taking into account only books combinations that took place at least in 0.05% of the transactions, that is to say in a few tens of occasions. As a matter of fact, it doesn't make sense to consider associations that happened less frequently than that: very rare books are not likely to bring any statistically significant information and they won't have much impact on Mr. Cricket revenues anyway.

CONFIDENCE - The confidence is the probability of purchasing book C when book A is purchased. In general, the higher the confidence, the more robust the association is.

LIFT - When both A and C are popular books, however, the confidence measure can be misleading. An association can be frequent just because both books involved are purchased frequently. Consider for instance the last association proposed. “The Little Prince” has been bought with “Harry Potter” quite frequently, but there is no meaningful association between the two. The thing is that these books are both very popular. The lift measure helps precisely to distinguish these situations. It is defined as the probability of purchasing “Harry Potter” when “The little Prince” is purchased, scaled by the overall probability of purchasing “Harry Potter” anyway. **Only combinations with lift > 1 are actually meaningful**.

It's interesting to explore the combinations with intermediate lift values, as shown below. A few valuable associations of books not belonging to the same series were discovered. Notice for instance “The Cat in the

Hat” and “The Very Hungry Catarpillar”, or “James and the Giant Peach” and “The BFG”. These associations are not obvious, and it would not be easy to spot them without a statistical analysis.

```
rules_df[(rules_df['LIFT']<30) & (rules_df['LIFT']>5)]
```

The output is:

	ANTECEDENT	CONSEQUENT	SUPPORT	CONFIDENCE	LIFT
6	A Light in the Attic	Where the Sidewalk Ends	0.000898	0.302789	29.616249
5	Where the Sidewalk Ends	A Light in the Attic	0.000898	0.087861	29.616249
40	The Bad Beginning (A Series of Unfortunate Eve...	The Reptile Room (A Series of Unfortunate Even...	0.004066	0.209246	28.012112
41	The Reptile Room (A Series of Unfortunate Even...	The Bad Beginning (A Series of Unfortunate Eve...	0.004066	0.544304	28.012112
10	The Marvelous Land of Oz (Oz, #2)	The Wonderful Wizard of Oz (Oz, #1)	0.000567	0.333333	26.065003
25	The Lorax	The Cat in the Hat	0.000662	0.139303	24.657008
24	The Cat in the Hat	The Lorax	0.000662	0.117155	24.657008
36	The Bad Beginning (A Series of Unfortunate Eve...	The Ersatz Elevator (A Series of Unfortunate E...	0.001371	0.070560	20.514904
37	The Ersatz Elevator (A Series of Unfortunate E...	The Bad Beginning (A Series of Unfortunate Eve...	0.001371	0.398625	20.514904
15	The Bad Beginning (A Series of Unfortunate Eve...	The Austere Academy (A Series of Unfortunate E...	0.001690	0.086983	19.624981
14	The Austere Academy (A Series of Unfortunate E...	The Bad Beginning (A Series of Unfortunate Eve...	0.001690	0.381333	19.624981

Now that the Blue Fairy data scientists have understood the data using the apriori analysis, they can play with the notebook and adjust the parameters of the Apriori algorithm until they find a configuration they are happy with. For instance, it might be a good idea to add a lift lower bound with the min_lift input parameter.

Save Results

Last thing left to do is to save the list of associations. Notice that ap.result_ is also a HANA dataframe, so it exists only in memory and it will be gone forever if the connection with HANA is dropped. If you want to persist the data in our HANA DB, you can use the save method as follows:

```
ap.result_.save(where='APRIORI_BOOK_ASSOCIATION',force=True)
```

After running the line above, you should be able to see that a new table named APRIORI_BOOK_ASSOCIATION has been created in your schema. Now it's all done you can happily close the connection to HANA.

```
conn.close()
```

The next step would then be operationalizing the association model using SAP Data Intelligence pipelines. Mr. Cricket is almost ready to have his special book recommendation application to help his business grow.