INTERNAL

SAP Data Intelligence hands-on exercises

This document will guide you step-by-step through the process of training and implementing a text analysis and developing a cluster machine learning model using Python and SAP DI Pipelines.

www.sap.com/contactsap

THE BEST RUN SAP®

Table of Contents

DISCLAIMER

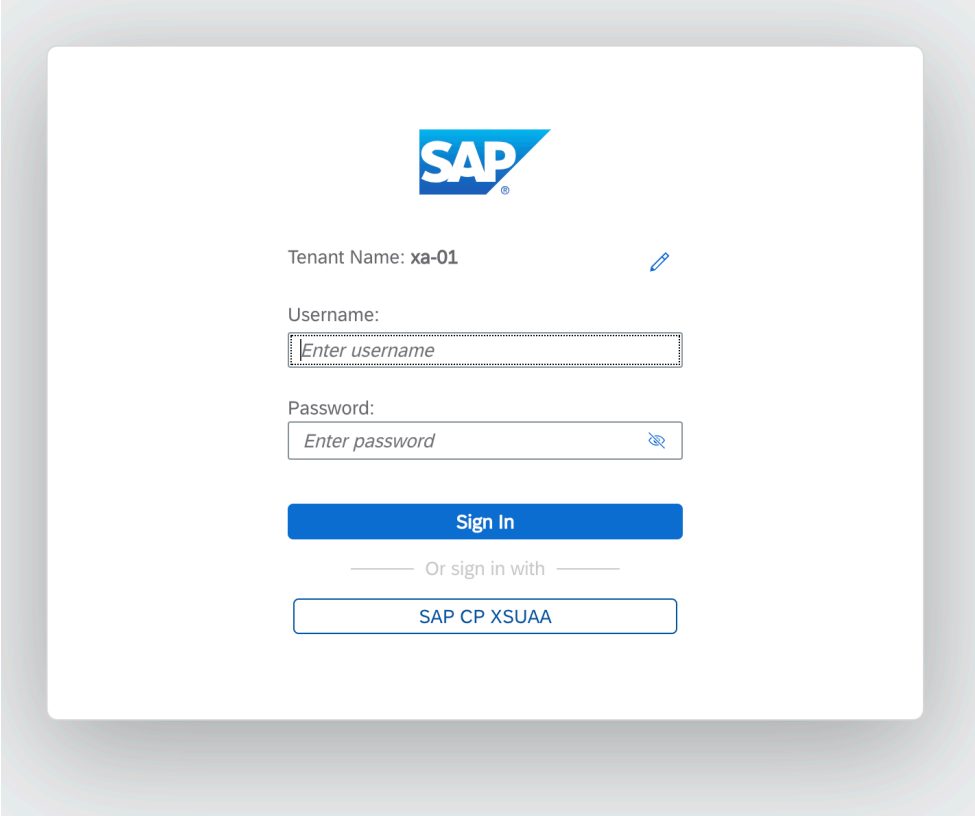
OBJECTIVE

The objective of this exercise is to give you an overview of how you can use the machine learning capabilities in SAP Data Intelligence.

SCENARIO

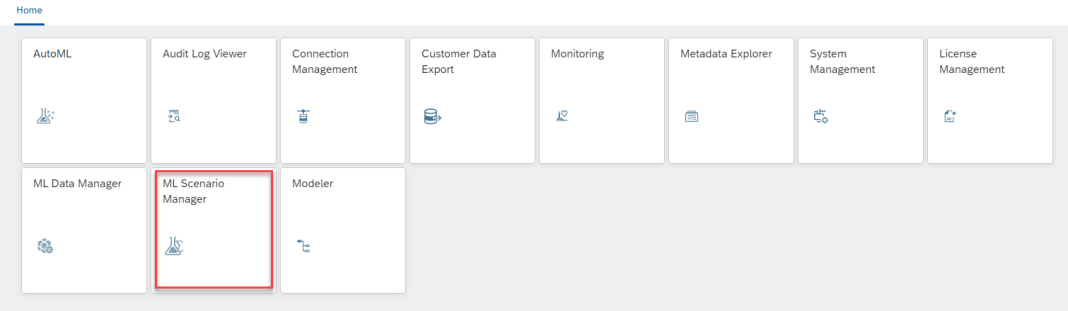
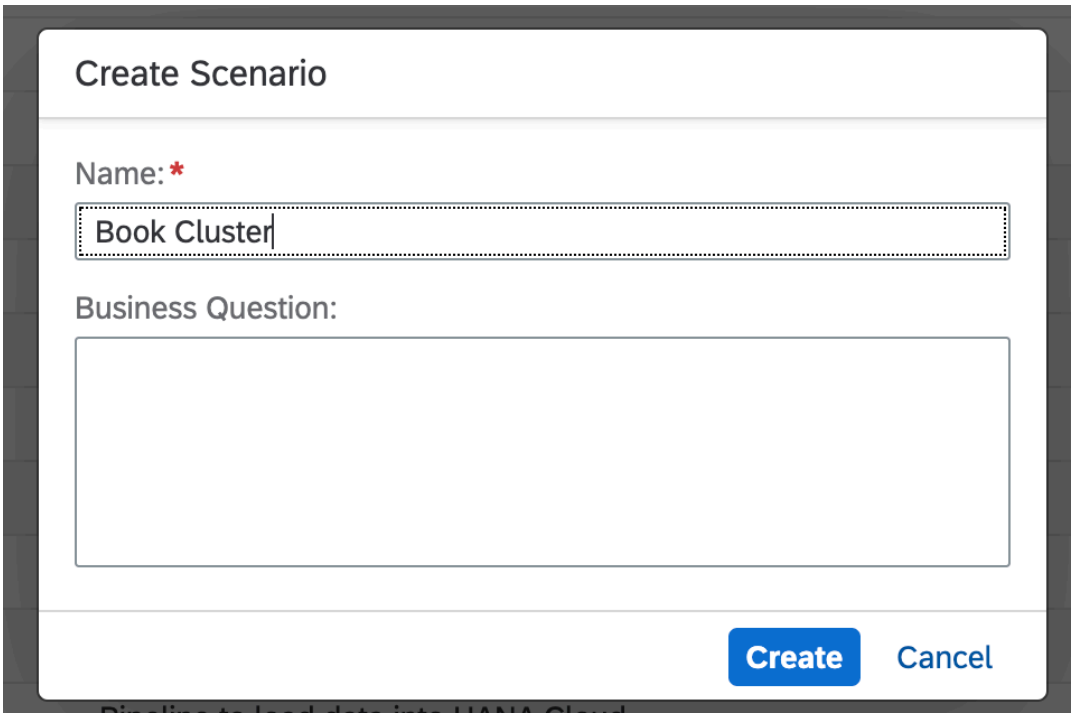
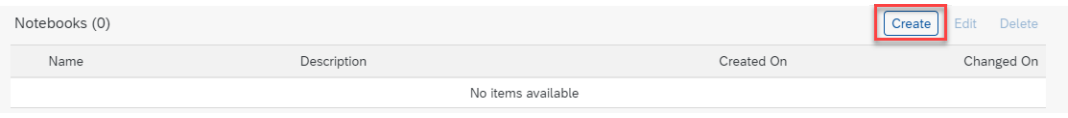
Books are grouped together in a bookshop based on their similarity, so that customers browsing for a book will find lots of similar books on the same shelf. This exercise analyzes the book description data using Python text mining algorithms and then uses this information to assign each book to a cluster. The books within a cluster are as similar as possible (based on the book description), so they are as homogeneous as possible, and there is as wide a difference as possible between clusters, so the different clusters are heterogeneous.

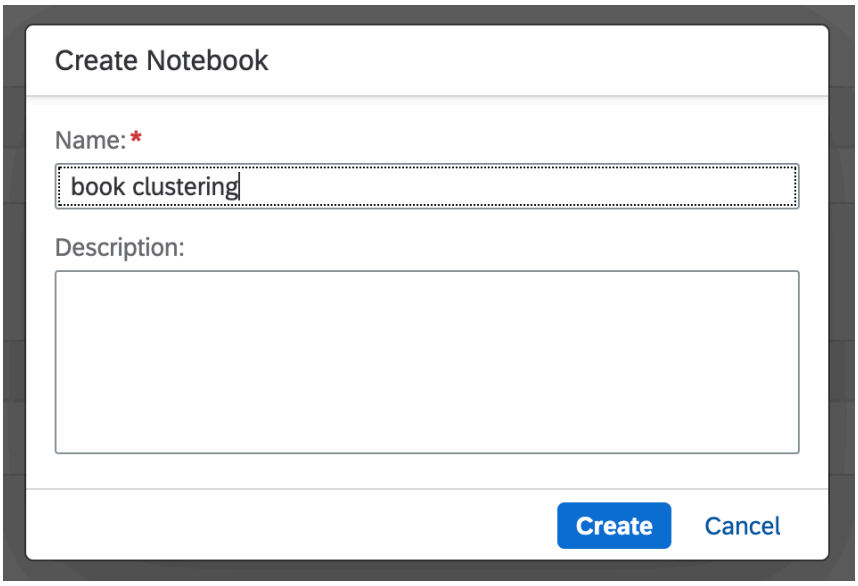
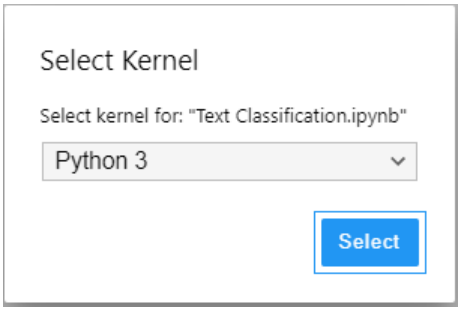
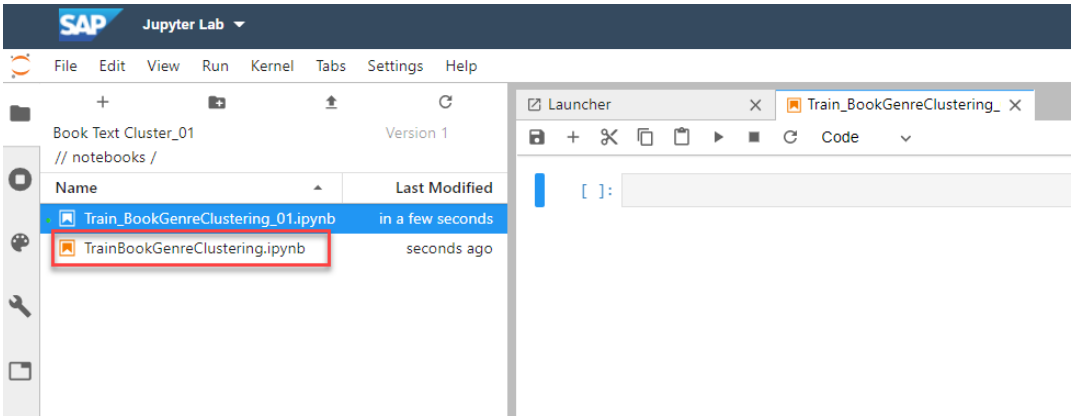
ENVIRONMENT ACCESS

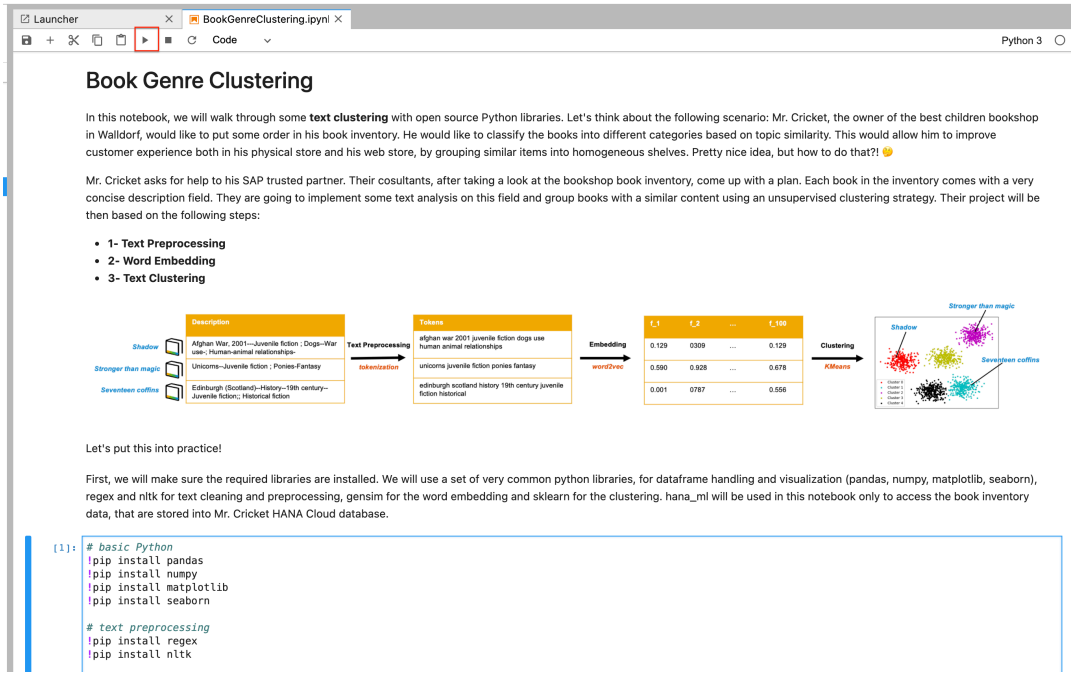
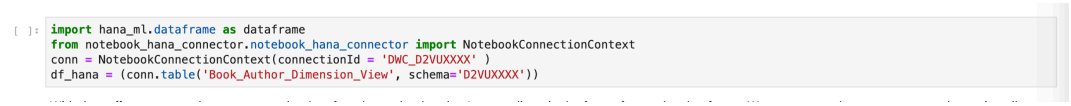
Explanation	Screenshot
Login to your SAP Data Intelligence Cloud tenant with the credentials provided by your instructor	

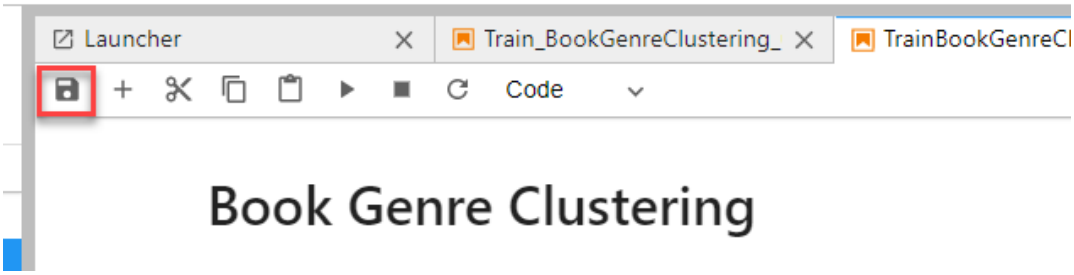
STEP 1 – USE A JUPYTER NOTEBOOK

A Jupyter Notebook environment is used to explore the data, and to run predictive model tests to compare the accuracy of different algorithms and the best settings for the hyper-parameters for the algorithms.

Explanation	Screenshot
Click to open ML Scenario Manager	
Click the Create button. Create a new scenario. Name the scenario "**Book Cluster <your user id>**". You see the empty scenario. First, you will use the Notebooks to explore the data and to script the text analysis and cluster model in Python. Next, pipelines will bring the code into production. Executions of these pipelines will create Machine Learning models, which are then deployed as REST-API for inference.	
In the Notebooks section, click Create to create a new notebook.	

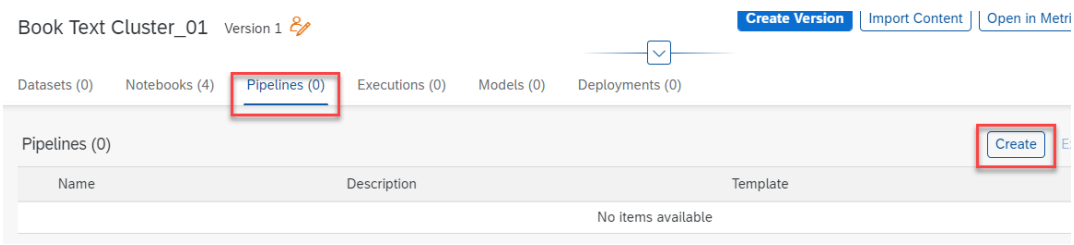
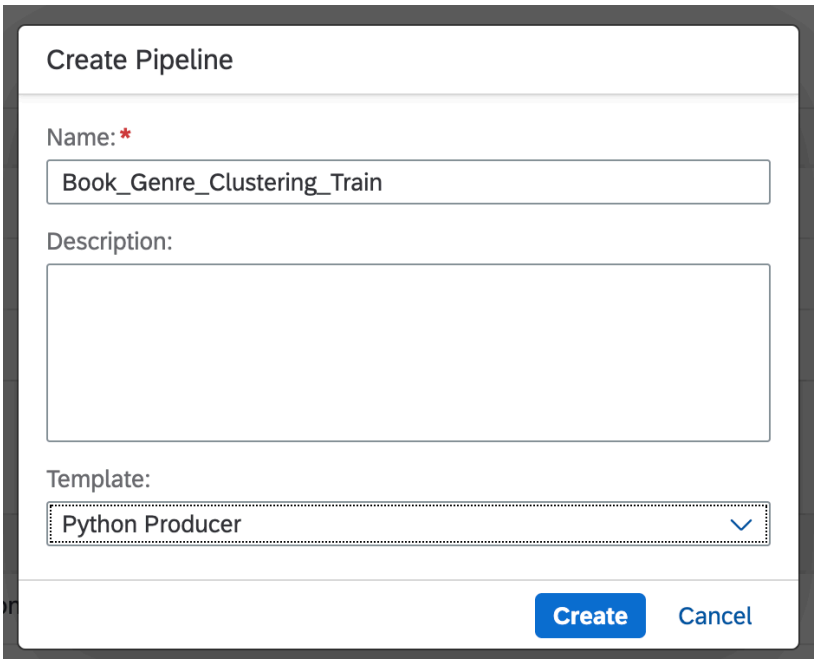
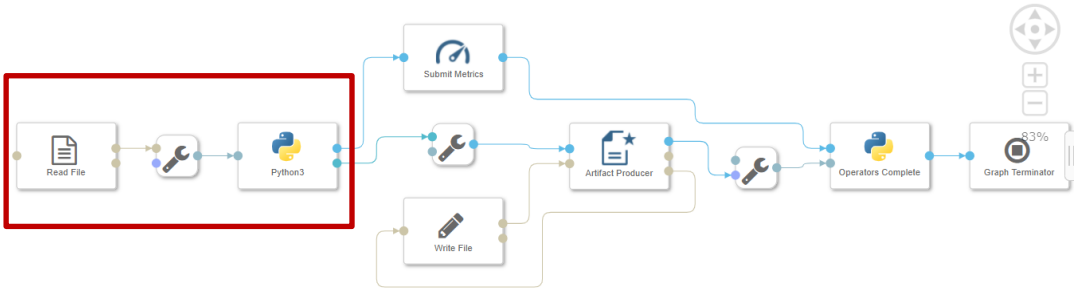
Explanation	Screenshot
Name the notebook “book clustering”. As soon as you click Create the Jupyter Lab will open in a new browser tab.	
Select the Python 3 Kernel option.	
A Jupyter notebook with the commented Python code was prepared for you and is available in the official GitHub repository at this link: DV220_Exercise01_Book_Genre_Clustering.ipynb Use the Upload Files function to upload the notebook into the console.	 NB: The notebook name in the screenshot is only an example. The notebook name is DV220_Exercise01_Book_Genre_Clustering.ipynb.

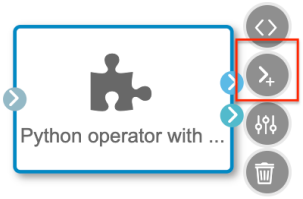
Explanation	Screenshot
Double click on the notebook that is uploaded. Step through the code to check it works correctly. Highlight each cell in sequence and click the arrow button to run the selected cell and advance.Note that line 9 loads the word embedding data and line 19 runs a tsne model. Both of these steps will take a few minutes to complete. Analyze the results of your exploratory data analysis and understand the text analysis and cluster model results.	
NB: The input data for this exercise are stored in the Book_Author_Dimension_View that you created in exercise DV200_Exercise01. In order to access it, please, use your connection to DWC, named DWC_<USER_ID>, and the proper schema, that is equal to your <USER_ID>.	

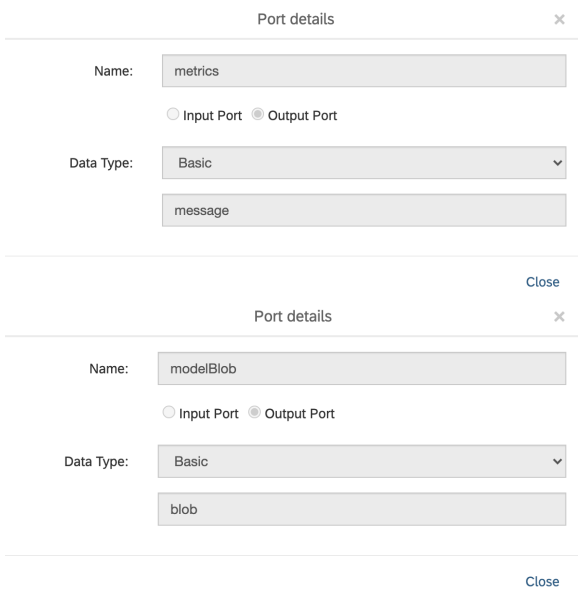
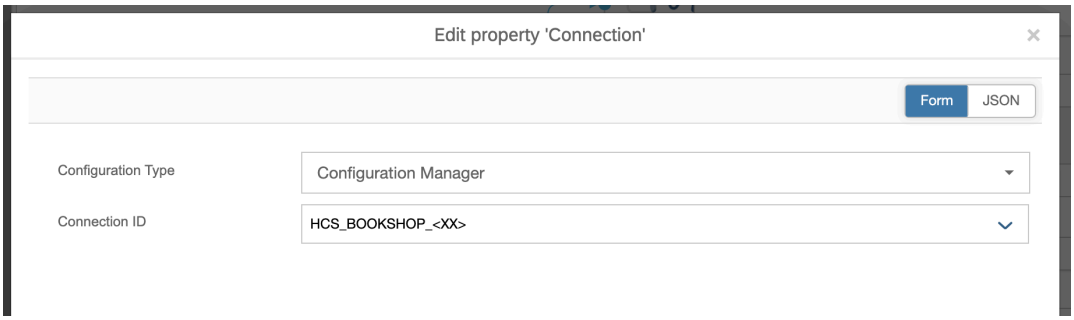
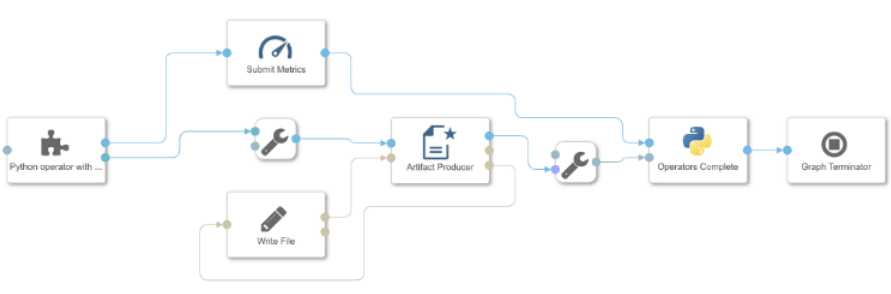
Explanation	Screenshot
Once you have stepped through to the end of the notebook, and there are no errors, save the notebook.	

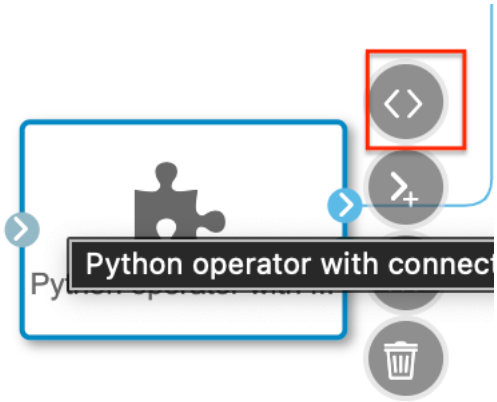
STEP 2 – BUILD MODEL PIPELINES

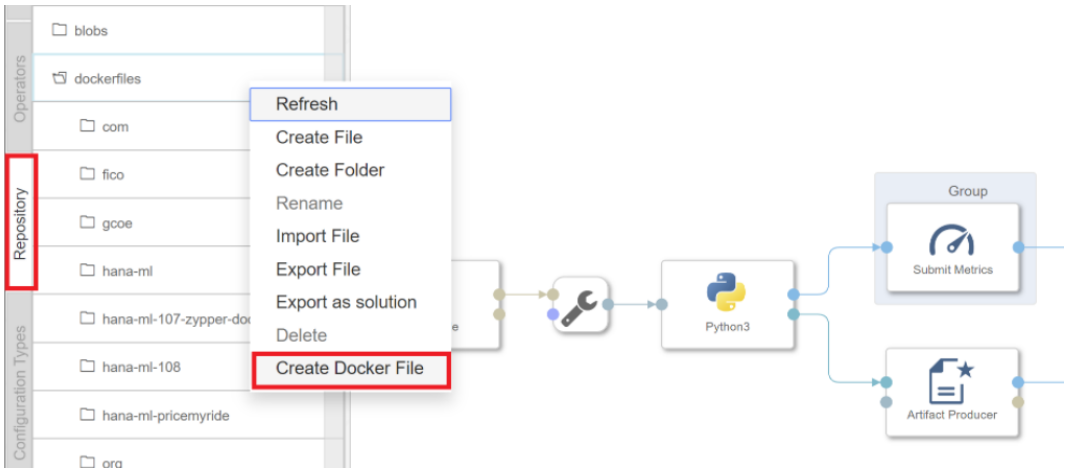
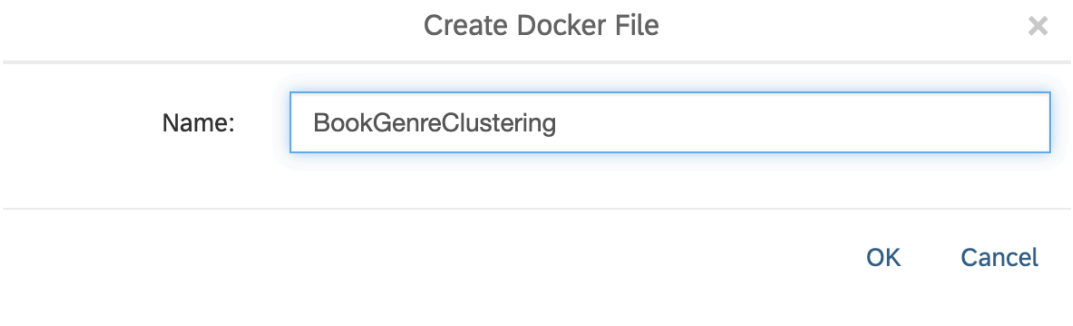
Model pipelines are used to operationalize the cluster model. Now that you have prepared the data, identified the best algorithm to use and which hyper-parameters work best, you want to take the model to production. You will build two pipelines. The first one is used to automate the model training, while the second one is used to inference the model on new data.

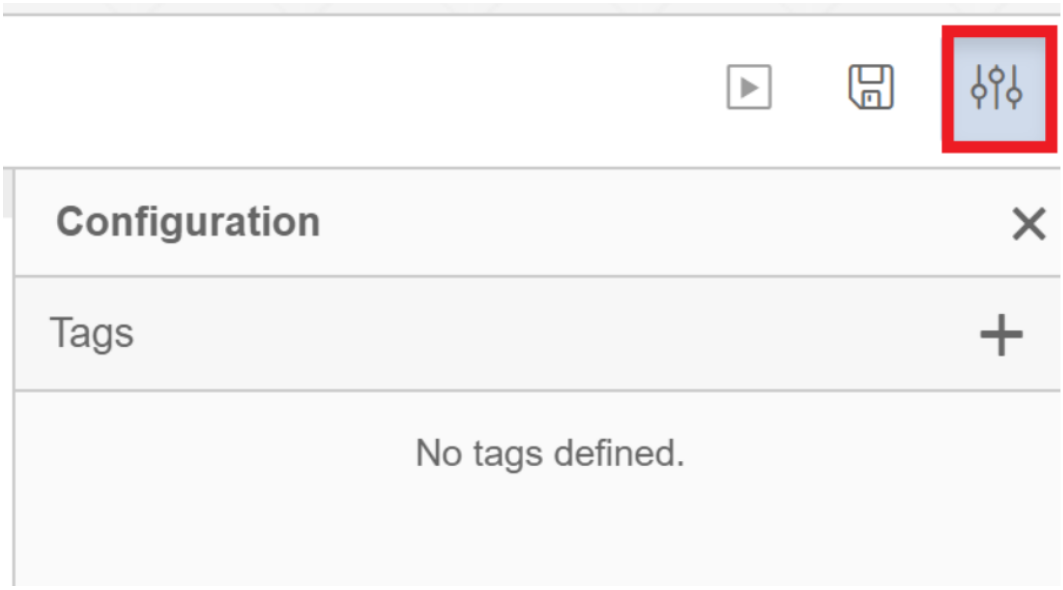
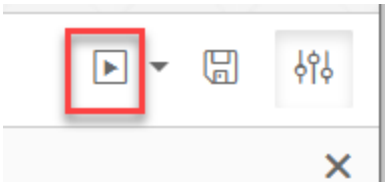
Explanation	Screenshot
	CREATE A DI PIPELINE TO TRAIN THE ML MODEL
To create the graphical pipeline to train the model, go to your ML Scenario's main page, select the "Pipelines" tab and click Create.	
Name the pipeline "Text Clustering Train" and select the "Python Producer" template. Click Create.	
Let's see what the template pipeline does. The pipeline loads data with the "Read File" operator. The data is passed to a Python operator, where the	

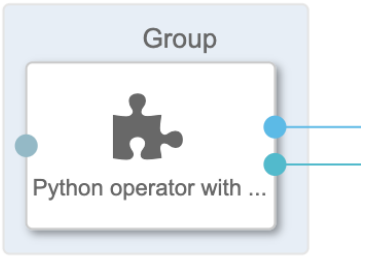
Explanation	Screenshot
ML model is trained. The same Python-operator stores the model in the ML Scenario through the “Artifact Producer”. The Python-operator’s second output passes a model quality metric to the ML Scenario. Once both model and metric are saved, the pipeline’s execution is ended with the “Graph Terminator”. We need to adjust the pipeline template, in particular the part highlighted in red.	
Since for us the input data are not stored in a file, but in HANA Cloud, we don’t need the Read File operator. We will adapt the custom python operator we built in the previous exercise (DV210_Exercise01). Insert a “Python with HANA ML connection operator” in the canvas. Click on the “Add port” symbol to add two output ports. Configure the ports with the parameter shown in the picture,	

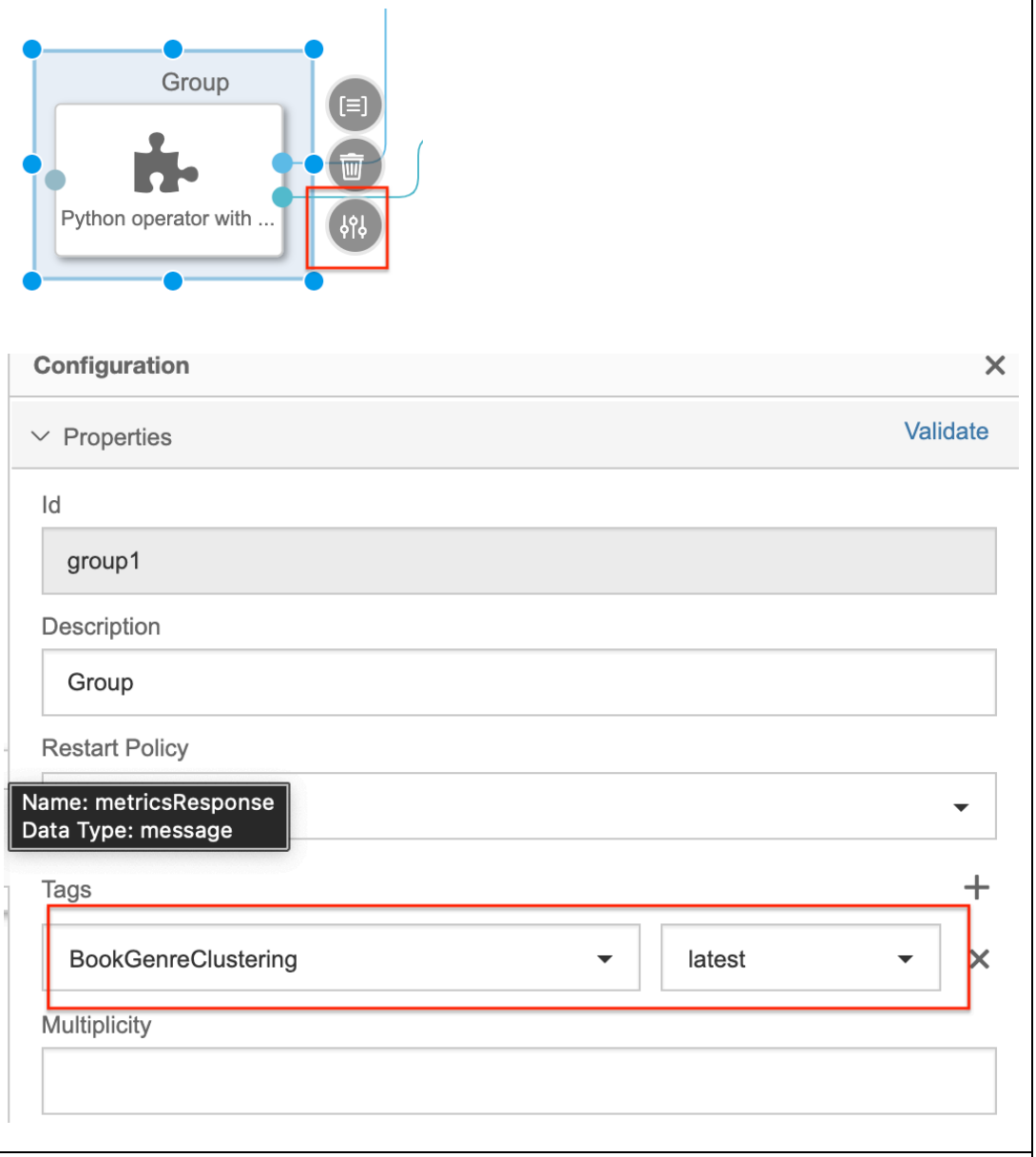
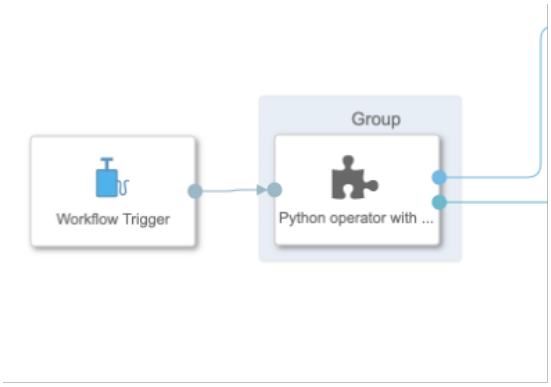
Explanation	Screenshot
paying attention to the names and types. The names are used in the training Python code we will execute.	
In the Configuration panel, open the “Connection” configuration, set “Configuration type” to “Connection Management” and select from the drop -down menu the **connection ID** related to your Hana Cloud instance. You have to use your connection to DWC, named DWC_<USER_ID>.	
Delete the Read File operator, and the Conversion operator. Replace the Python Operator with the custom operator that you have just configured. Your pipeline should look like in the picture.	

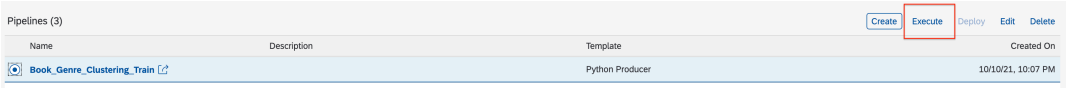
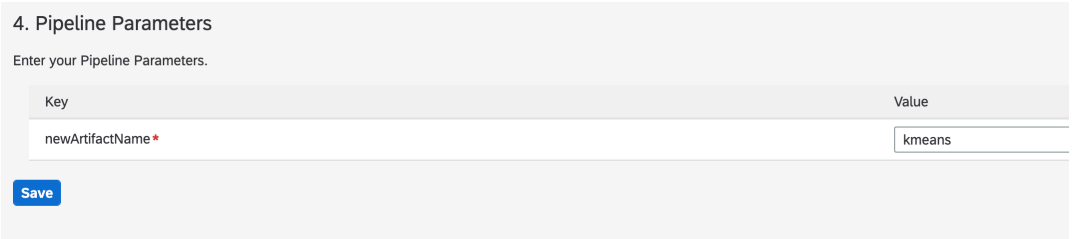
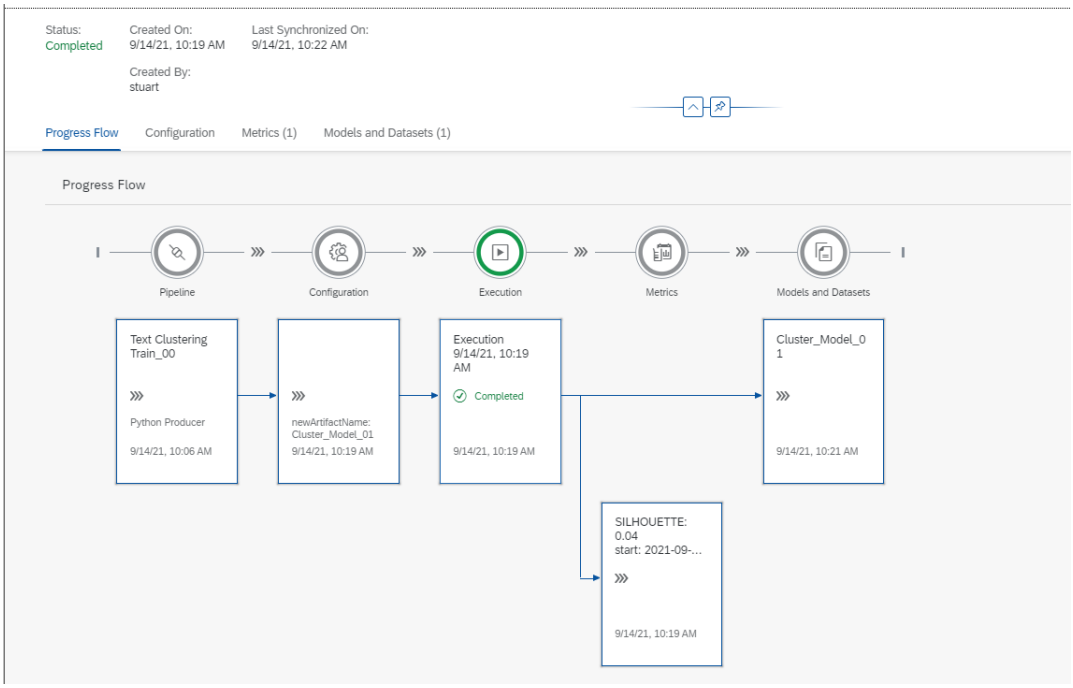
Explanation	Screenshot
Please, don't touch the second and third Conversions operator! Please, notice also that the second and third Conversion operators are different: they are a ToMessage and a ToString converter, respectively.	
Next, adjust the Python code that analyzes the text data and trains the cluster model. Select the custom operator and click the Script option.	
The template code opens up. It shows how to pass the text analysis, model and metrics into the ML Scenario. **Carefully** copy and paste to replace the existing code with the code given here, so that we can operationalize the clustering model we developed in the notebook. In this code you have to customize the name of the table to use and the schema where the	https://github.com/SAP-samples/btp-data-to-value-workshop/blob/main/02-data-modeling%26processing/exercises/code_snippets/dv220_train.py

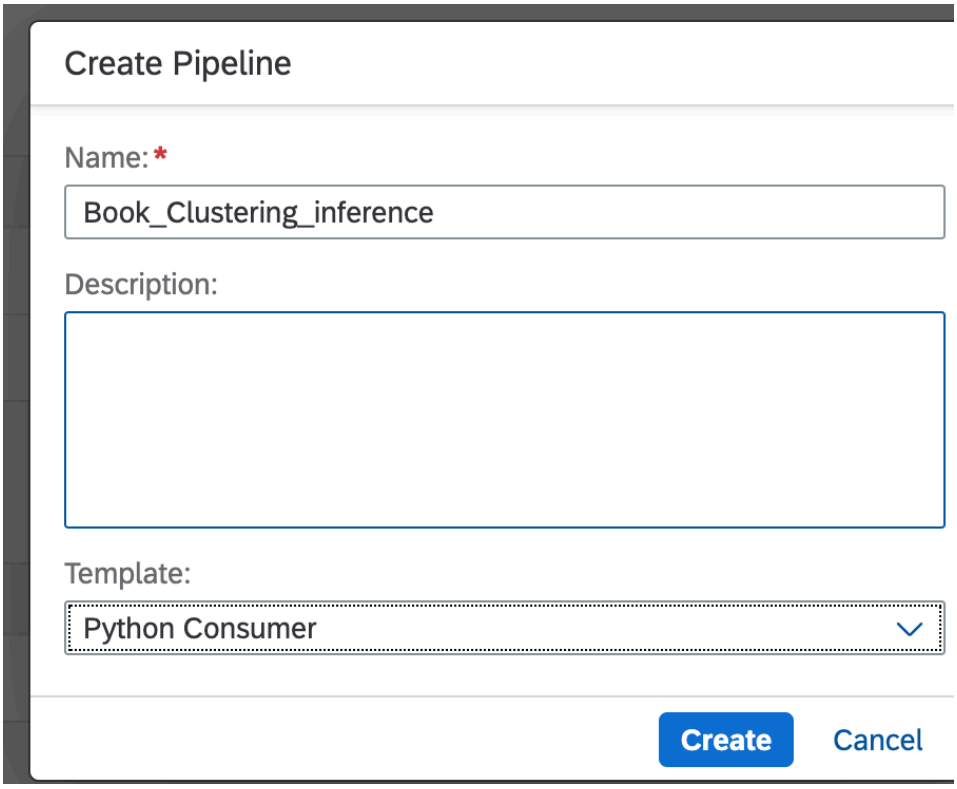
Explanation	Screenshot
table resides. We will use again the Book_Author_Dimension_View prepared during exercise DV200_Exercise01 and that resides in DWC under the schema named like your <USER_ID>.	
Close the Script-window, then click “Save” in the menu bar.	
You need to create a Docker image for the custom python operator. This gives the flexibility to leverage virtually any Python library. The Docker file installs the necessary libraries. You find the docker files by clicking into the “Repository” tab on the left. To create a new one, right-click the “**dockerfiles**” folder and select “Create Docker File”.	
Name the file **BookGenreClustering**.	

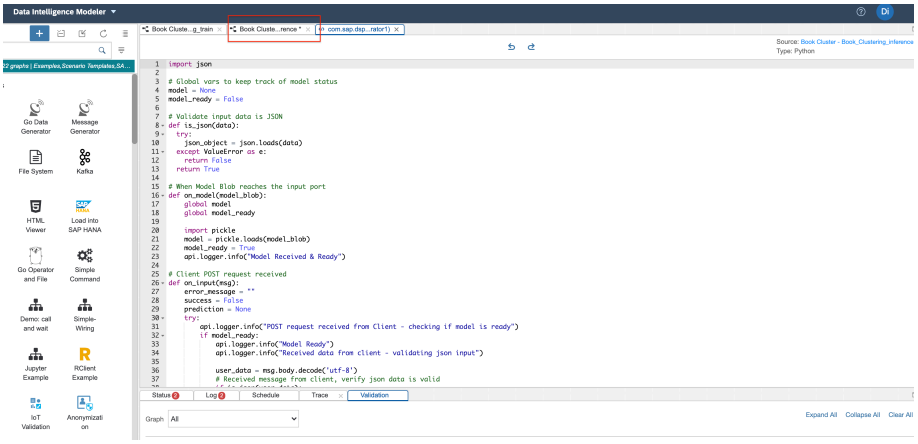
Explanation	Screenshot
Enter this code into the Docker File window. This code leverages a base image that comes with SAP Data Intelligence and installs the necessary libraries on it. NB: Please, don't use the Docker file definition used for the book recommendation exercise.	FROM $com.sap.sles.base RUN pip install --user numpy RUN pip install --user pandas RUN pip install --user sklearn RUN pip install --user hana_ml RUN pip install --user regex RUN pip install --user nltk RUN pip install --user gensim
Open the Configuration panel for the Docker File with the icon on the top-right hand corner.	
Assign a tag to the docker image.	
Now save the Docker file and click the "Build" icon to start building the Docker image. Wait a few minutes and	

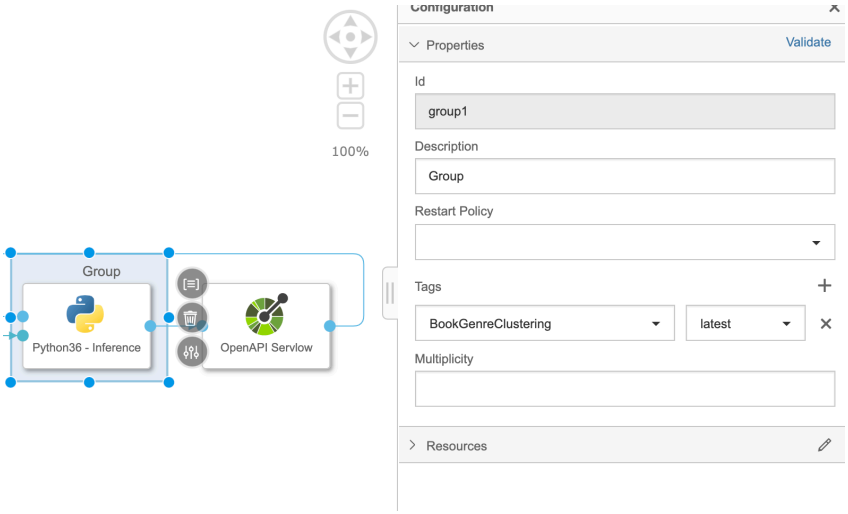
Explanation	Screenshot
you should receive a confirmation that the build completed successfully.	
Now you need to configure the custom operator, which trains the model, to use this Docker image. Go back to the graphical pipeline and right-click the custom operator and select “Group”.	

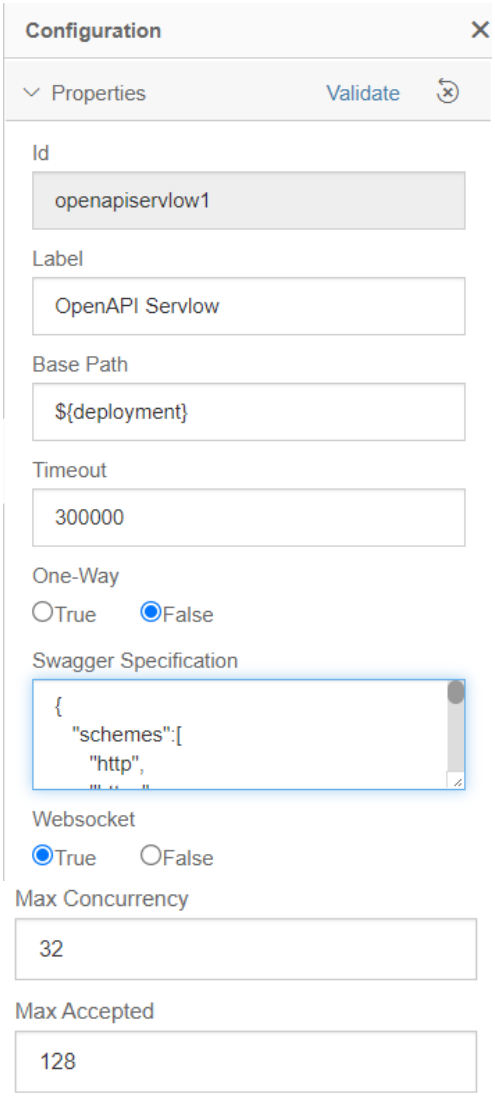
Explanation	Screenshot
You specify which Docker image should be used. Select the group which surrounds the “Python 3” Operator. In the group’s Configuration select the docker image you have built.	
Select the Workflow Trigger operator in the operators panel. Add it to the pipeline and connect it to the trigger port of the custom operator. Save your pipeline. Please, make sure your Python operator has an input port named	

Explanation	Screenshot
“trigger” of type “string”.	
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.	
Click “Step 3” and then “Step 4” to skip the optional steps until you get to the “Enter your Pipeline Parameters”. Set “newArtifactName” Value to “kmeans”. Click Save. The trained model will be saved under this name.	
Wait a few minutes until the pipeline executes and completes. The metrics section shows the trained model’s silhouette metric.The model itself was saved successfully under the name “kmeans”.The model has a Technical Identifier. NB: Any errors can be investigated in the pipeline editor.	

Explanation	Screenshot
	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. Name the pipeline “Book Clustering Consumer”. Select the template “Python Consumer”. This template contains a pipeline that provides a REST-API.	
The “OpenAPI Servlow” operator provides the REST-API. The “Artifact Consumer” loads the trained model from your ML scenario.The “Python36 – Inference” operator ties the two operators together. It receives the input from the REST-API call (here the user’s text input book description) and uses the loaded model to assign the cluster, which is then returned by the “OpenAPI Servlow”	

Explanation	Screenshot
to the client, which had called the REST-API.	
You only need to update the script of the Python3.6 Inference operator. Select the Python3.6 operator in the pipeline and click on the script icon	
Carefully copy and paste to replace the whole code with the code given here. Close the editor window.	https://github.com/SAP-samples/btp-data-to-value-workshop/blob/main/02-data-modeling%26processing/exercises/code_snippets/dv220_consumer.py
Go back to the pipeline tab and save the pipeline.	

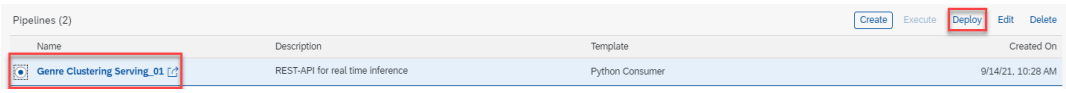
Explanation	Screenshot
Finally, you need to assign the Docker image to the “Python36 – Inference” operator. As before, right-click the operator and select “Group”. Add the tag of your docker image Save the changes. NB: Please, don’t use the Docker file definition used for the book recommendation exercise, but the one created for the training named BookGenreClustering.	

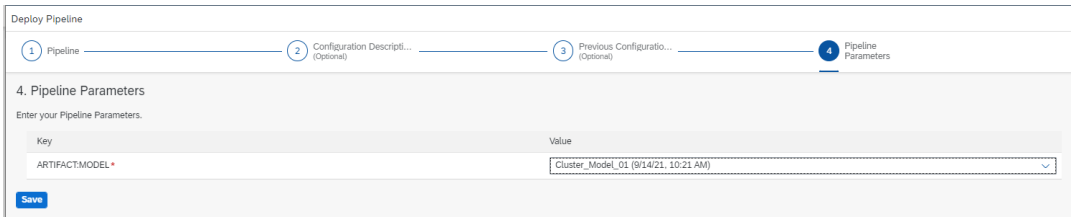
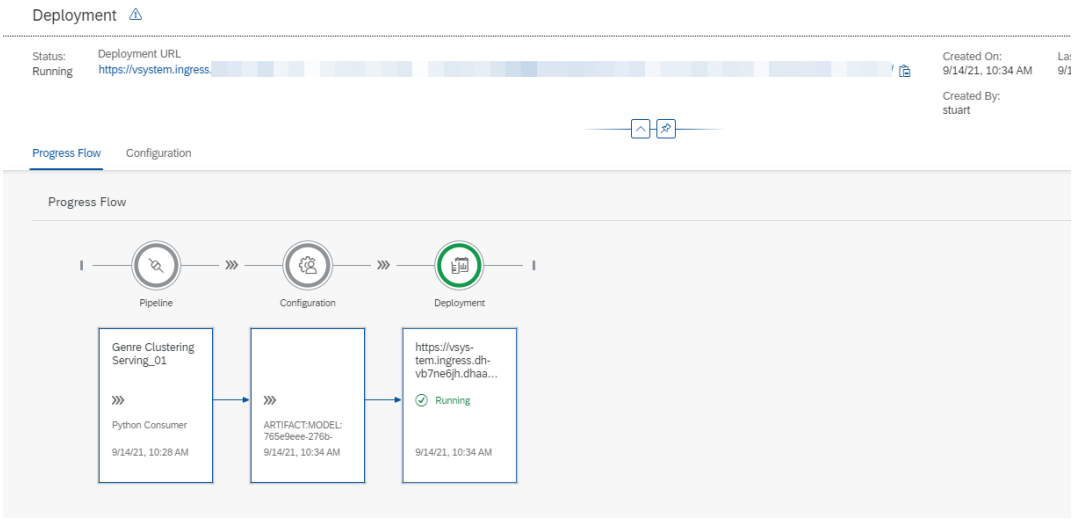
Explanation	Screenshot
Click on OpenAPIServlow and have a look at the configuration.	

Notice in particular the content of the Swagger Specification, but don't change anything here!	{
	"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",

Explanation	Screenshot

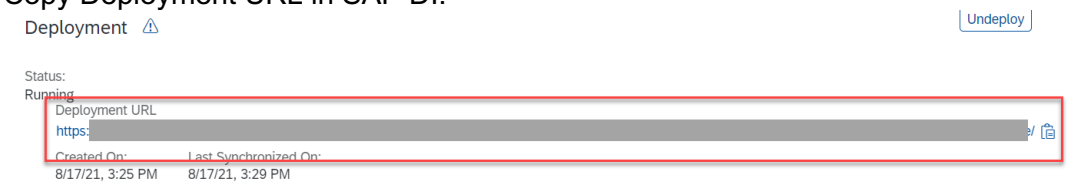
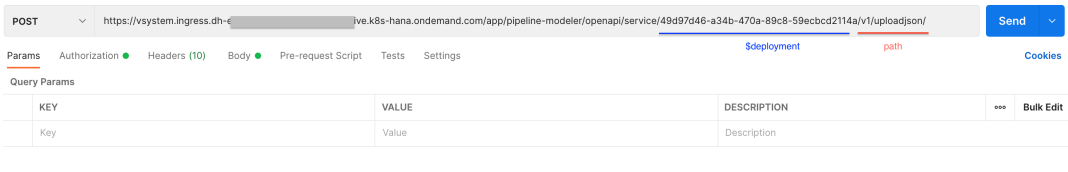
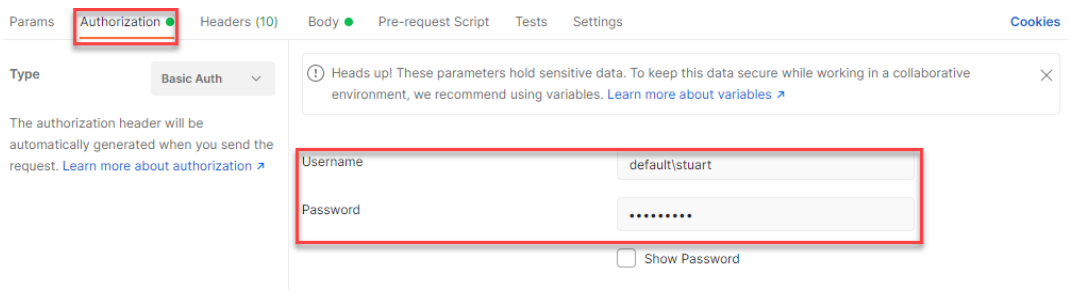
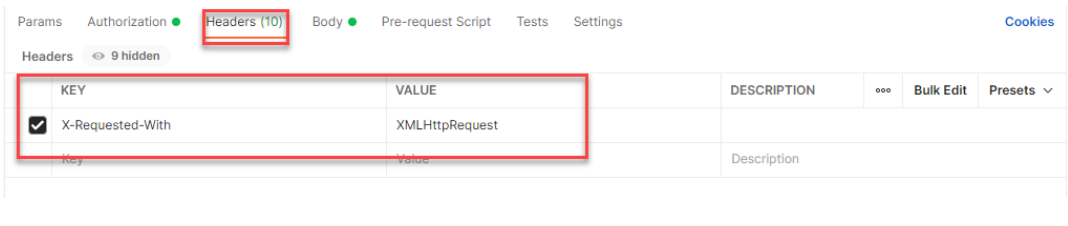
	"url":"http://www.apache.org/licenses/LICENSE-2.0.html"
	},
	"version":"1.0.0"
	},
	"basePath":"/$deployment",
	"paths":{
	"/v1/uploadjson":{
	"post":{
	"description":"Upload data in json format",
	"consumes":[
	"application/json"
	],
	"produces":[
	"application/json"
	],
	"summary":"Upload JSON data to be used in the Python operator's script",
	"operationId":"upload",
	"parameters":[
	{
	"type":"object",
	"description":"json data",
	"name":"body",
	"in":"body",
	"required":true
	}
	],
	"responses":{
	"200":{
	"description":"Data uploaded"
	},
	"500":{
	"description":"Error during upload of json"
	}
	}
	}
	}
	},
	"definitions":{
	},
	"securityDefinitions":{
	"UserSecurity":{
	"type":"basic"
	}
	}
	}

Go back to the ML Scenario. Now deploy the new pipeline. Select the pipeline and click “Deploy”.	

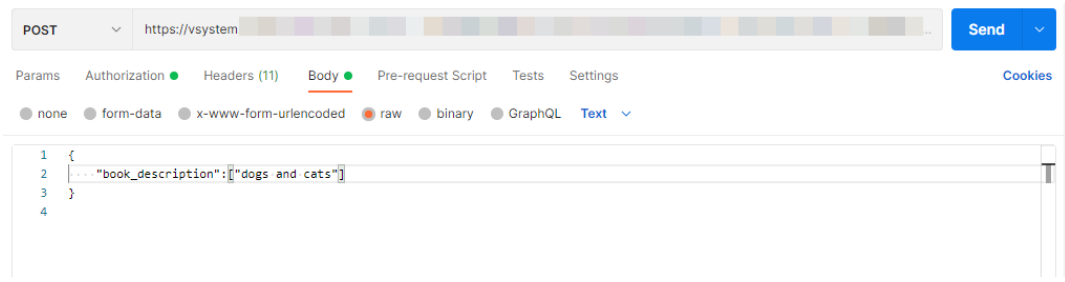
Explanation	Screenshot
Click through the screens until you can select the trained model from the drop-down. Click “Save”.	
After a few minutes the pipeline will start running. NB: Any errors can be investigated in the pipeline editor. NB: Once the consumer pipeline is started, it will run until you stop 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!**	

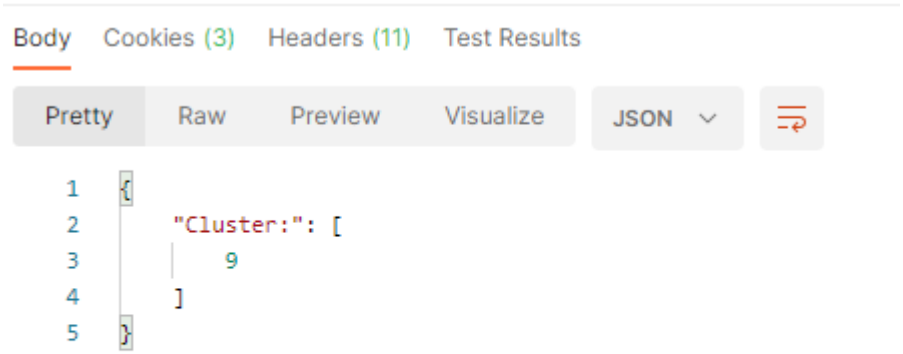
STEP 3 – USE YOUR CLUSTER MODEL

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

Explanation	Screenshot
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”.	Copy Deployment URL in SAP DI:  To Postman: 
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.	
Go to the “Headers” tab and enter the key “X-Requested-With” with value “XMLHttpRequest”.	

Explanation	Screenshot

Finally, pass the input data to the REST-API. Select the “Body” tab, choose “raw” and enter the syntax given here.	
	{
	"book_description”: ["dogs and cats"]
	}

Press “Send” and after a few minutes you will see the genre prediction that comes from SAP Data Intelligence. Try the REST-API with different text to see how the cluster allocations change.	
You have now completed the exercise.	