



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

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See www.sap.com/copyright for additional trademark information and notices.

Table of Contents

DISCLAIMER	4
OBJECTIVE	4
SCENARIO	4
ENVIRONMENT ACCESS	5
STEP 1 – USE A JUPYTER NOTEBOOK.....	6
STEP 2 – BUILD MODEL PIPELINES	10
STEP 3 – USE YOUR CLUSTER MODEL	25

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.

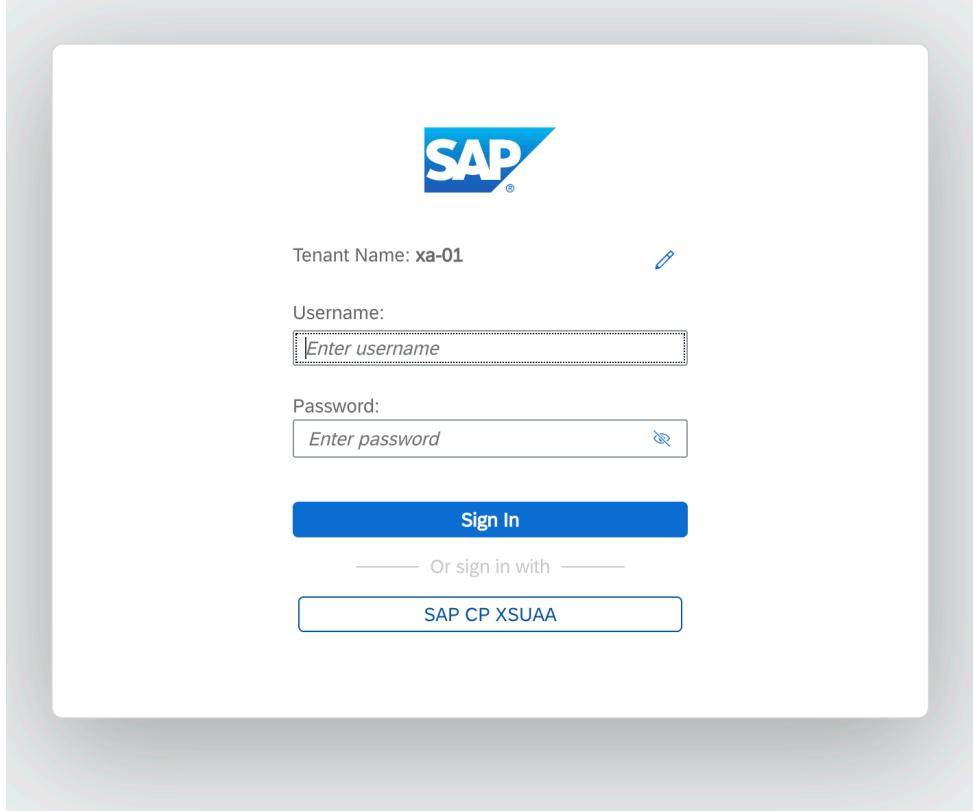
OBJECTIVE

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

SCENARIO

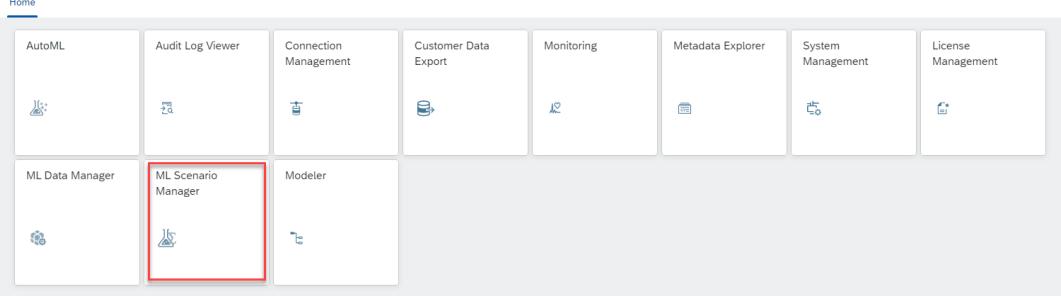
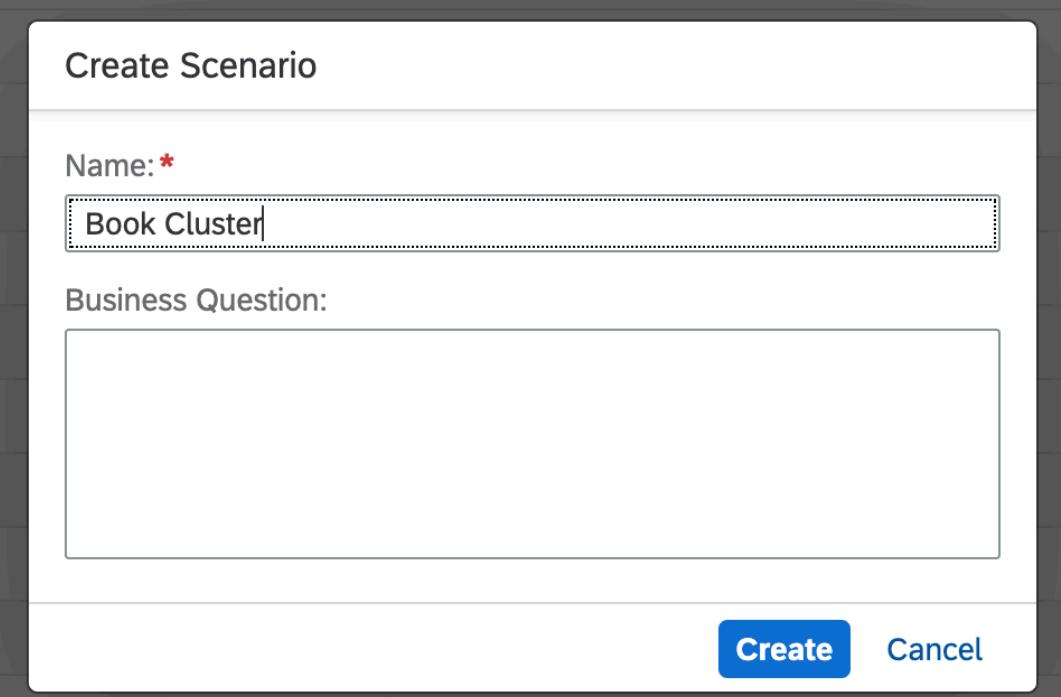
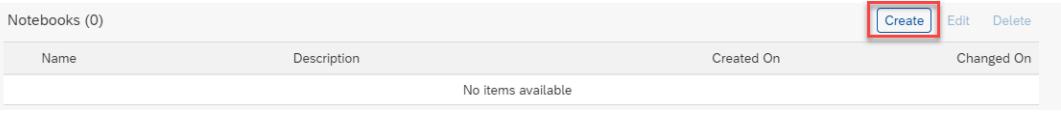
Books are grouped together in a bookshop based on their similarity, so that customers browsing for a book will find lots of similar books on the same shelf. This exercise analyzes the book description data using Python text mining algorithms and then uses this information to assign each book to a cluster. 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.

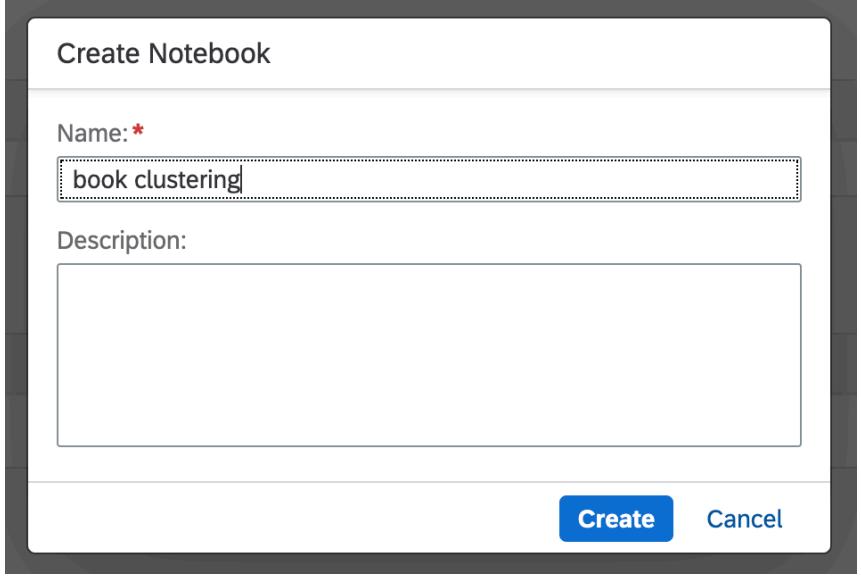
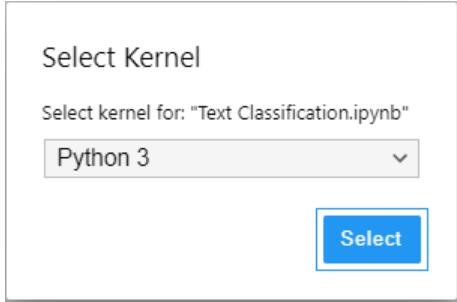
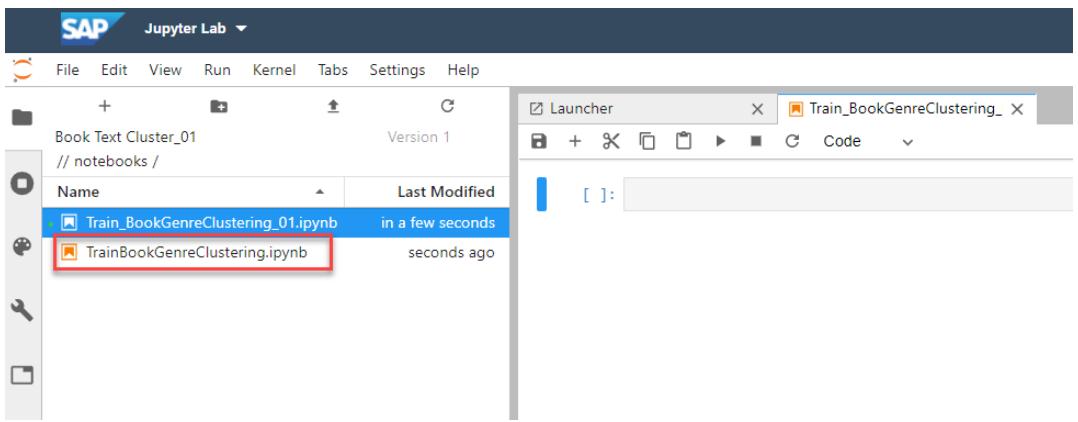
ENVIRONMENT ACCESS

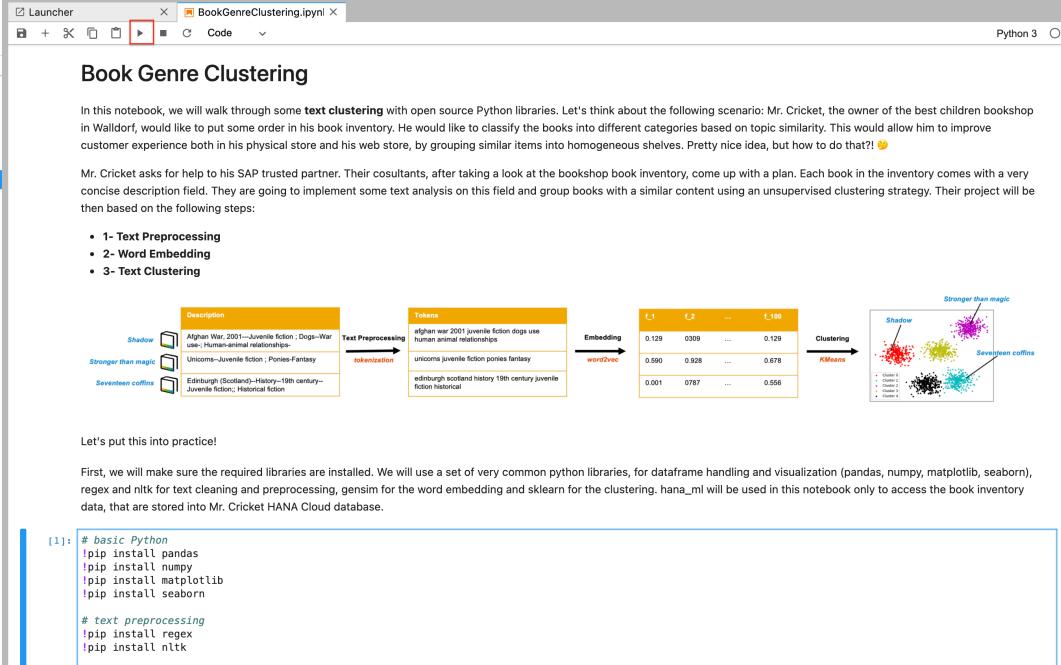
Explanation	Screenshot
Login to your SAP Data Intelligence Cloud tenant with the credentials provided by your instructor	

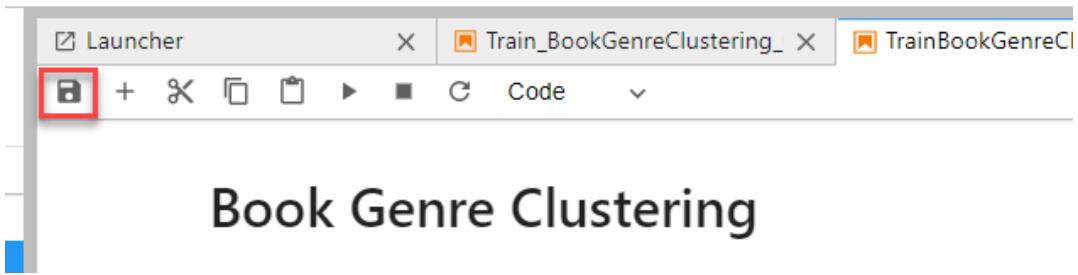
STEP 1 – USE A JUPYTER NOTEBOOK

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	
<p>Click the Create button. Create a new scenario. Name the scenario “Book Cluster <your user id>”.</p> <p>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.</p> <p>Executions of these pipelines will create Machine Learning models, which are then deployed as REST-API for inference.</p>	
In the Notebooks section, click Create to create a new notebook.	

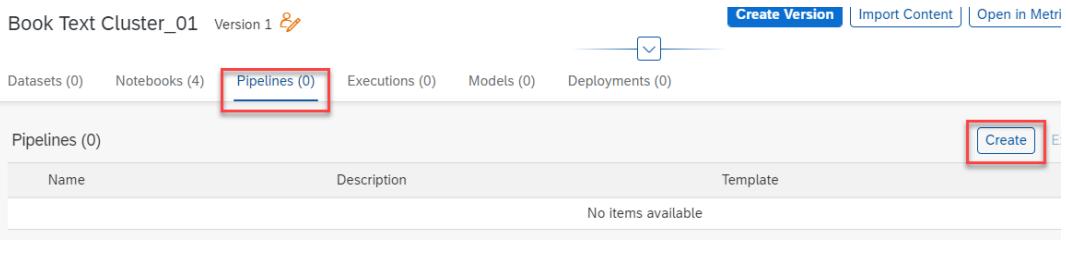
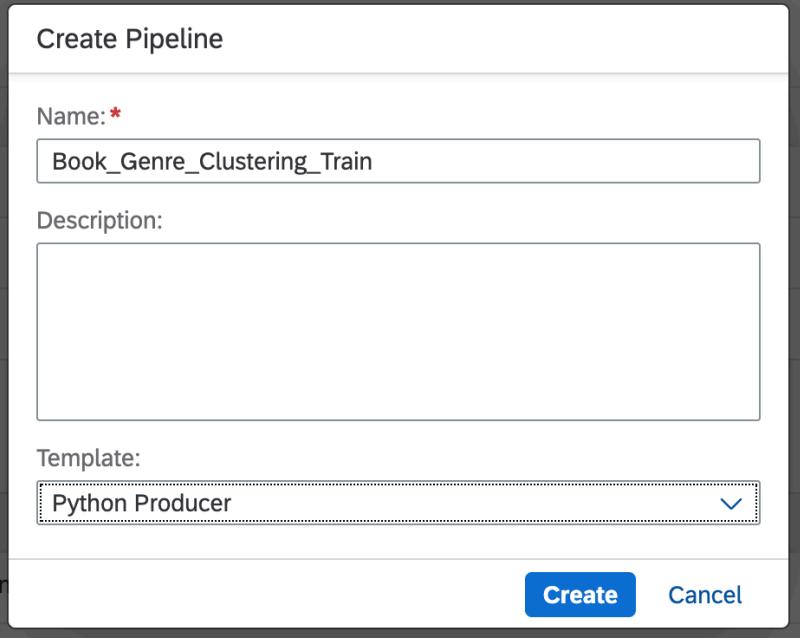
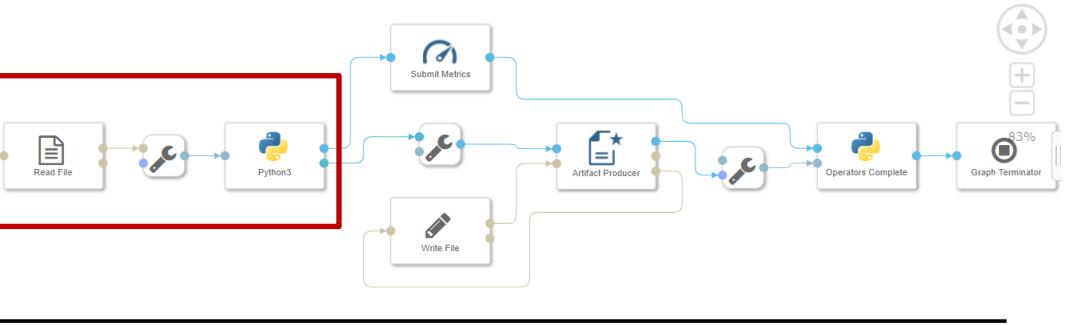
Explanation	Screenshot
<p>Name the notebook "book clustering".</p> <p>As soon as you click Create the Jupyter Lab will open in a new browser tab.</p>	
<p>Select the Python 3 Kernel option.</p>	
<p>A Jupyter notebook with the commented Python code was prepared for you and is available in the official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

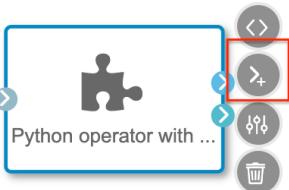
Explanation	Screenshot
<p>Double click on the notebook that is uploaded.</p> <p>Step through the code to check it works correctly.</p> <p>Highlight each cell in sequence and click the arrow button to run the selected cell and advance. Note that line 9 loads the word embedding data and line 19 runs a 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.</p>	 <p>The screenshot shows a Jupyter Notebook interface with the title 'BookGenreClustering.ipynb'. The notebook content is as follows:</p> <ul style="list-style-type: none"> Text Preprocessing: Describes the process of tokenization and creating word embeddings using word2vec. Embedding: Shows a 2D t-SNE plot of word embeddings for four books: 'Shadow', 'Stronger than magic', 'Unicorns', and 'Seventeen coffins'. The plot shows that 'Shadow' and 'Stronger than magic' are clustered together, while 'Unicorns' and 'Seventeen coffins' are clustered together. Code: <pre>[1]: # basic Python !pip install pandas !pip install numpy !pip install matplotlib !pip install seaborn # text preprocessing !pip install regex !pip install nltk</pre>
<p>NB: The input data for this exercise are stored in the Book_Author_Dimension_View that you created in exercise DV200_Exercise01.</p> <p>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>.</p>	<pre>[1]: import hana_ml.dataframe as dataframe from notebook_hana_connector.notebook_hana_connector import NotebookConnectionContext conn = NotebookConnectionContext(connectionId = 'DWC_D2VUXXXX') df_hana = (conn.table('Book_Author_Dimension_View', schema='D2VUXXXX'))</pre>

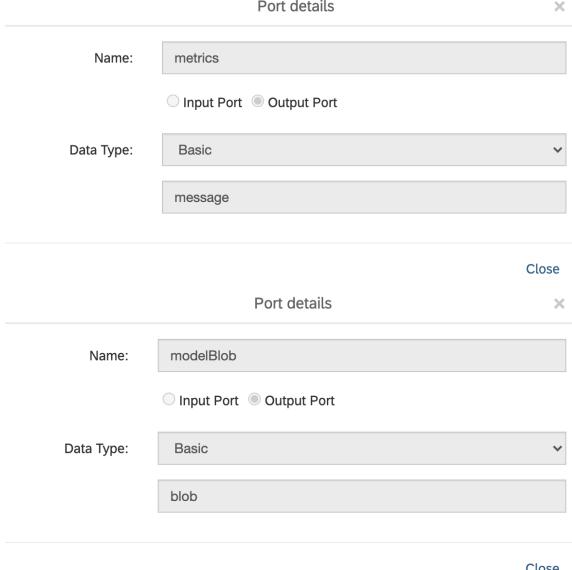
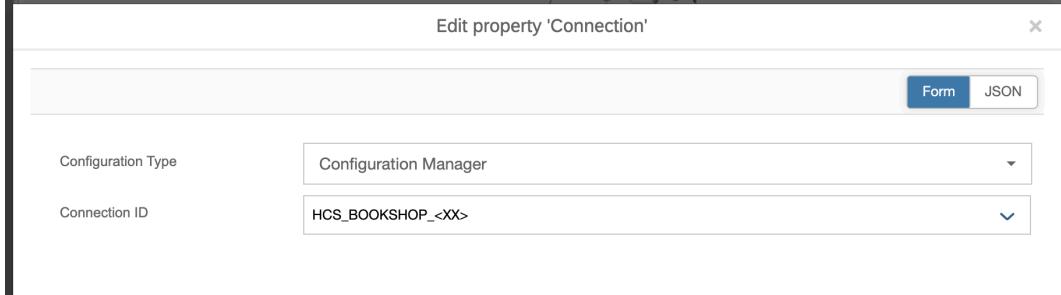
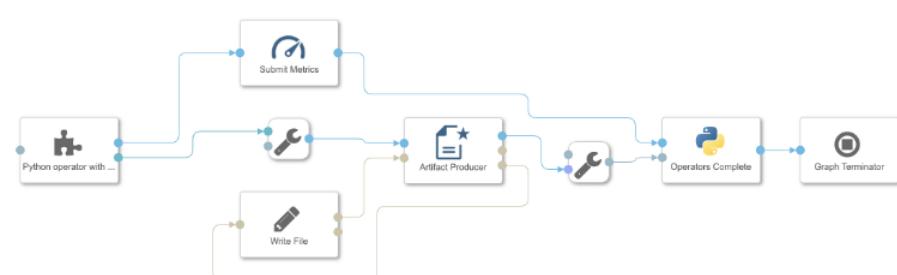
Explanation	Screenshot
Once you have stepped through to the end of the notebook, and there are no errors, save the notebook.	 A screenshot of a Jupyter Notebook interface. The title bar shows three tabs: 'Launcher', 'Train_BookGenreClustering_...', and 'TrainBookGenreCl...'. The main area displays the text 'Book Genre Clustering'. In the top toolbar, the 'Save' icon (a blue square with a white circle) is highlighted with a red box. Other icons include '+' (New), 'X' (Close), and various file operations like 'Open' and 'Copy'.

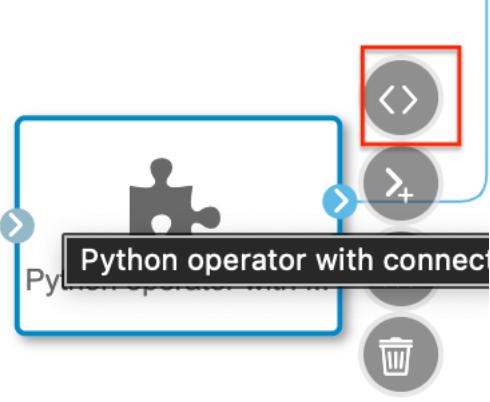
STEP 2 – BUILD MODEL PIPELINES

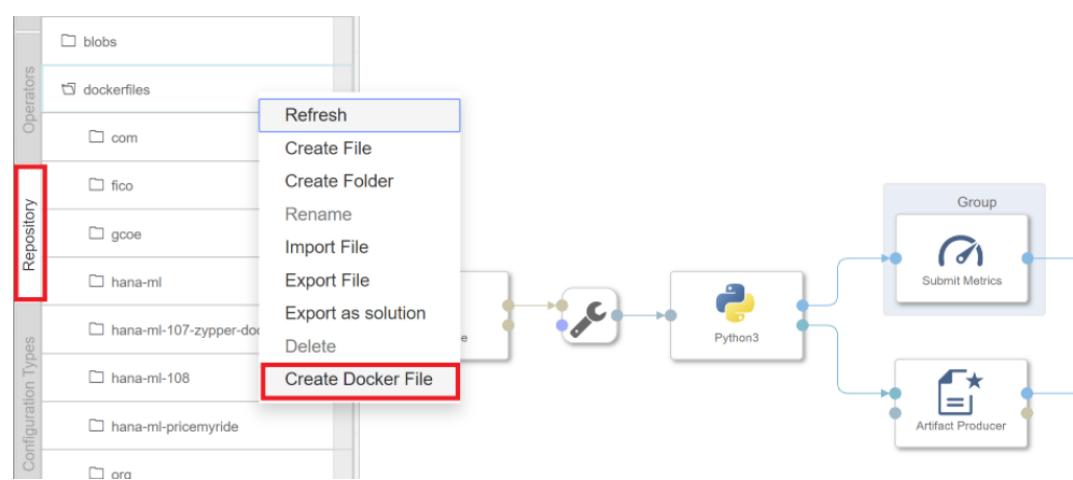
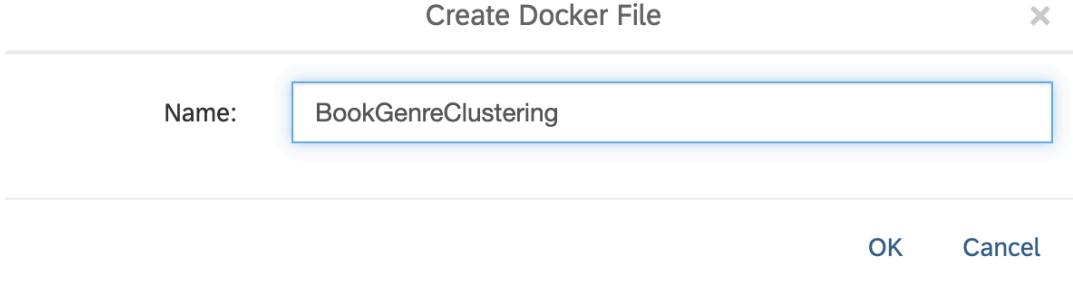
Model pipelines are used to operationalize the cluster model. Now that you have prepared the data, identified the best algorithm to use and which hyper-parameters work best, you want to take the model to production. You will build two pipelines. The first one is used to automate the model training, while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

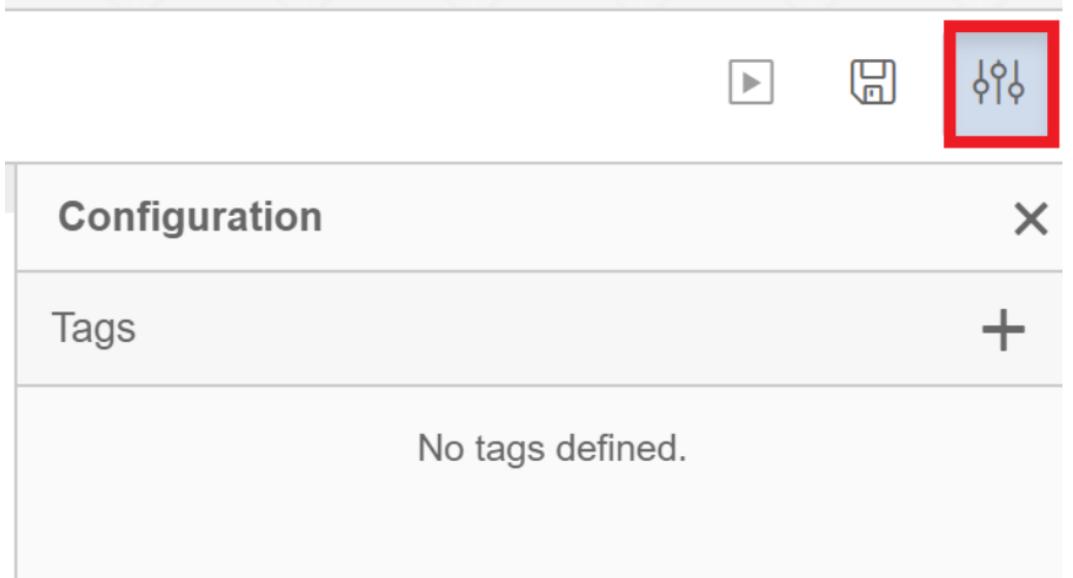
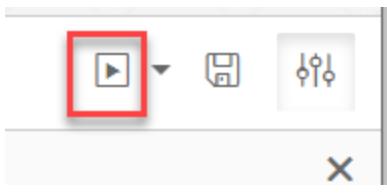
Explanation	Screenshot
	<p style="text-align: center;">CREATE A DI PIPELINE TO TRAIN THE ML MODEL</p>
<p>To create the graphical pipeline to train the model, go to your ML Scenario's main page, select the "Pipelines" tab and click Create.</p>	 <p>The screenshot shows the 'Book Text Cluster_01 Version 1' interface. At the top, there are tabs for Datasets (0), Notebooks (4), Pipelines (0) [highlighted with a red box], Executions (0), Models (0), and Deployments (0). Below the tabs, there is a table titled 'Pipelines (0)' with columns for Name, Description, and Template. A large red box highlights the 'Create' button in the top right corner of the table area.</p>
<p>Name the pipeline "Text Clustering Train" and select the "Python Producer" template. Click Create.</p>	 <p>The screenshot shows the 'Create Pipeline' dialog box. It has fields for 'Name:' (set to 'Book_Genre_Clustering_Train'), 'Description:' (empty), 'Template:' (set to 'Python Producer'), and 'Create' and 'Cancel' buttons at the bottom. A large red box highlights the 'Create' button.</p>
<p>Let's see what the template pipeline does.</p> <p>The pipeline loads data with the "Read File" operator. The data is passed to a Python operator, where the</p>	 <p>The screenshot shows the completed pipeline graph. The flow starts with a 'Read File' operator, followed by a 'Python3' operator. A large red box highlights this segment. The output of the Python3 operator connects to a 'Submit Metrics' operator, which then connects to an 'Artifact Producer' operator. The 'Artifact Producer' operator connects to a 'Write File' operator, which then connects to an 'Operators Complete' operator. Finally, the 'Operators Complete' operator connects to a 'Graph Terminator' operator. The pipeline is shown with various status icons and progress bars.</p>

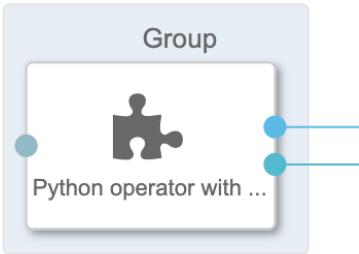
Explanation	Screenshot
<p>ML model is trained. The same Python-operator stores the model in the ML Scenario through the “Artifact Producer”. The Python-operator’s second output passes a model quality metric to the ML Scenario. Once both model and metric are saved, the pipeline’s execution is ended with the “Graph Terminator”.</p> <p>We need to adjust the pipeline template, in particular the part highlighted in red.</p>	
<p>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)</p> <p>Insert a “Python with HANA ML connection operator” in the canvas. Click on the “Add port” symbol to add two output ports.</p> <p>Configure the ports with the parameter shown in the picture,</p>	

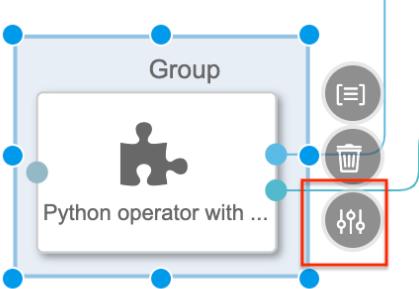
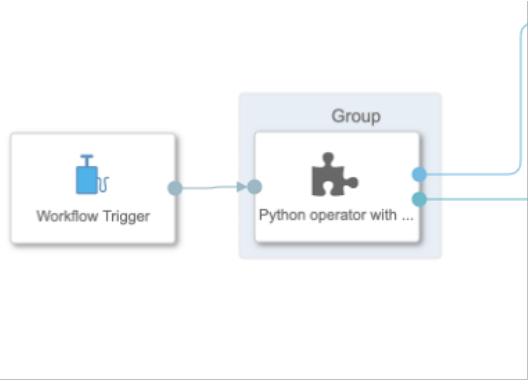
Explanation	Screenshot
<p>paying attention to the names and types. The names are used in the training Python code we will execute.</p>	
<p>In the Configuration panel, open the “Connection” configuration, set “Configuration type” to “Connection Management” and select from the drop-down menu the connection ID related to your Hana Cloud instance.</p> <p>You have to use your connection to DWC, named DWC_<USER_ID>.</p>	
<p>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.</p>	

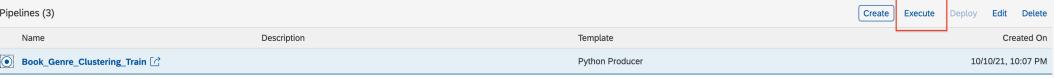
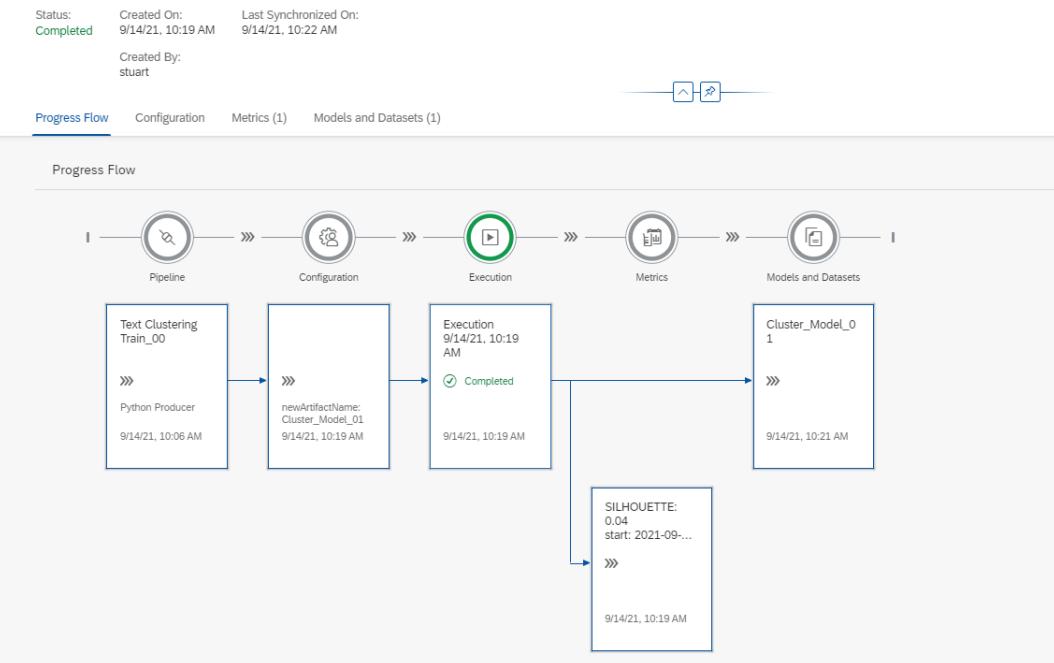
Explanation	Screenshot
<p>Please, don't touch the second and third Conversions operator! Please, notice also that the second and third Conversion operators are different: they are a ToMessage and a ToString converter, respectively.</p>	
<p>Next, adjust the Python code that analyzes the text data and trains the cluster model. Select the custom operator and click the Script option.</p>	
<p>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.</p> <p>In this code you have to customize the name of the table to use and the schema where the</p>	<p>https://github.com/SAP-samples/btp-data-to-value-workshop/blob/main/02-data-modeling%26processing/exercises/code_snippets/dv220_train.py</p>

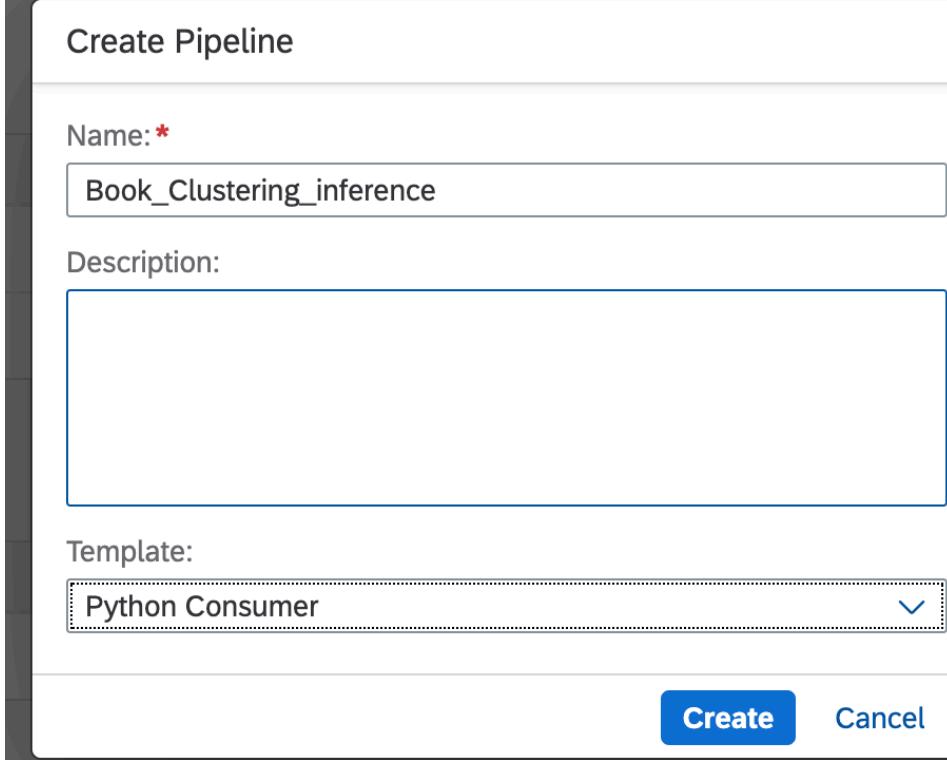
Explanation	Screenshot
<p>table resides. We will use again the Book_Author_Dimension_View prepared during exercise DV200_Exercise01 and that resides in DWC under the schema named like your <USER_ID>.</p>	
<p>Close the Script-window, then click “Save” in the menu bar.</p>	
<p>You need to create a Docker 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”.</p>	
<p>Name the file BookGenreClustering.</p>	

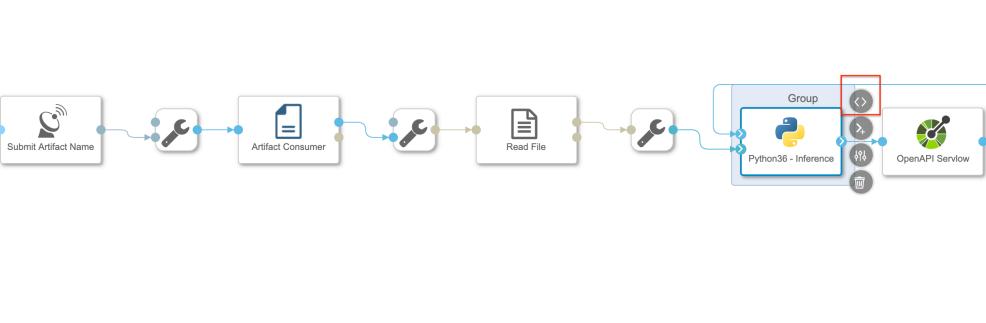
Explanation	Screenshot
<p>Enter this code into the Docker File window. This code leverages a base image that comes with SAP Data Intelligence and installs the necessary libraries on it.</p> <p>NB: Please, don't use the Docker file definition used for the book recommendation exercise.</p>	<pre>FROM \$com.sap.sles.base 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</pre>
<p>Open the Configuration panel for the Docker File with the icon on the top-right hand corner.</p>	 <p>The screenshot shows the SAP Data Intelligence interface with a 'Configuration' panel open. The panel title is 'Configuration'. Below it is a 'Tags' section with a '+' button. A message 'No tags defined.' is displayed. In the top right corner of the configuration panel, there is a red box highlighting the 'Configuration' button.</p>
<p>Assign a tag to the docker image.</p>	 <p>The screenshot shows the SAP Data Intelligence interface with a Dockerfile code editor and a configuration panel. The Dockerfile contains the following code:</p> <pre>1 FROM \$com.sap.sles.base 2 RUN pip3.6 install --user numpy==1.16.4 3 RUN pip3.6 install --user pandas==0.24.0 4 RUN pip3.6 install --user sklearn 5 RUN pip3.6 install --user hdfs 6 RUN pip3.6 install --user regex 7 RUN pip3.6 install --user nltk 8 RUN pip3.6 install --user gensim 9 10</pre> <p>To the right, a 'Configuration' panel is shown with a red box around the 'Tags' section. It lists 'BookGenreClustering' and 'latest' as tags.</p>
<p>Now save the Docker file and click the "Build" icon to start building the Docker image. Wait a few minutes and</p>	 <p>The screenshot shows the SAP Data Intelligence interface with build controls. The 'Build' button (highlighted with a red box) is the leftmost icon in the row, followed by 'Save' and 'Configuration'.</p>

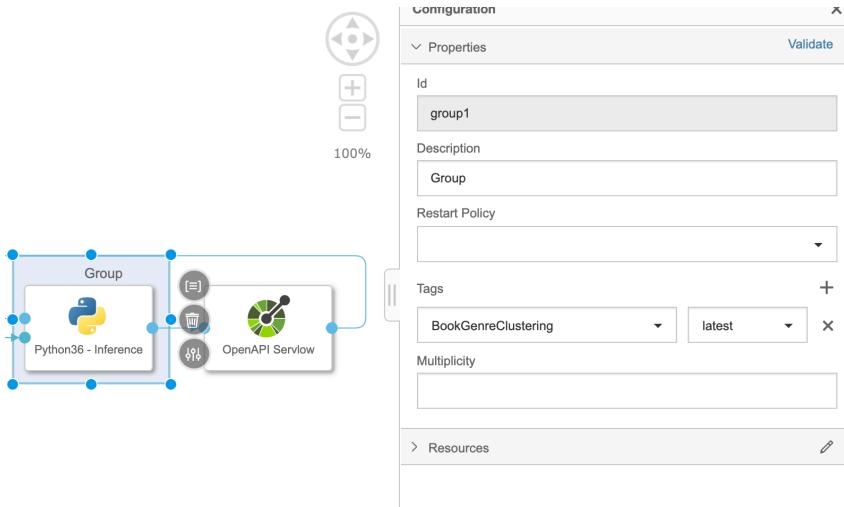
Explanation	Screenshot
you should receive a confirmation that the build completed successfully.	
Now you need to configure the custom operator, which trains the model, to use this Docker image. Go back to the graphical pipeline and right-click the custom operator and select "Group".	

Explanation	Screenshot
<p>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.</p>	 <div data-bbox="430 599 1491 1417"> <p>Configuration</p> <p>Properties Validate</p> <p>Id: group1</p> <p>Description: Group</p> <p>Restart Policy: Name: metricsResponse Data Type: message</p> <p>Tags: BookGenreClustering, latest</p> <p>Multiplicity</p> </div>
<p>Select the Workflow Trigger operator in the operators panel. Add it to the pipeline and connect it to the <u>trigger port of the custom operator</u>. Save your pipeline.</p> <p>Please, make sure your Python operator has an input port named</p>	

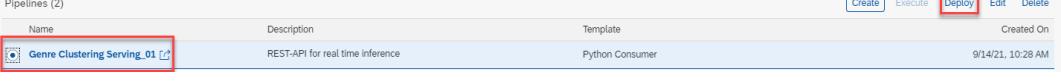
Explanation	Screenshot				
“trigger” of type “string”.					
The pipeline is now complete and you can run it. Go back to the ML Scenario. Select the pipeline in the ML Scenario and click the “Execute” button on the right.					
<p>Click “Step 3” and then “Step 4” to skip the optional steps until you get to the “Enter your Pipeline Parameters”. Set “newArtifactName” Value to “kmeans”. Click Save. The trained model will be saved under this name.</p>	<p>4. Pipeline Parameters</p> <p>Enter your Pipeline Parameters.</p> <table border="1"> <thead> <tr> <th data-bbox="463 783 577 804">Key</th> <th data-bbox="1318 783 1367 804">Value</th> </tr> </thead> <tbody> <tr> <td data-bbox="463 819 577 840">newArtifactName*</td> <td data-bbox="1328 819 1377 840">kmeans</td> </tr> </tbody> </table> <p>Save</p>	Key	Value	newArtifactName*	kmeans
Key	Value				
newArtifactName*	kmeans				
<p>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.</p> <p>NB: Any errors can be investigated in the pipeline editor.</p>					

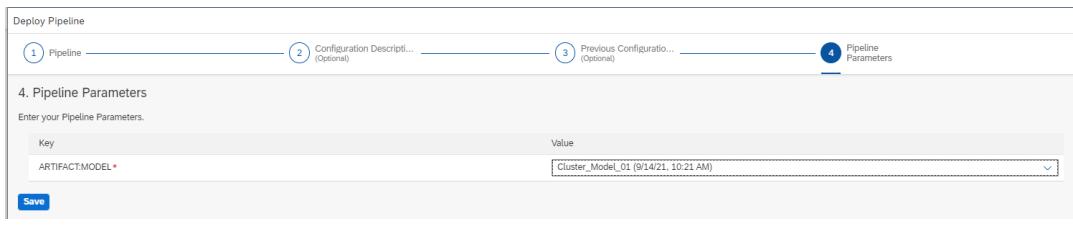
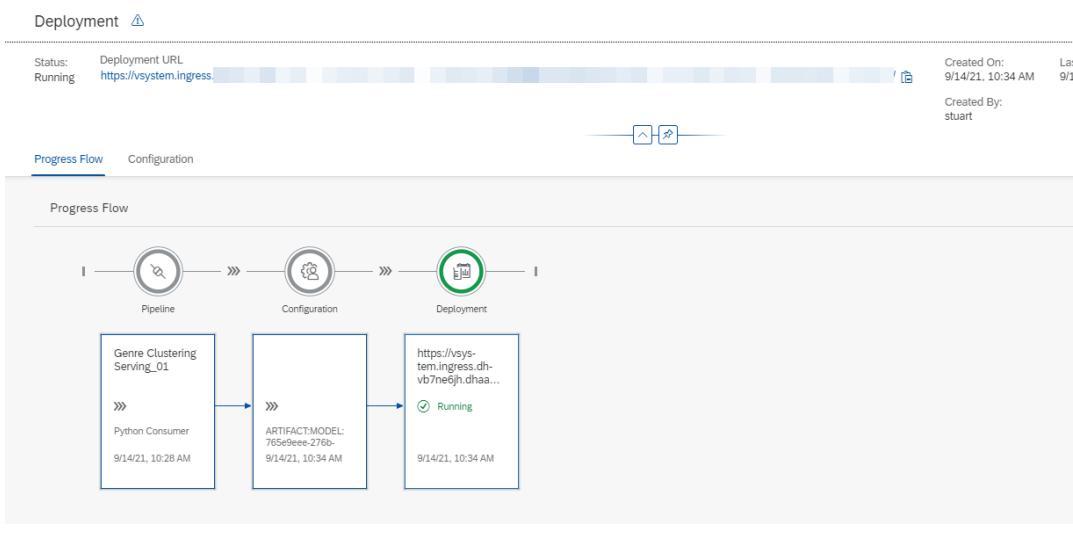
Explanation	Screenshot
	<p style="color: red; font-weight: bold;">CREATE A DI PIPELINE TO SERVE AND CONSUME THE ML MODEL</p>
<p>You will now use the model for real-time inference with REST-API. Go back to the main page of your ML Scenario and create a second pipeline. This pipeline will provide the REST-API 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.</p>	 <p>The screenshot shows the 'Create Pipeline' dialog box. It has fields for 'Name:' (containing 'Book_Clustering_inference'), 'Description:' (empty), and 'Template:' (set to 'Python Consumer'). At the bottom are 'Create' and 'Cancel' buttons.</p>
<p>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”</p>	 <p>The diagram illustrates the data flow of the pipeline. It starts with 'Submit Artifact Name', followed by 'Artifact Consumer', 'Read File', 'Python36 - Inference', and finally 'OpenAPI Servlow'. The components are connected by arrows, showing the sequence of operations.</p>

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	 <pre> graph LR A[Submit Artifact Name] --> B[Artifact Consumer] B --> C[Read File] C --> D[Python36 - Inference] D --> E[OpenAPI Servoy] style D fill:#0072bc,color:#fff </pre>
Carefully copy and paste to replace the whole code with the code given here.	https://github.com/SAP-samples/btp-data-to-value-workshop/blob/main/02-data-modeling%26processing/exercises/code_snippets/dv220_consumer.py
Close the editor window.	
Go back to the pipeline tab and save the pipeline.	

Explanation	Screenshot
<p>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.</p> <p>NB: Please, don't use the Docker file definition used for the book recommendation exercise, but the one created for the training named BookGenreClustering.</p>	 <p>The screenshot shows a configuration interface with a sidebar titled "Configuration". On the left, there's a diagram of a "Group" node containing two operators: "Python36 - Inference" and "OpenAPI Servlow". The "Python36 - Inference" operator has its "Tags" field set to "BookGenreClustering latest". The "OpenAPI Servlow" operator has its "Tags" field set to "latest". The main panel shows the configuration details for the "group1" group, including fields for "Id", "Description", "Restart Policy", "Tags", "Multiplicity", and a "Resources" section.</p>

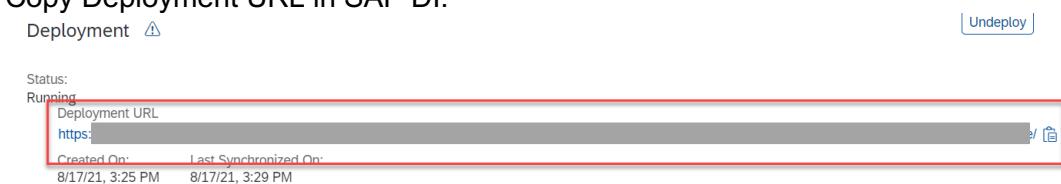
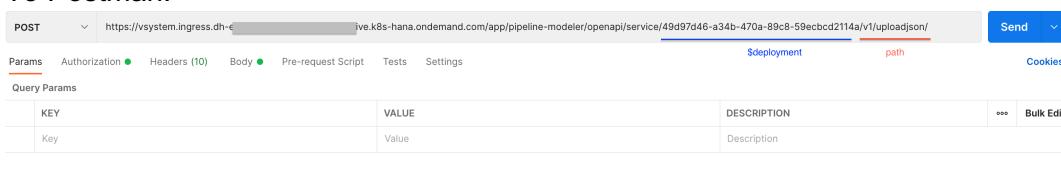
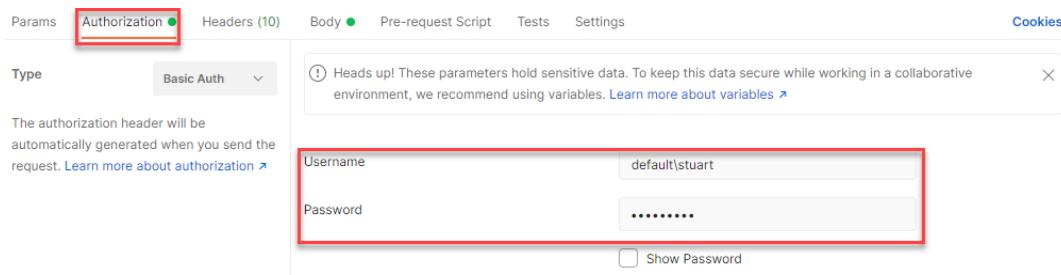
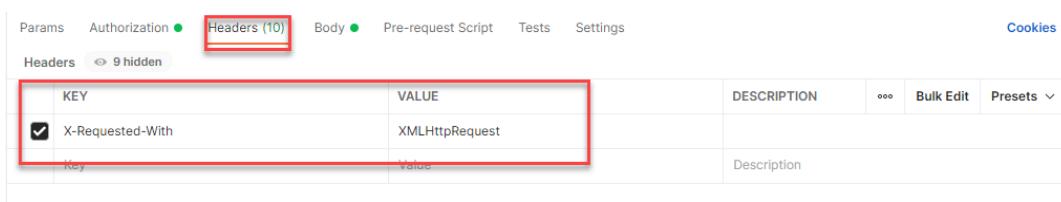
Explanation	Screenshot
<p>Click on OpenAPIServlow and have a look at the configuration.</p>	<p>The screenshot shows the 'Configuration' dialog for an 'OpenAPIServlow' service. The 'Properties' tab is selected. Key configuration settings include:</p> <ul style="list-style-type: none"> Id: openapiservlow1 Label: OpenAPI Servlow Base Path: \${deployment} Timeout: 300000 One-Way: False (radio button selected) Swagger Specification: A JSON object containing: <pre>{ "schemes": ["http", "https"] }</pre> Websocket: True (radio button selected) Max Concurrency: 32 Max Accepted: 128
<p>Notice in particular the content of the Swagger Specification, but don't change anything here!</p>	<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" } } }</pre>

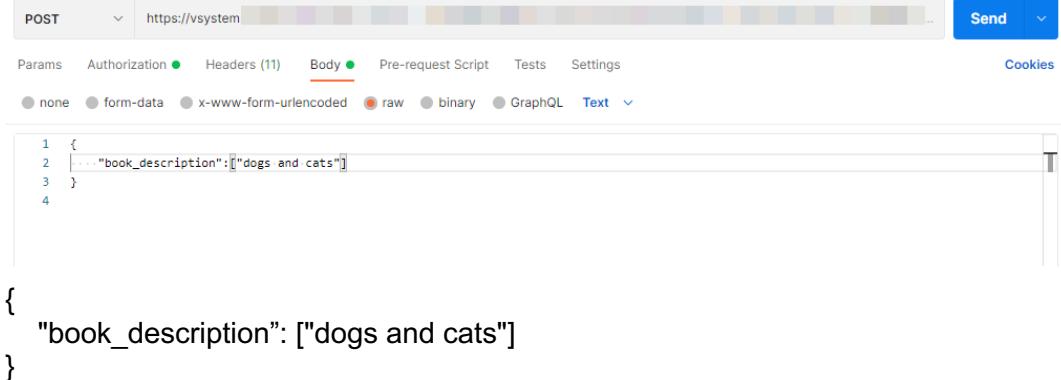
Explanation	Screenshot												
	<pre> "url":"http://www.apache.org/licenses/LICENSE-2.0.html" }, "version":"1.0.0" }, "basePath":"/\$deployment", "paths":{ "/v1/uploadjson":{ "post":{ "description":"Upload data in json format", "consumes":["application/json"], "produces":["application/json"], "summary":"Upload 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" } } } } </pre>												
<p>Go back to the ML Scenario. Now deploy the new pipeline. Select the pipeline and click “Deploy”.</p>	 <table border="1" data-bbox="437 1685 1498 1769"> <thead> <tr> <th colspan="4">Pipelines (2)</th> </tr> <tr> <th>Name</th> <th>Description</th> <th>Template</th> <th>Created On</th> </tr> </thead> <tbody> <tr> <td>Genre Clustering Serving_01</td> <td>REST-API for real time inference</td> <td>Python Consumer</td> <td>9/14/21, 10:28 AM</td> </tr> </tbody> </table>	Pipelines (2)				Name	Description	Template	Created On	Genre Clustering Serving_01	REST-API for real time inference	Python Consumer	9/14/21, 10:28 AM
Pipelines (2)													
Name	Description	Template	Created On										
Genre Clustering Serving_01	REST-API for real time inference	Python Consumer	9/14/21, 10:28 AM										

Explanation	Screenshot
<p>Click through the screens until you can select the trained model from the drop-down. Click "Save".</p>	 <p>The screenshot shows the 'Deploy Pipeline' interface. Step 4, 'Pipeline Parameters', is active. It displays a table with one row: 'Key' (ARTIFACT:MODEL) and 'Value' (Cluster_Model_01 (9/14/21, 10:21 AM)). A 'Save' button is at the bottom.</p>
<p>After a few minutes the pipeline will start running.</p> <p>NB: Any errors can be investigated in the pipeline editor.</p> <p>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!</p>	 <p>The screenshot shows the 'Deployment' progress flow. It includes a header with status 'Running', deployment URL 'https://vsystem.ingress...', and creation details ('Created On: 9/14/21, 10:34 AM' by 'stuart'). Below is a 'Progress Flow' diagram with three nodes: 'Pipeline' (Icon: gear), 'Configuration' (Icon: person), and 'Deployment' (Icon: server). Arrows show the flow from Pipeline to Configuration, and Configuration to Deployment. The 'Deployment' node is highlighted with a green circle and labeled 'Running'. The 'Configuration' node also has a green circle.</p>

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
<p>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”.</p>	<p>Copy Deployment URL in SAP DI:</p>  <p>To Postman:</p> 
<p>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.</p> <p>Please, go to the Teams channel General>System Access to check the name of your tenant.</p>	
<p>Go to the “Headers” tab and enter the key “X-Requested-With” with value “XMLHttpRequest”.</p>	

Explanation	Screenshot
<p>Finally, pass the input data to the REST-API. Select the “Body” tab, choose “raw” and enter the syntax given here.</p> <pre data-bbox="425 255 1486 635">{ "book_description": ["dogs and cats"] }</pre>	
<p>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.</p>	 <pre data-bbox="474 846 768 1015">1 { 2 "Cluster": [3 9 4] 5 }</pre>
<p>You have now completed the exercise.</p>	