Governed MLOps Workshop

Automation with Watson Pipelines

Document version: June 2023

DISCLAIMER

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."
Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

OpenShift is a trademark of Red Hat, Inc.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Table of Contents

Introduction

In previous modules, you have seen how to train AI models for churn prediction using AutoAI and Jupyter notebooks. You've also leveraged Data Refinery to prepare data that had been virtualized and cataloged from different sources into one dataset to drive the training of churn prediction models. In this module you will learn how to automate all the previous steps using Watson Pipelines.

Watson Pipelines help to manage the AI lifecycle by automating the various tasks involved in training machine learning models including data preparation, model exploration and training, and model deployment. Please review the following blogs to better understand use cases, scenarios and advantages of Watson Studio Pipelines:

- ⇨ Data Threads: Introducing IBM Watson Studio Pipelines Beta on Cloud Pak for Data 4.5
- ⇨ Automating the AI Lifecycle with IBM Watson Studio Pipelines

This lab consists of 2 parts, you can choose to do either one or both:

- Lab A – Build a pipeline step-by-step.
- Lab B – Import and run a pre-built pipeline.

Lab A – Defining automation with Watson Pipelines Step-by-step

Prerequisites and setup

For this lab, we assume that you have completed the previous modules including the following steps:

1. Virtualized and cataloged relevant data assets (CUSTOMER_TRANSACTION_DATA, CUSTOMER_PERSONAL_INFO, and CUSTOMER_CHURN).
2. Created and executed a Data Refinery job to join the 3 data assets referenced in step 1 and generate a CUSTOMER_DATA_ready data asset.
3. Run an AutoAI experiment to explore multiple algorithms and select the best performing AI model for predicting the likelihood of a customer to churn.
4. Executed a Jupyter notebook which leverages PySpark open-source libraries for training a Random Forest model for predicting the likelihood of a customer to churn.

For this module, you will also leverage two deployment spaces:

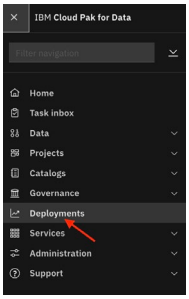
⇨ churn_prediction_dev_space: This deployment space will be used during the development, experimentation, and exploration of different AI models for predicting customer churn.

⇨ churn_prediction_preProd_space: This deployment space will be used for hosting pre-production (preProd) models that would be tested by the validation team.

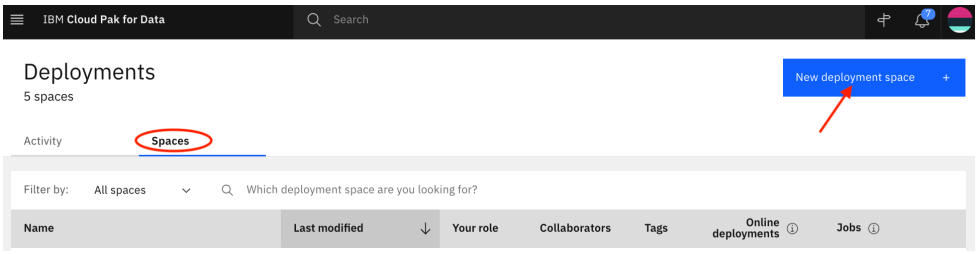
If you don't have these deployment spaces created already, please run through the following steps to create them.

1- Log into Cloud Pak for Data as datascientist user.

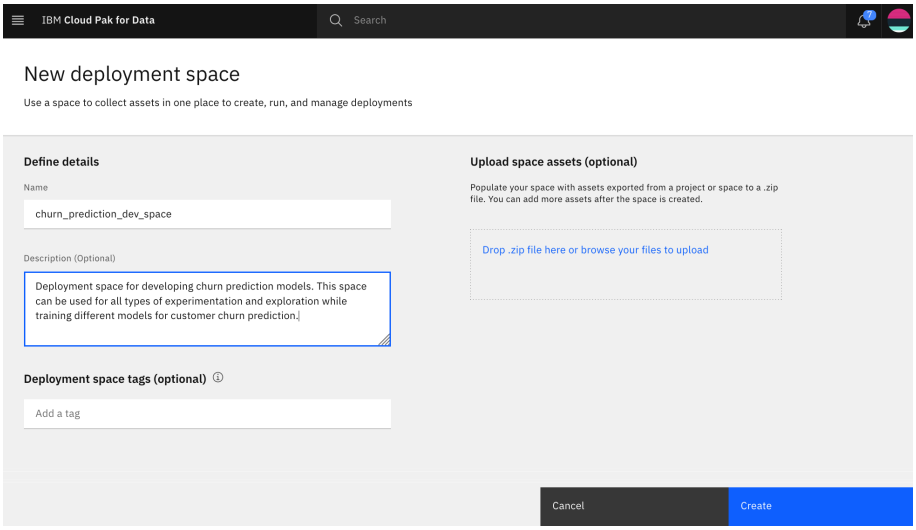
2- Navigate to Deployments by clicking on the Navigation menu (top left hamburger icon) and selecting Deployments (annotated with red arrow).



3- On the Deployments page, click the Spaces tab (annotated with red oval) and click New deployment space + (annotated with red arrow).



4- On the New deployment space page, provide a Name, churn_prediction_dev_space, and a Description (optional) and click Create.



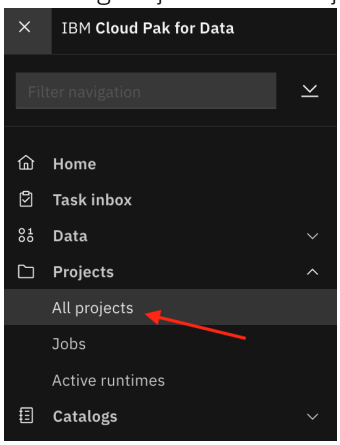
It takes a minute or so to get the new deployment space created. After deployment space is created, click Close.

5- Repeat the process for creating churn_prediction_preProd_space.

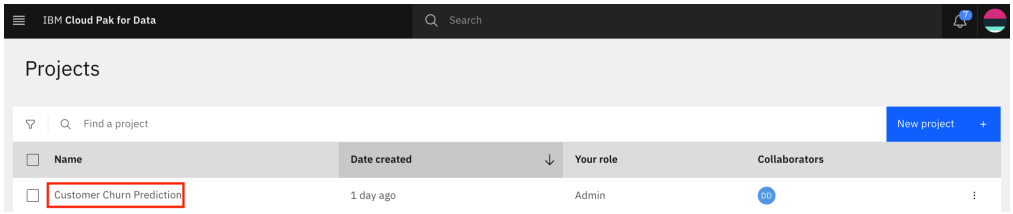
Build the pipeline manually

1- Log into Cloud Pak for Data as datascientist user.

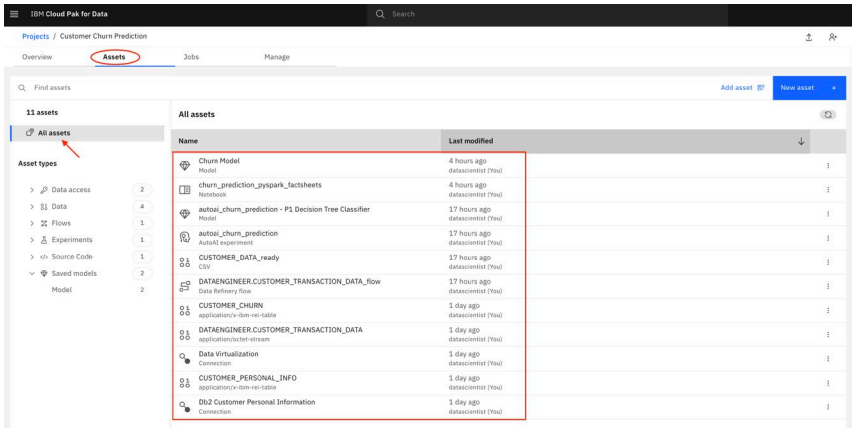
2- Navigate to your Projects view by clicking on the Navigation menu (top left hamburger icon) and selecting Projects ➔ All Projects (annotated with red arrow).



3- Select the Customer Churn Prediction project (annotated with red rectangle) which was created in earlier modules.

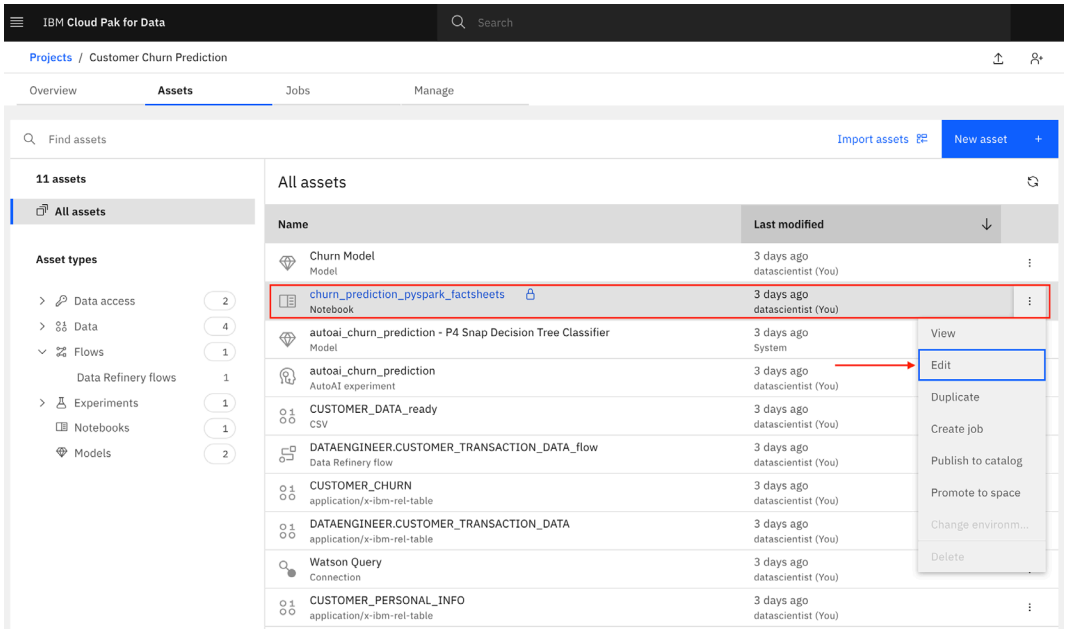


4- Select the Assets tab (annotated with red oval) and select All assets (annotated with red oval) to verify you can see all the following assets in your Customer Churn Prediction project including the data assets, the data refinery flow, the AutoAI experiment and the notebook (annotated with red rectangle). In subsequent steps, you will need to leverage some of these assets.

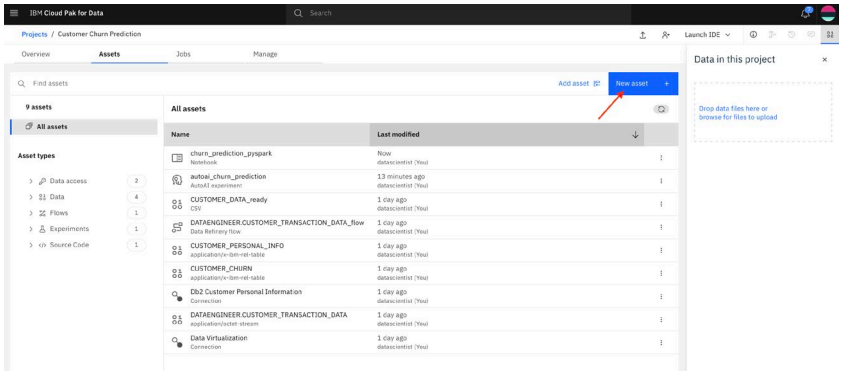


5-

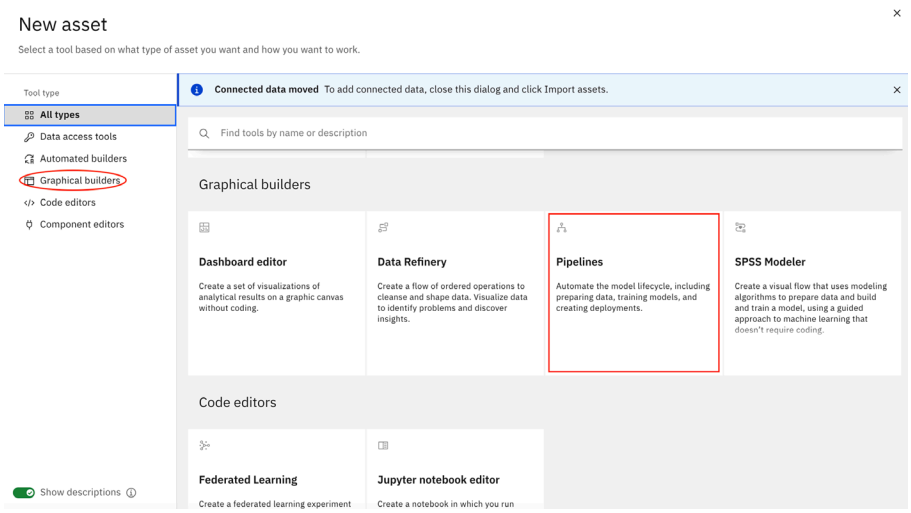
6- Find the Jupyter notebook (churn_prediction_pysaprk_factsheets) you used to train the churn prediction model (annotated with red rectangle) and click the open and close list of actions (3 vertical dots to the right of the asset) and select Edit (annotated with red arrow).



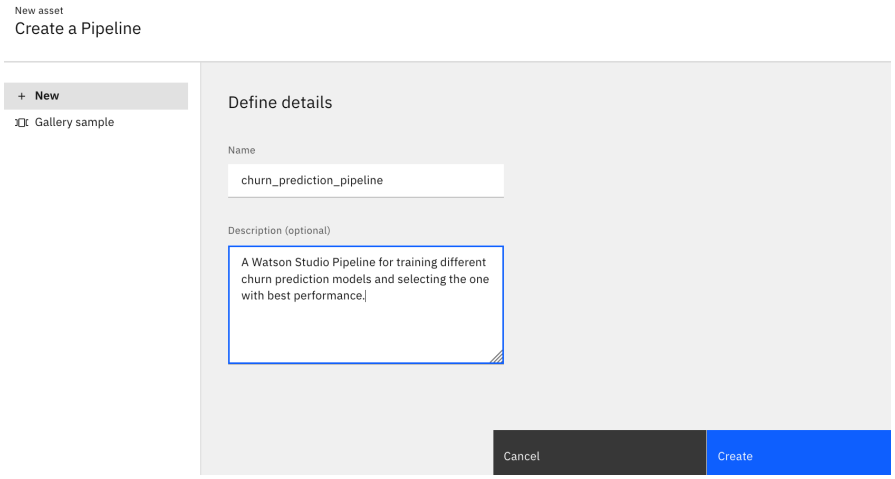
7- Click New asset + (annotated with red arrow) to add a new asset to your project.



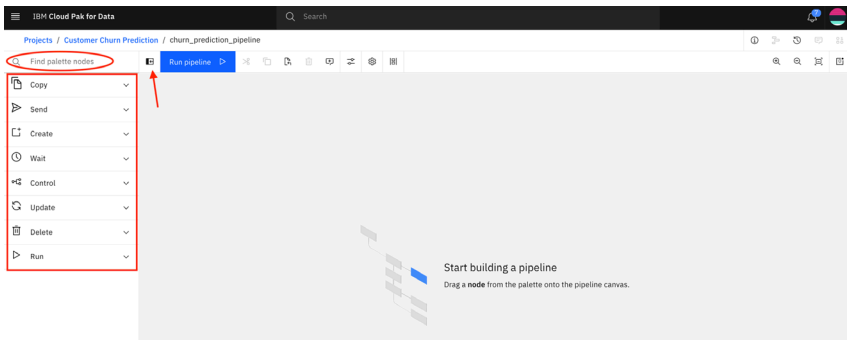
8- Scroll down and click the Pipelines tile (annotated with red rectangle). Note that you can also filter the Tool type to Graphical builders (annotated with red oval) to quickly find the Pipelines tile.



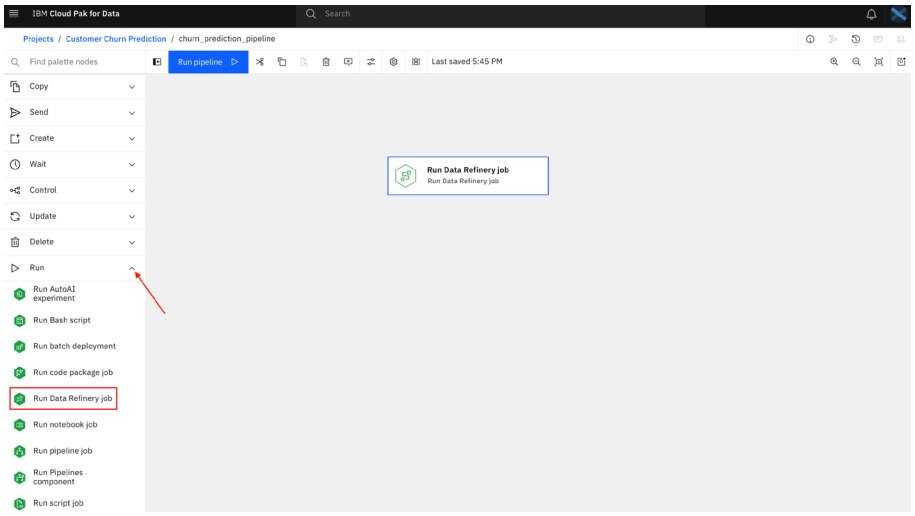
9- Provide a name and description (optional) for the pipeline and click Create.



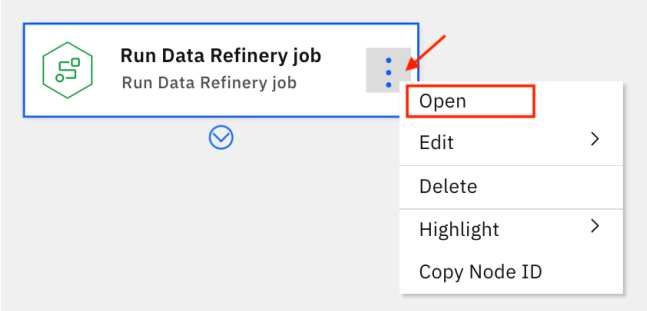
10- Next, you'll be creating a pipeline by dragging and dropping nodes (stages) from the palette onto the canvas. Take a minute to explore the various nodes in the palette (annotated with red rectangle). You can also search for a specific node by typing the name in the Find palette nodes search field (annotated with red oval). You can always hide/show the palette by clicking the Palette icon (annotated with red arrow).



11- Add a Run Data Refinery job node to the canvas by expanding the Run section (annotated with red arrow) and dragging/dropping the Run Data Refinery flow node (annotated with red rectangle).



12- Once on the canvas, open the Run Data Refinery job node either by double clicking it or by clicking the actions menu (3 vertical dots, annotated with red arrow) and clicking Open (annotated with red rectangle).



13- On the details view of the Run Data Refinery job node, edit the title of the node (annotated with red oval) to make it relevant to the task performed; for example, Prepare Customer Data using Refinery job. Also, click the selection method (annotated with red arrow) to review the different options for selecting the flow:

a. Select resource: You can select the resource explicitly from a project or a space that you have access to.

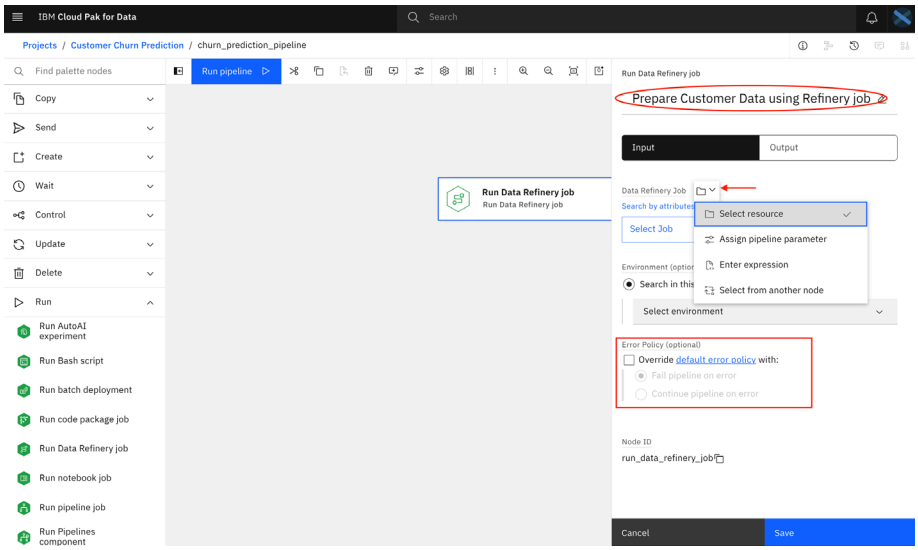
b. Assign pipeline parameter: You can use a pipeline parameter to reference the flow. You can define the pipeline parameter and modify it for different runs.

c. Enter expression: You can define an expression for this node. This would not be very relevant for this specific node but these selection methods apply to all nodes in the pipeline.

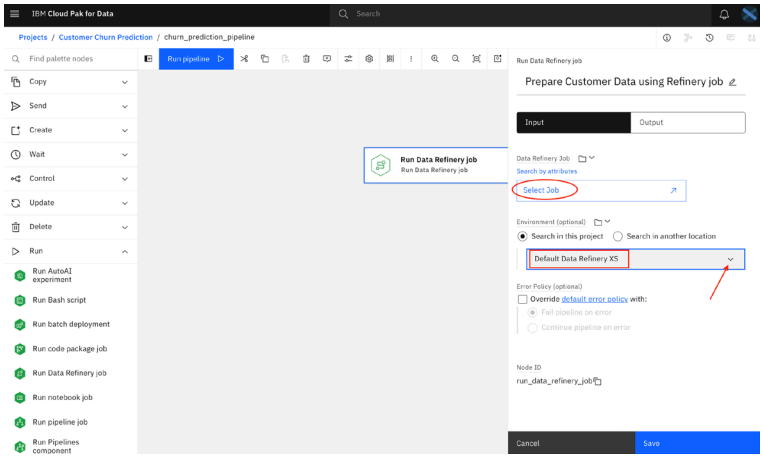
d. Select from another node: You can select this refinery flow as output from another node in the pipeline.

For this step, keep the default selection method of Select resource.

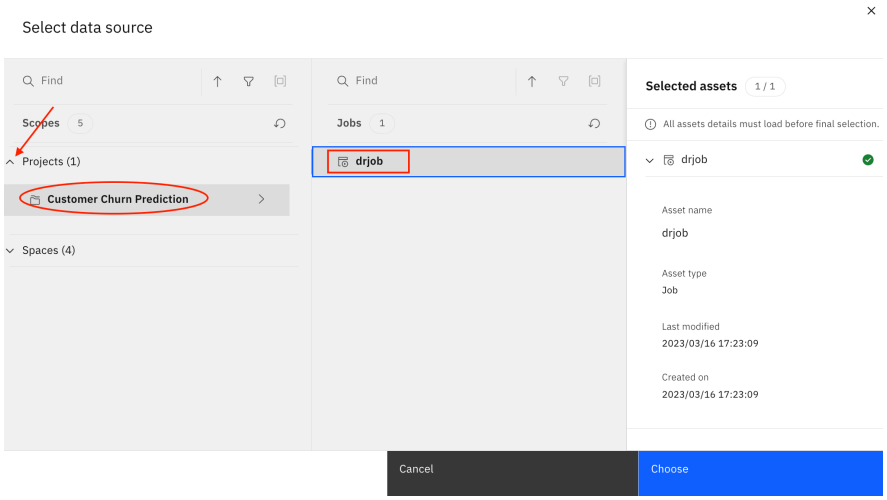
Also, please note the Error Policy (annotated with red rectangle) which effectively indicates that the pipeline would fail if this node had an error. You can modify the error policy if you'd like but for this step, keep the default behavior.



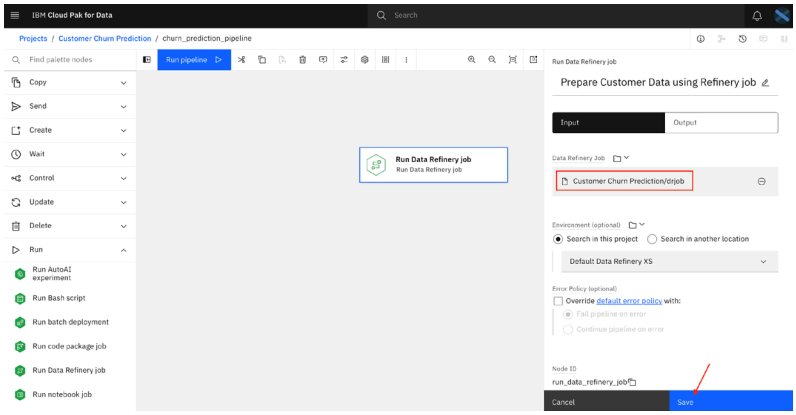
14- Next, click the Environment drop down (annotated with red arrow) and select the Default Data Refinery XS environment (annotated with red rectangle). Then, click the Select Job (annotated with red oval) to select the data refinery job to be executed by this node.



15- On the select job pop-up, expand the Projects list (annotated with red arrow), click the Customer Churn Prediction (annotated with red oval) and select the drjob job (annotated with red rectangle). Click Choose to select that data refinery job. Remember this was the job we created to run the Data Refinery flow which joins the 3 customer tables into one table, called CUSTOMER_DATA_ready.

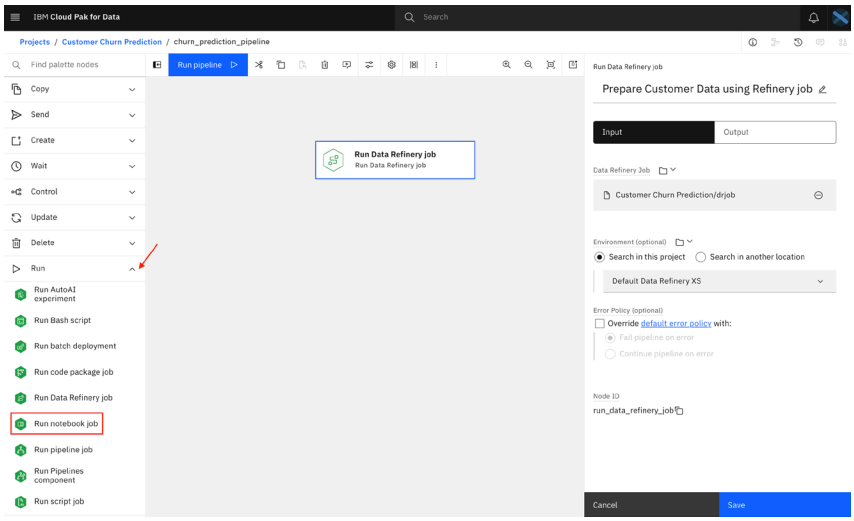


16- Back on the pipeline canvas page, note that the selected job is now associated with this node (annotated with red rectangle) and click Save (annotated with red arrow).

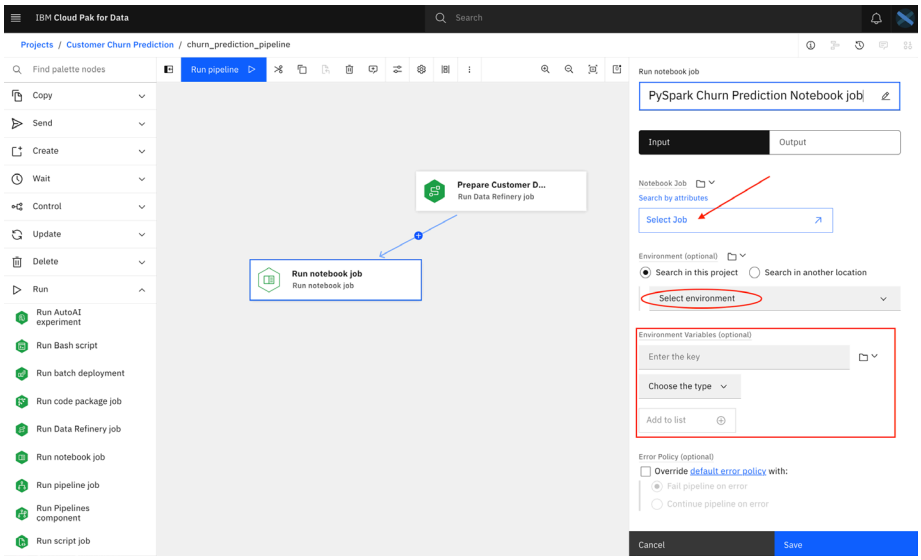


At this point, you have a simple 1-node pipeline which runs the referenced data refinery job. If you'd like, you can click Run pipeline button and select Trial Run to try out the pipeline. For now, hold off running the pipeline until we add some more nodes/stages.

17- Next, you will add a node to run a Jupyter notebook for training an AI model for customer churn prediction using PySpark. To do so, add a Run notebook job node by expanding the Run section in the palette (annotated with red arrow) and dragging/dropping the Run notebook job node (annotated with red rectangle) to the canvas.



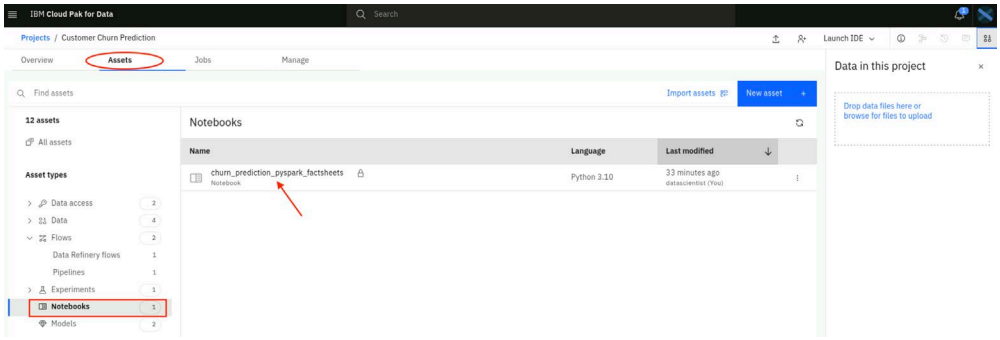
18- Connect the Run Data Refinery job node to the Run notebook job node by selecting the Run Data Refinery job node, clicking the connection arrow and dragging to the Run notebook job node. Then double click the Run notebook job node to open its detailed view. Notice that the required fields include the Notebook job to run (annotated with red arrow) and the Environment to use for executing that notebook (annotated with red oval). Also note the optional Environment variables (annotated with red rectangle) that can be passed to this node. In the example of Run notebook job node, environment variables are very important as they are the mechanism to communicate information between the pipeline and the notebook.



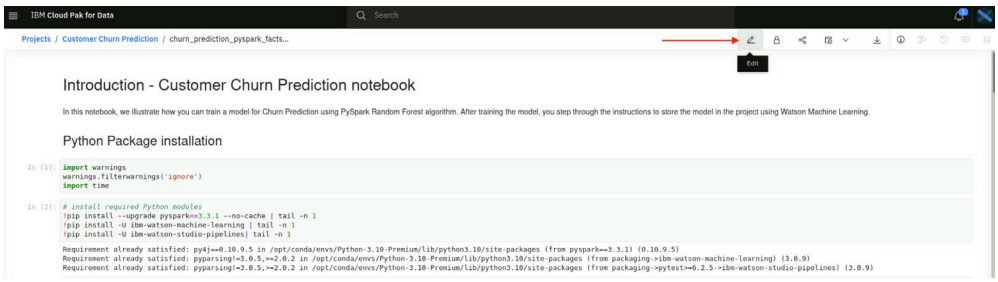
19- Before proceeding, you need to create a job of the customer churn prediction notebook you had developed in the previous. To do so:

   a. Navigate to your Customer Churn Prediction project (you can right click the project name in the breadcrumb and open in new tab).

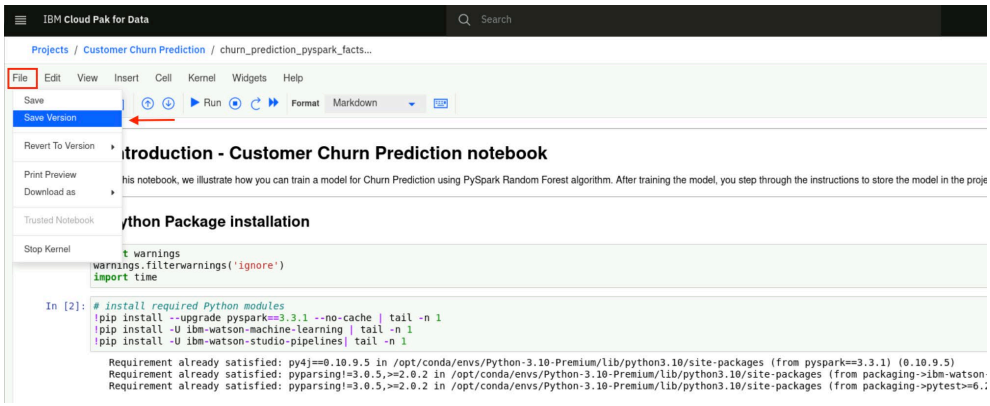
b. Click the Assets tab (annotated with red oval), select Notebooks (annotated with red rectangle) and click the churn_prediction_pyspark_factsheets notebook (annotated with red arrow).



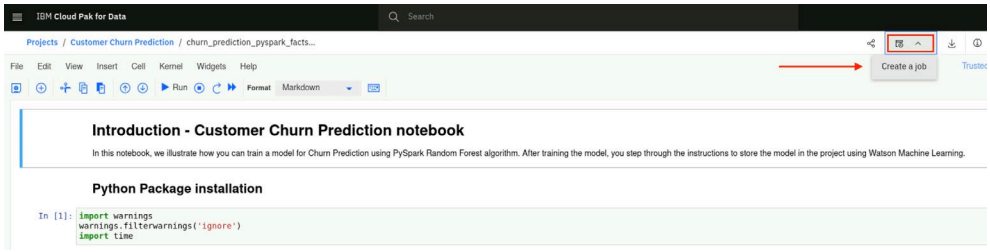
c. Click the pencil icon (annotated with red arrow) to load the notebook in Edit mode.



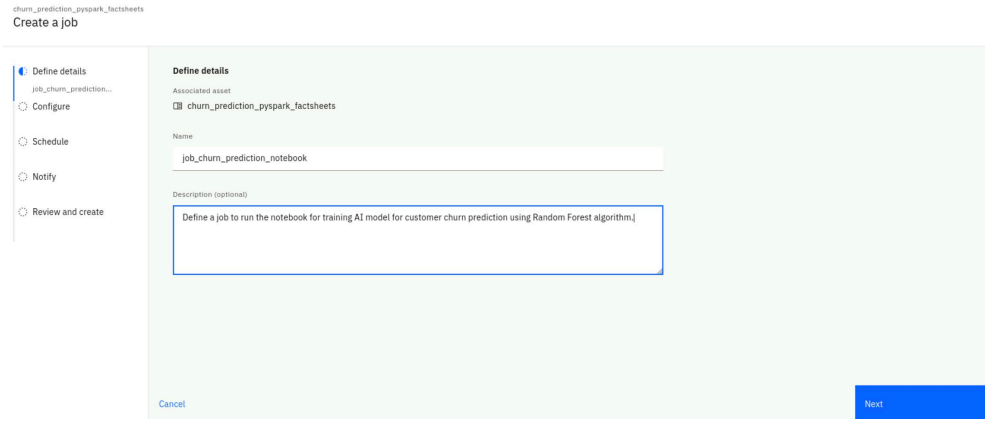
d. Once the notebook loads in edit mode, click File menu (annotated with red rectangle) and select Save Version (annotated with red arrow).



e. Next, click the jobs icon (annotated with red rectangle) and select Create a job (annotated with red arrow).

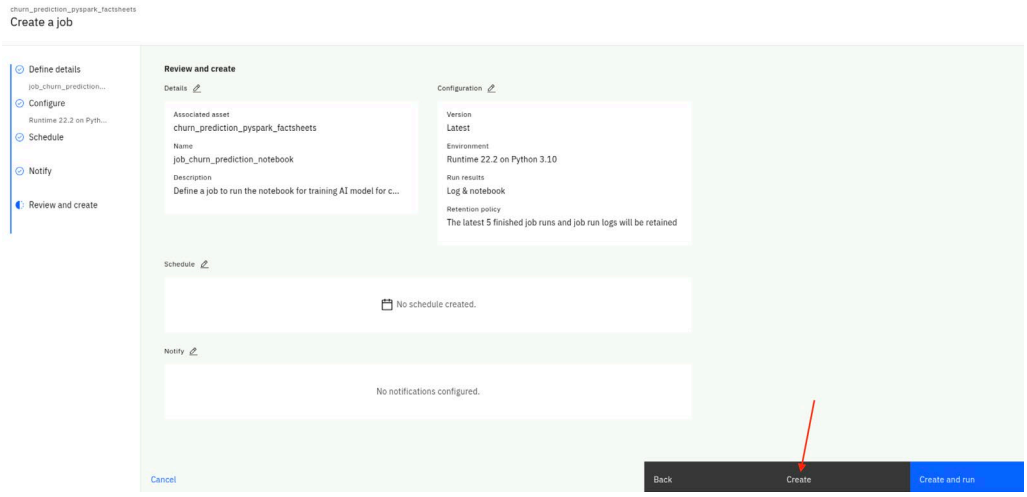


f. Provide a name and description for the job and click Next.

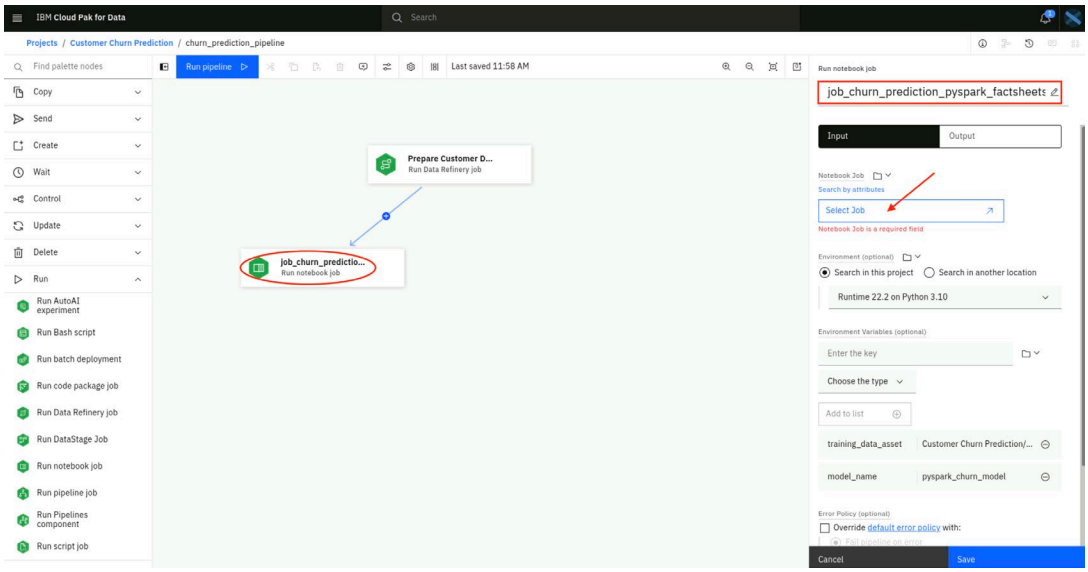


g. For Configure, Schedule, and Notify, accept the defaults (latest version of the notebook, default runtime, no scheduling and no notification) and click Next for each step.

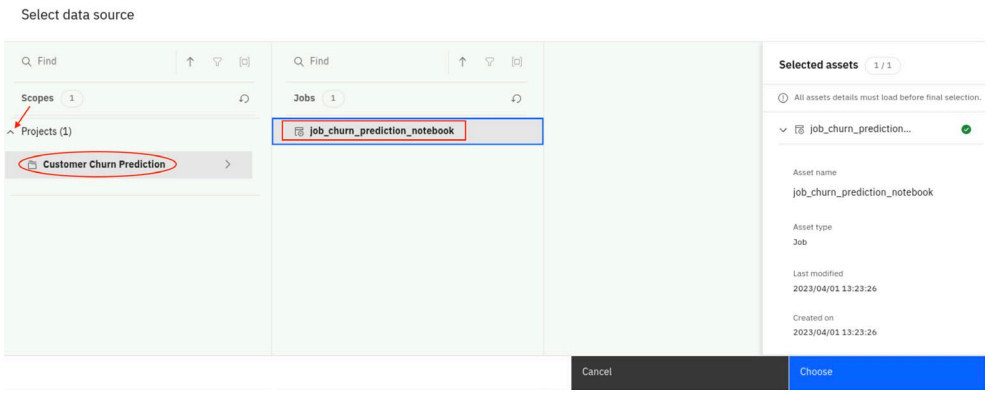
h. On the Review step, review the details of the job and click Create (annotated with red arrow). Don't select Create and run as you don't need to run the job at this time.



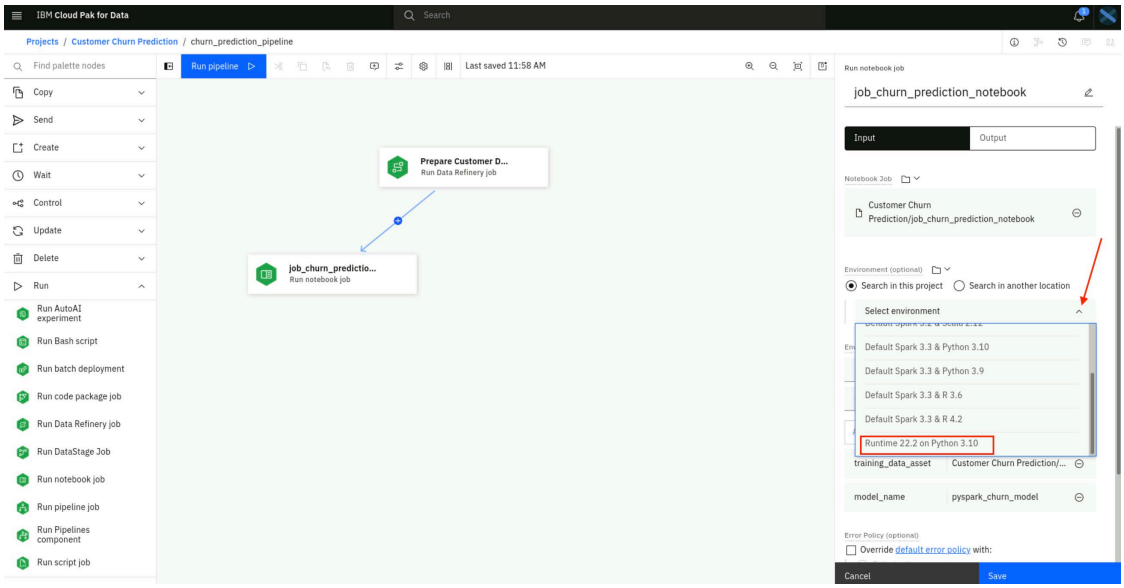
20- Back on the Pipeline browser tab (if it is closed, you can navigate back to your Customer Churn Prediction project, select Assets tab, and then open the Flows ➔ Pipeline and select the churn_prediction_pipeline), double click the Run notebook job node (annotated with red oval). Change the title of the node to job_churn_prediction_pyspark_factsheets (annotated with red rectangle). Then click the Select Job button (annotated with red arrow). to specify which notebook to execute.



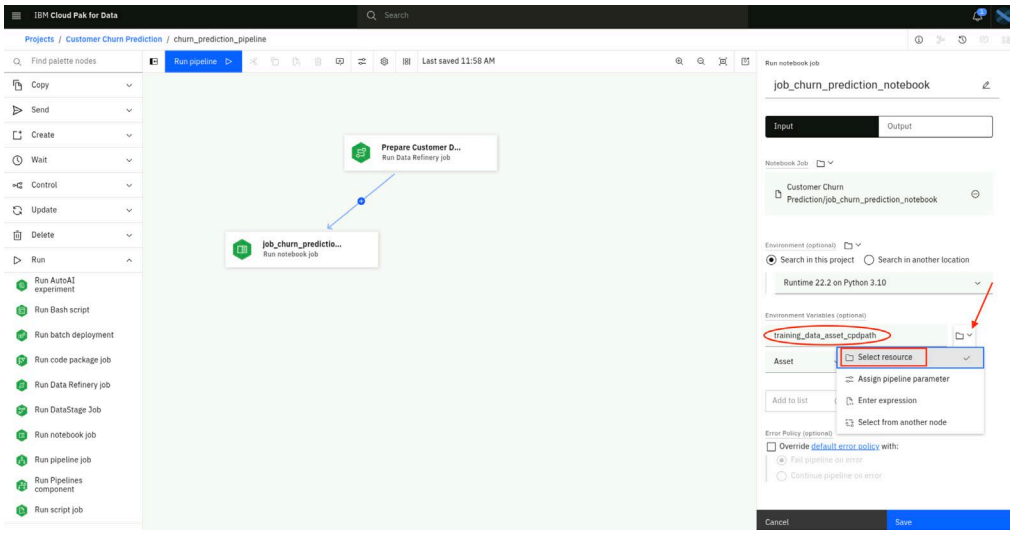
21- On the selection page, click the Projects drop down (annotated with red arrow), click the Customer Churn Prediction project (annotated with red oval) and select the job_churn_prediction_notebook notebook job (annotated with red rectangle). Click Choose.



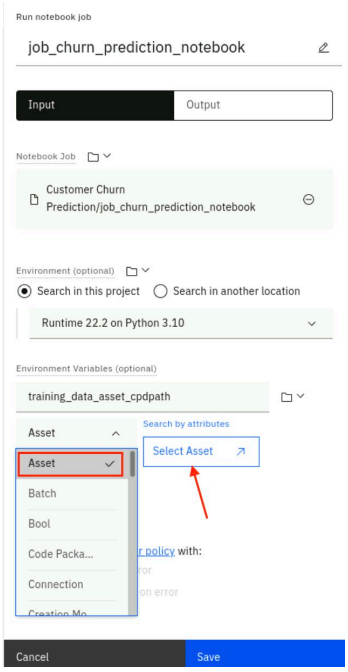
22- Next, select the environment to execute the notebook by clicking the Environment drop down (annotated with red arrow) and selecting the Runtime 22.2 on Python 3.10 (annotated with red rectangle). This is the same runtime we used for executing the churn prediction notebook earlier.



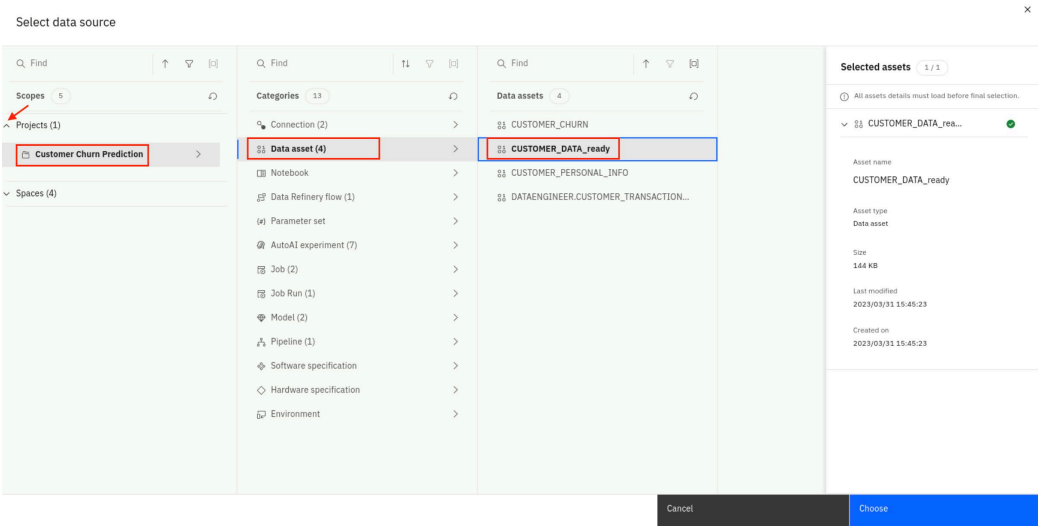
23- As mentioned earlier, the notebook expects certain environment variables to be provided from the pipeline, specifically the training data asset cpdpath and name as well as the name of the generated model. Under the Environment variables section, type the variable name training_data_asset_cpdpath (annotated with red oval) and click the selection method drop down (annotated with red arrow) and click Select resource (annotated with red rectangle).



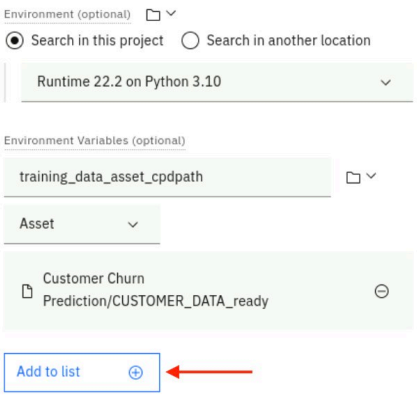
24- For the environment variable type, choose Asset (annotated with red rectangle) and then click Select Asset (annotated with red arrow).



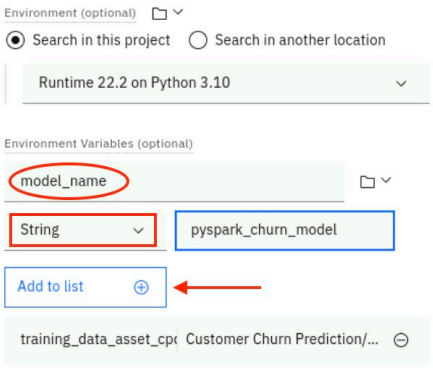
25- Navigate to the CUSTOMER_DATA_ready asset by clicking the Projects drop-down (annotated with red arrow) and selecting the project (Customer Churn Prediction), data asset (4), and CUSTOMER_DATA_ready (annotated with red rectangles). Click Choose.



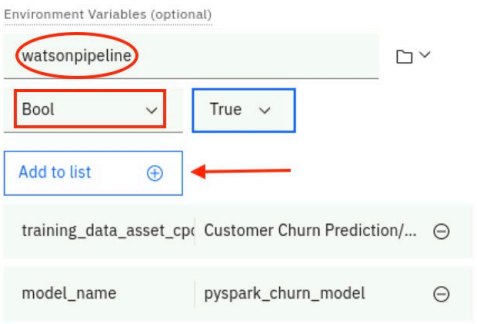
26- After selecting the CUSTOMER_DATA_ready asset, click Add to list (annotated with red arrow) to add this environment to the list of environment variables that will be passed to the job.



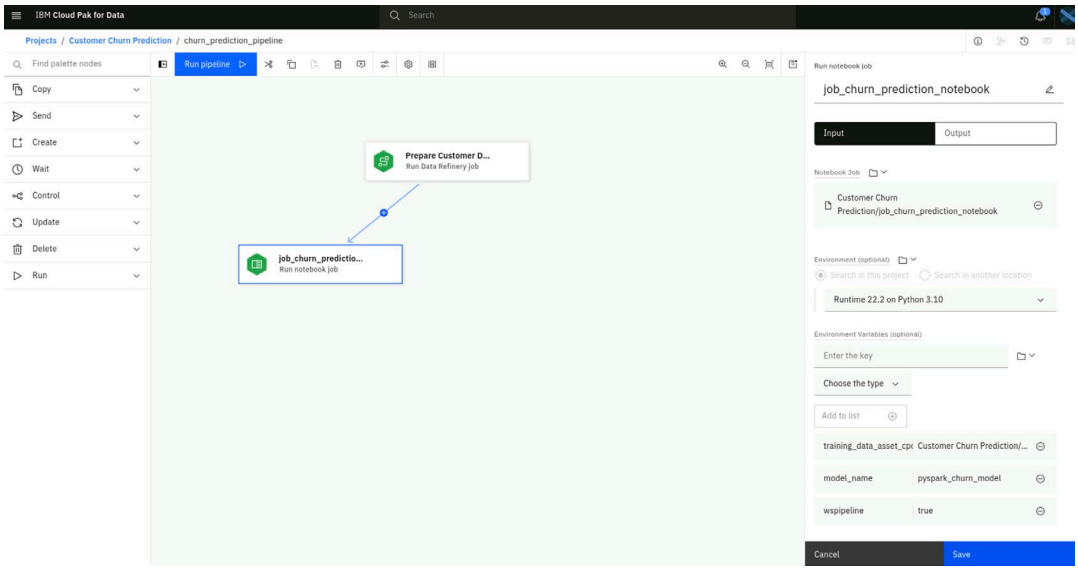
27- Define another environment variable, model_name (annotated with red oval), and for selection method, choose Select resource, then choose String for type (annotated with red rectangle) and provide pyspark_churn_model for the value of model_name variable. Then click Add to list (annotated with red arrow).



28- Define another environment variable, watsonpipeline (annotated with red oval), and for selection method, choose Select resource, then choose Bool for type (annotated with red rectangle) and select True for the value of wspipeline variable. Then click Add to list (annotated with red arrow).

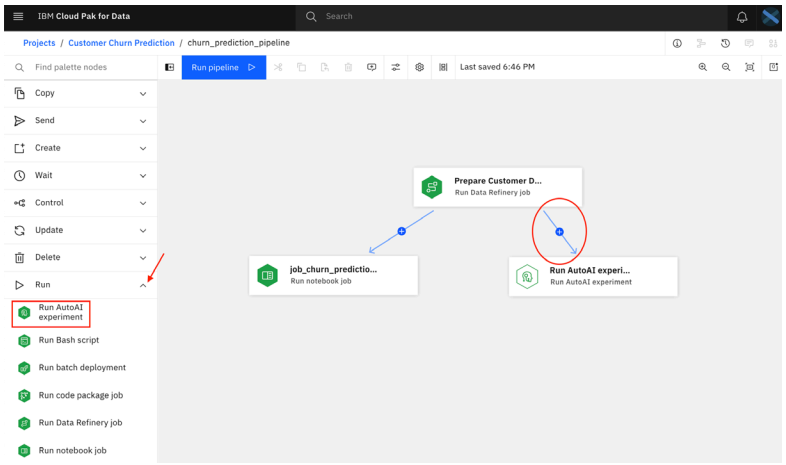


29- Review the details of the Run notebook job node and click Save.

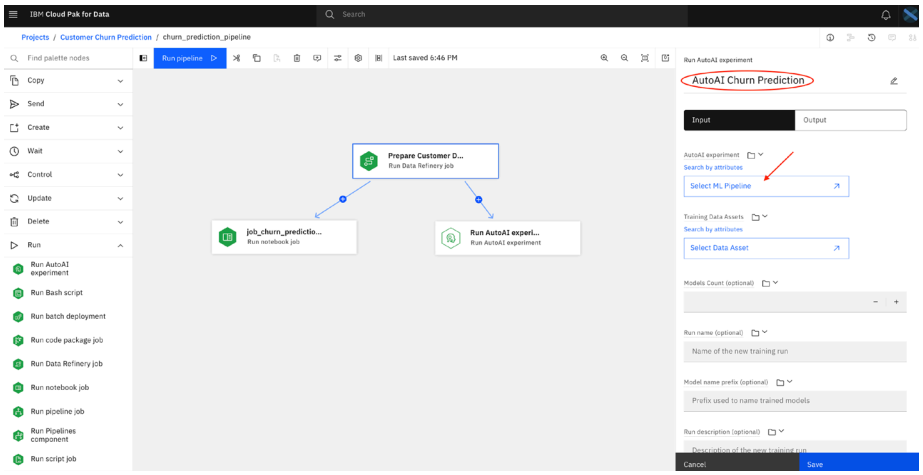


30- Next, add another node for running an AutoAI experiment. To do so, expand the Run section in the palette (annotated with red arrow) and drag/drop the Run AutoAI experiment (annotated with red rectangle) onto the canvas.

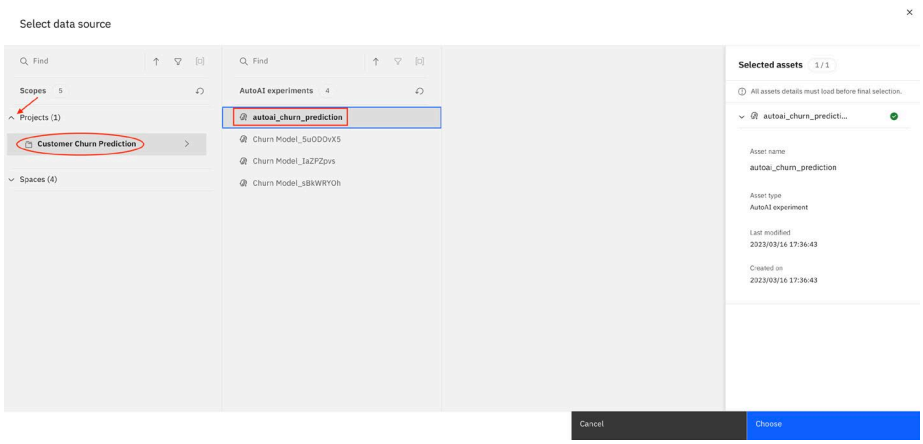
Drag and drop a connection from Run Data Refinery job node to the Run AutoAI experiment node (annotated with red oval).



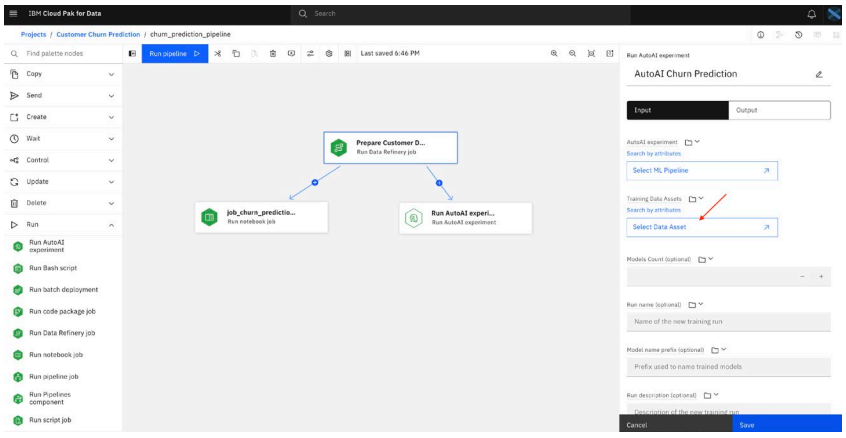
31- Double click the Run AutoAI experiment node to open the detailed view. Change the name of the node to AutoAI Churn Prediction (annotated with red oval). Click the Select ML Pipeline button (annotated with red arrow) to select the AutoAI experiment to execute.



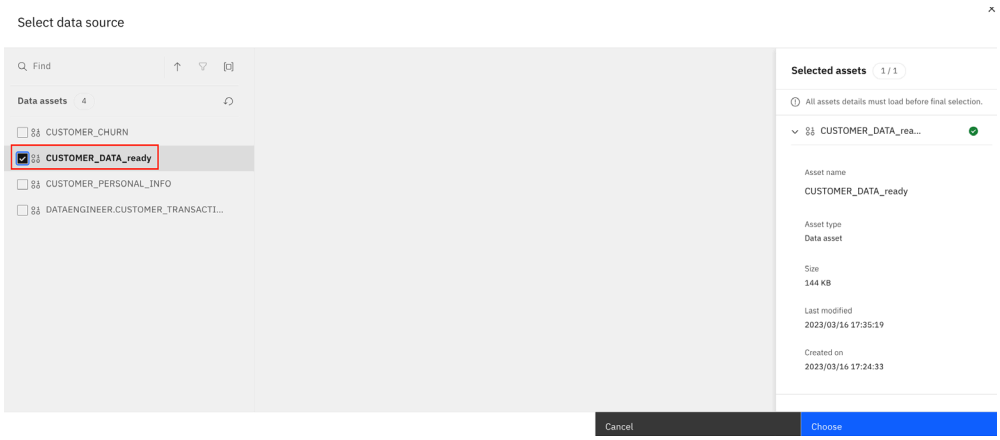
32- On the experiment selection page, click the Projects drop down (annotated with red arrow), click the Customer Churn Prediction project (annotated with red oval) and select autoai_churn_prediction AutoAI experiment (annotated with red rectangle). Click Choose.



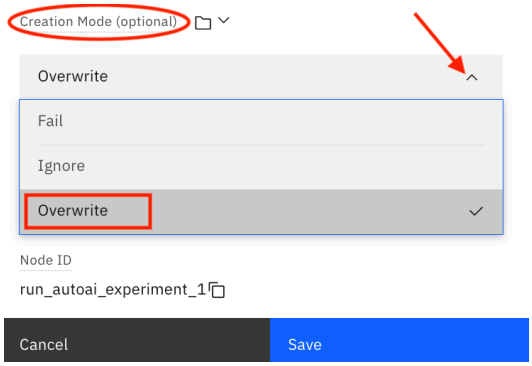
33- For Training Data Assets, click Select Data Asset button (annotated with red arrow)



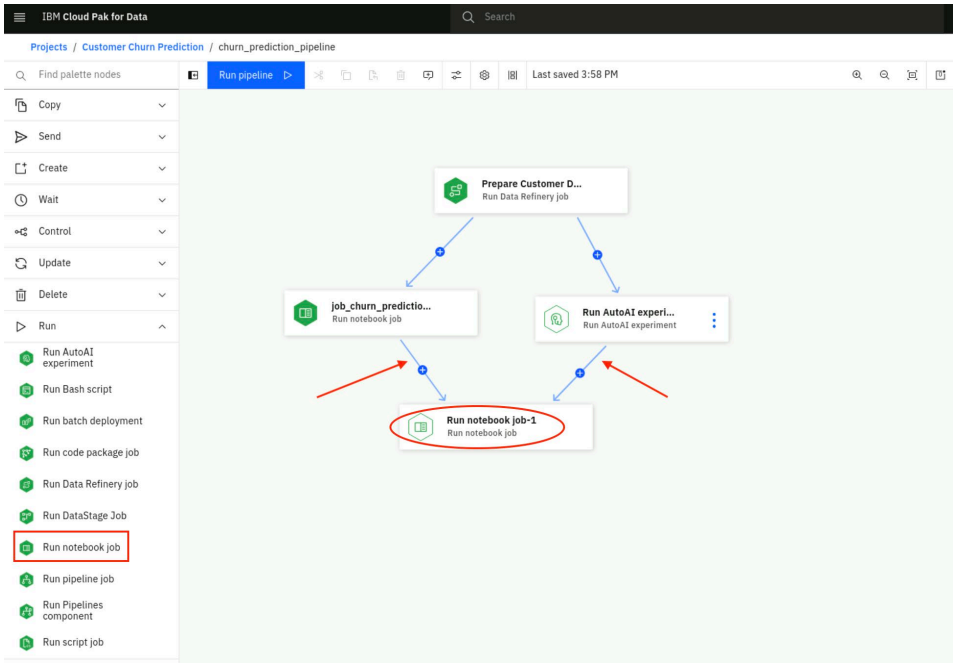
34- On the Select data source window, click the checkbox next to CUSTOMER_DATA_ready (annotated with red rectangle) and click Choose. Remember the CUSTOMER_DATA_ready is the data asset generated by the Data Refinery job.



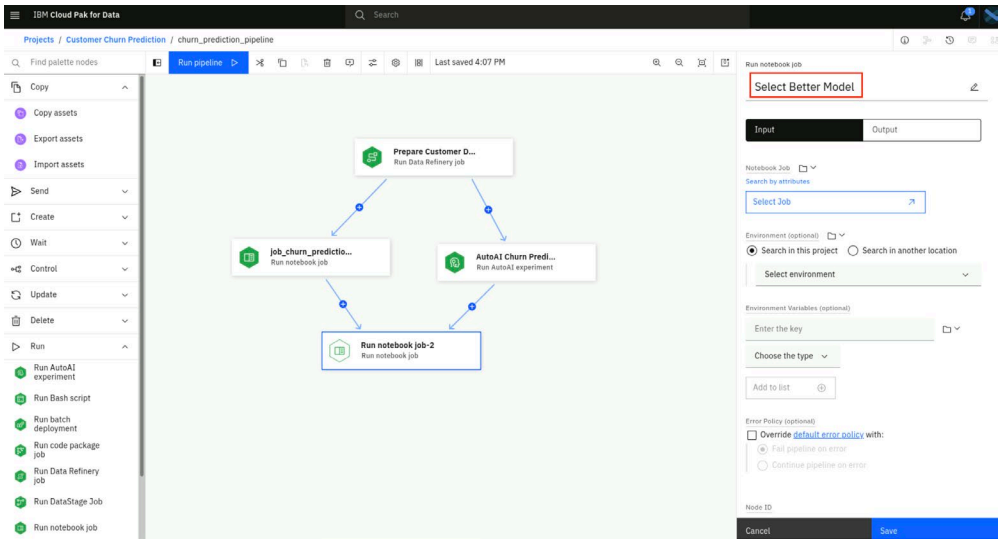
35- Scroll down to the Creation mode (annotated with red oval), click the drop-down selection (annotated with red arrow) and select Overwrite (annotated with red rectangle). Click Save.



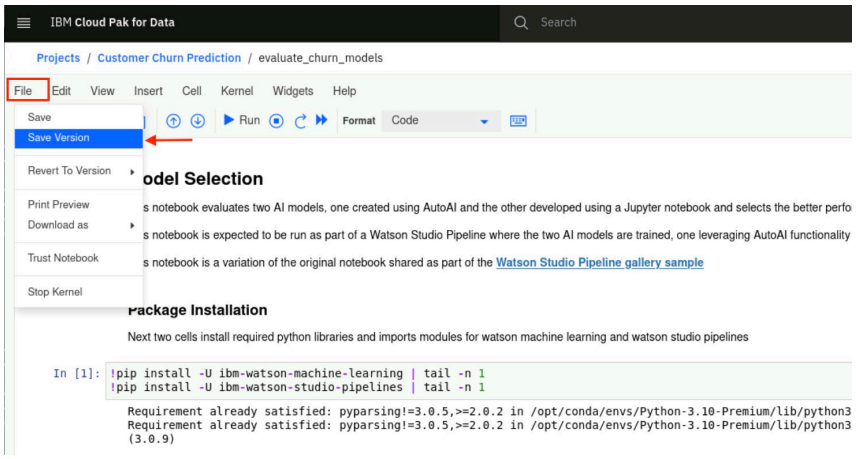
36- Next step is to add a Run notebook node to execute a Jupyter notebook which selects the better model between the two models, the one generated by the PySpark Churn Prediction notebook and the other by AutoAI experiment. Drag and drop the Run notebook job node (annotated with red rectangle) onto the canvas and connect both job_churn_prediction_notebook and AutoAI Churn Prediction nodes to the new Run notebook job node (annotated with red arrows). Double click the Run notebook job node (annotated with red oval) to edit its details.



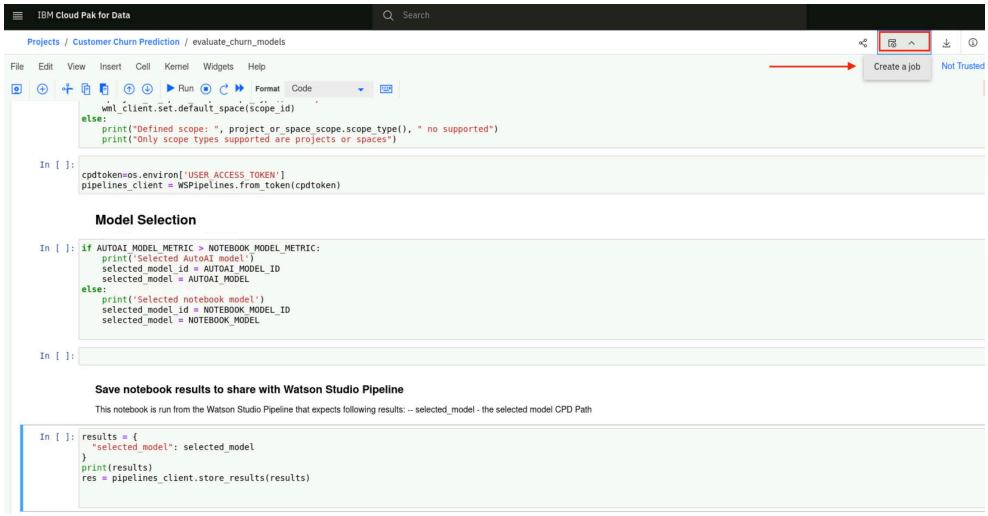
37- Change the name of the node to Select Better Model (annotated with red rectangle). To execute this node, we need to specify the job to execute this notebook. We will also need to pass environment variables to the notebook. Click Save.



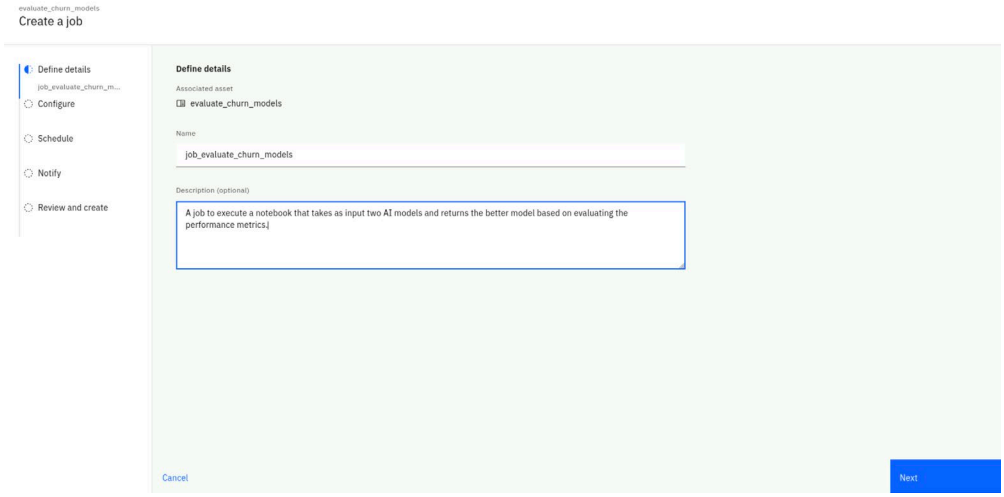
38- To populate the details of the newly added Run notebook job node, you need to upload the relevant Jupyter notebook to your project. To do so, download the evaluate_churn_models.ipynb notebook to your local machine. Then, upload this notebook to your Customer Churn Prediction project in a similar fashion to what you did earlier (navigate to your project Assets, and add a New Asset, select Jupyter Notebook and choose from file option). Once the notebook is loaded in edit mode, click the File menu (annotated with red rectangle) and select Save Version (annotated with red arrow).



39- Next, create a job to execute this notebook by selecting the jobs icon (annotated with red rectangle) and selecting Create a job (annotated with red arrow).

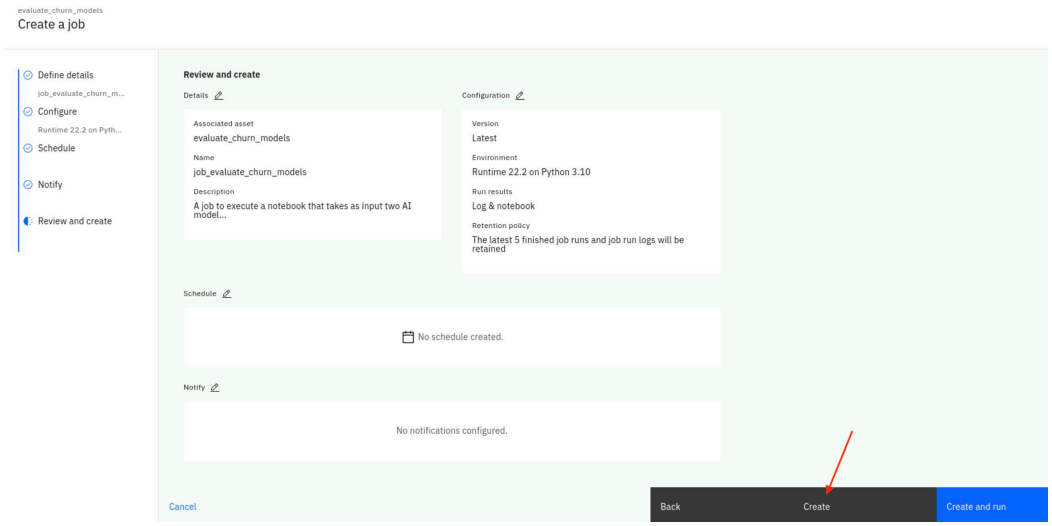


40- On the Create a job window, provide a name, job_evaluate_churn_models, and a description for the job and click Next.

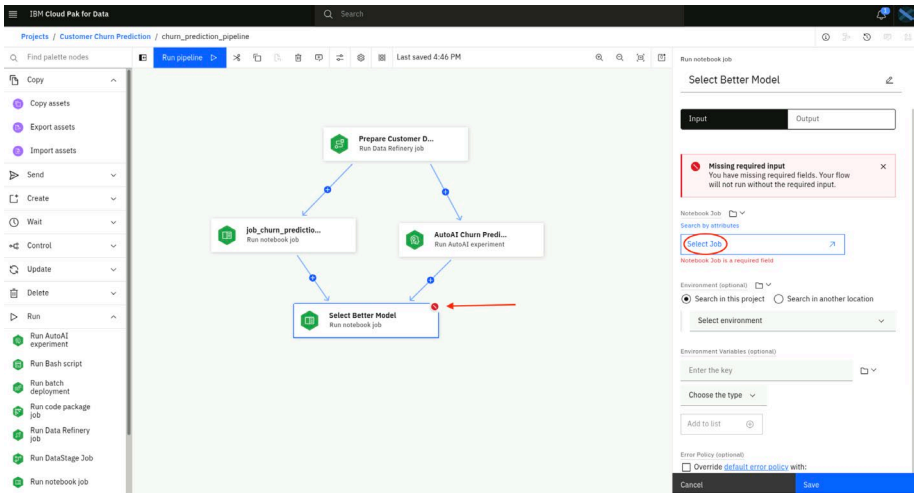


41- On the Configure, Schedule, and Notify steps, review the defaults and click Next without making any changes.

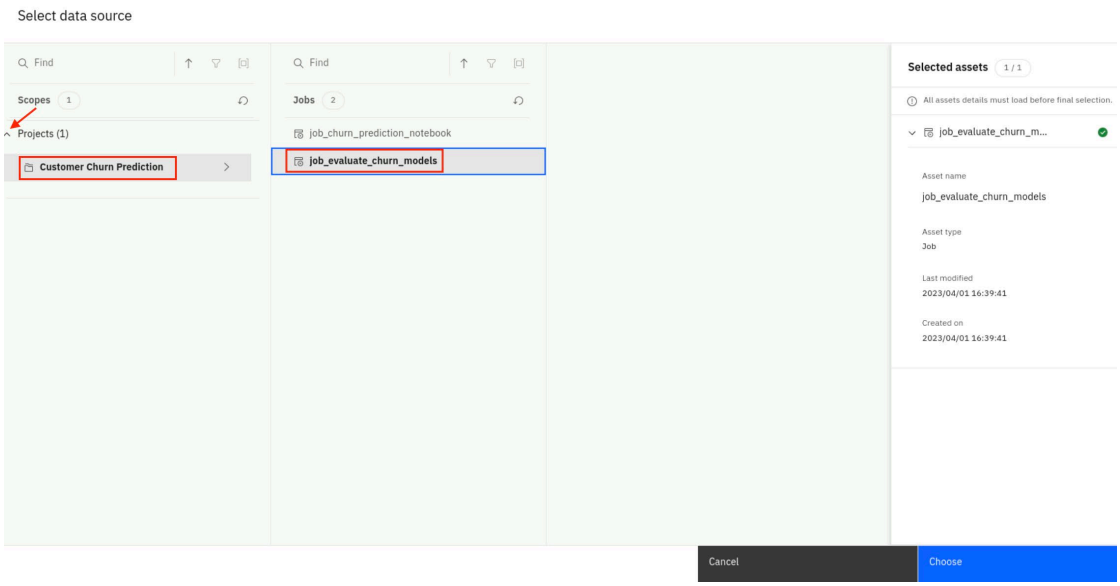
42- Lastly on the Review and create step, review the details of the job and click Create (annotated with red arrow). Please DO NOT select Create and Run as that will fail since the notebook expects certain environment variables to be passed from the Watson Pipeline.



43- Navigate back to the pipeline canvas in the other browser tab. If you had closed that, you can access it again by navigating to your project, selecting Assets tab, finding the Flows ➔ Pipelines and opening the churn_prediction_pipeline flow. Double click the Select Better Model not. Note the error marker (annotated with red arrow) indicating there is some configuration missing for this node. When you open the node, the error description is provided. In this case, a required parameter is missing which is the job to be executed. Click Select Job button (annotated with red oval).



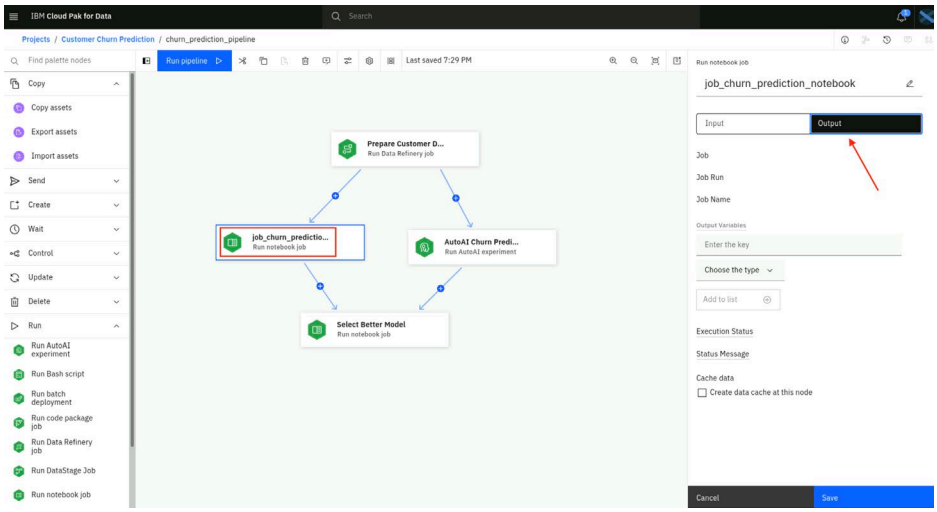
44- Click the projects selection drop-down (annotated with red arrow) and select your project (Customer Churn Prediction). From the list of defined jobs, select the job_evaluate_churn_models job and click Choose.



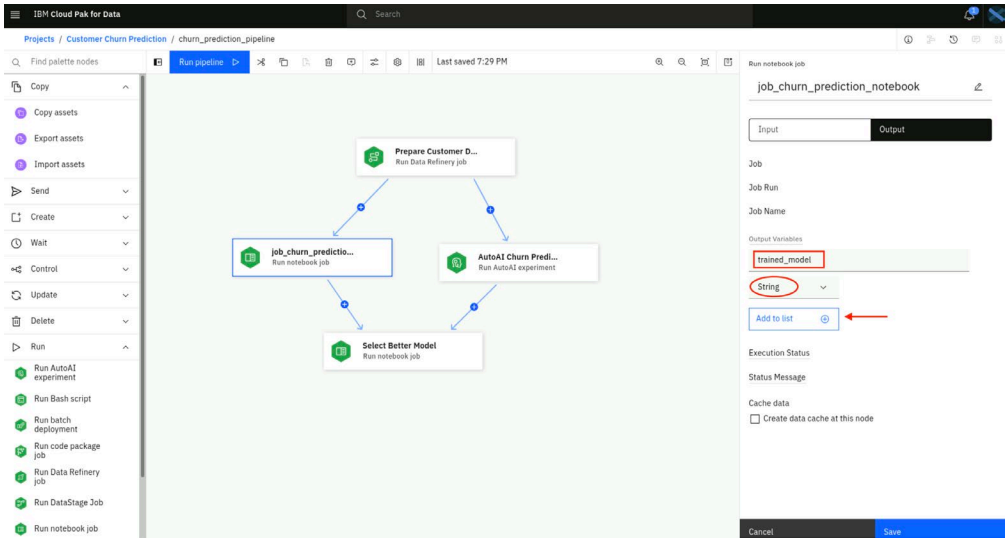
45- Next, you need to pass the required information to the job_evaluate_churn_models notebook job. This can be done via environment variables. Specifically, upon reviewing the notebook, observe that it expects the following 4 environment variables.

- ⇨ autoai_model: Path to AutoAI model generated by the AutoAI experiment.
- ⇨ autoai_model_metric: Evaluation metric from AutoAI model.
- ⇨ notebook_model: Path to churn prediction model generated by the notebook.
- ⇨ notebook_model_metric: evaluation metric from notebook model.

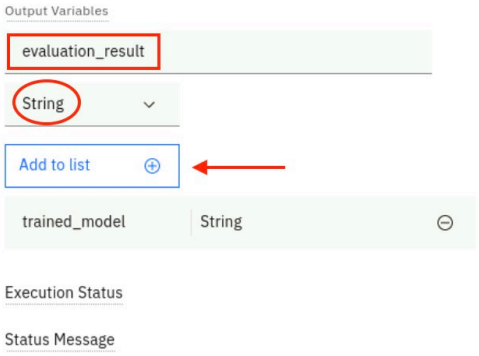
46- These environment variables need to be populated from other nodes in the pipeline, namely, the job_churn_prediction_notebook and the AutoAI Churn Prediction nodes. Double click the job_churn_prediction_notebook node (annotated with red rectangle) to open its details view. Click the Output tab (annotated with red arrow) to capture outputs of this node.



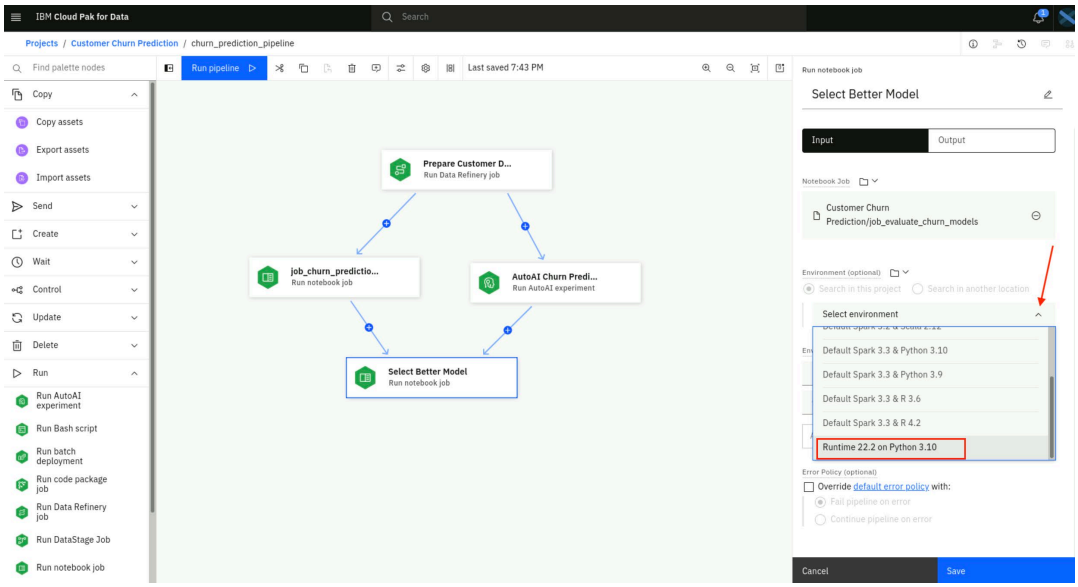
47- In the Output Variables section, type trained_model as the variable name (annotated with red rectangle), click the drop-down and select String for variable type (annotated with red oval) and click Add to list (annotated with red arrow). The trained_model name has to be exact because that is the output provided by the notebook.



48- Repeat the process to add another output variable, evaluation_result of type String and Add to list. Click Save to save these updates to the node.



49- Back on the pipeline canvas, double click the Select Better Model (annotated with red oval). In the Environments section, click the Environment selection drop down (annotated with red arrow) and select the IBM Runtime 22.1 on Python 3.10 (annotated with red rectangle).



50- Next, you'll define four environment variables as inputs to this notebook.

a. notebook_model: Path for the model generated by the PySpark Churn Prediction Notebook node.

i. Selection method: Select from another node

ii. Select node: job_churn_prediction_notebook

iii. Select an output: trained_model

b. notebook_model_metric: Evaluation metric for the model trained using PySpark Churn Prediction Notebook node.

i. Selection method: Select from another node

ii. Select node: PySpark Churn Prediction Notebook

iii. Select an output: evaluation_result

c. autoai_model: Path for Best model generated by AutoAI Churn Prediction node.

i. Selection method: Select from another node

ii. Select node: AutoAI Churn Prediction

iii. Select an output: Best Model

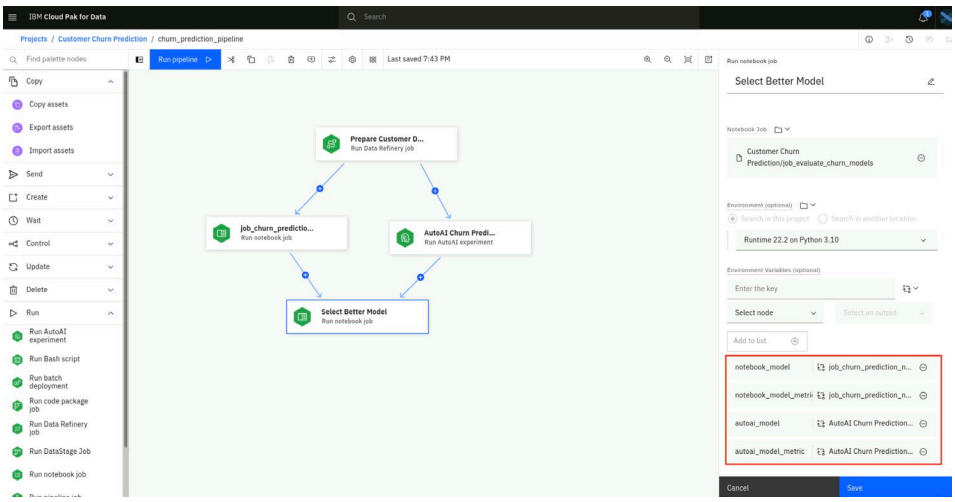
d. autoai_model_metric: Evaluation metric for the Best Model generated by AutoAI Churn Prediction node.

i. Selection method: Select from another node

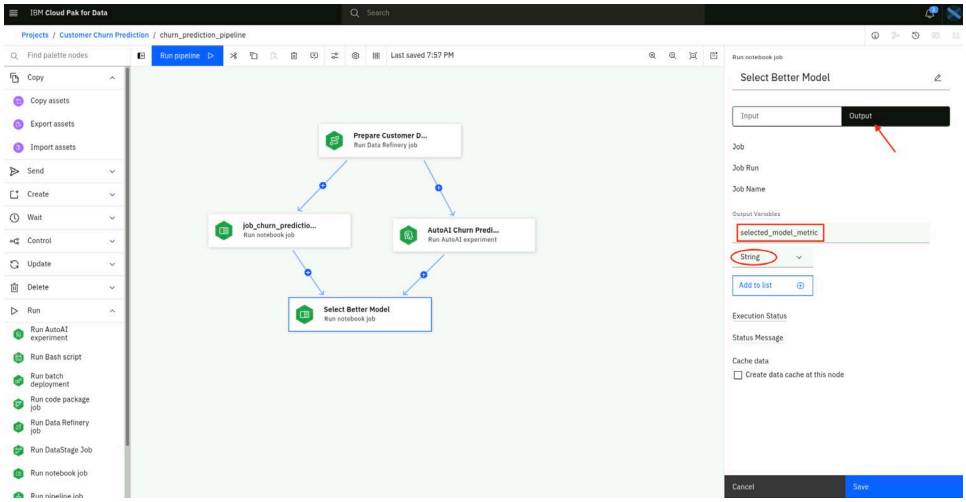
ii. Select node: AutoAI Churn Prediction

iii. Select an output: Winning Model Metric

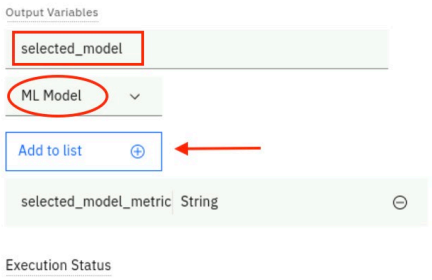
Click Save to save all the information for this node.



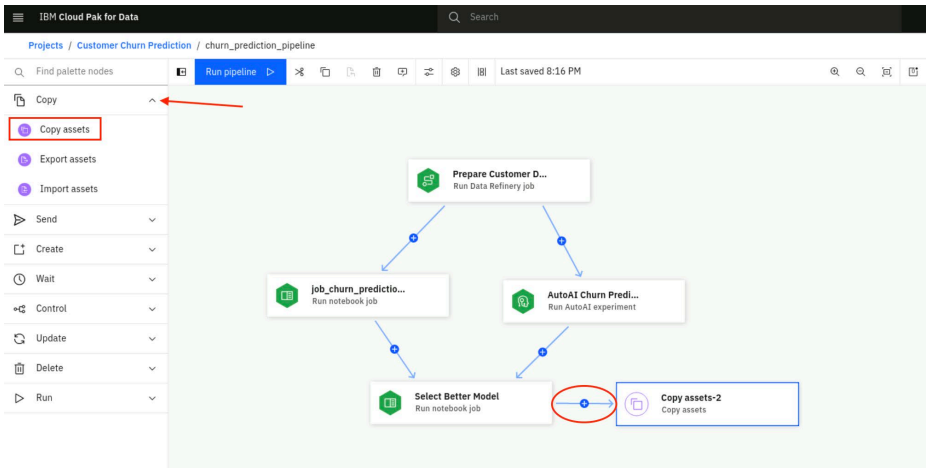
51- Next, we need to define the output from the Select Better Model node. Double click the Select Better Model node again to open its details view and then click the Output tab (annotated with red arrow). In the Output Variables section, type selected_model_metric as the variable name (annotated with red oval), click the drop-down and select String for variable type (annotated with red rectangle) and click Add to list.



52- Repeat the process to add another output variable called selected_model (annotated with red rectangle) of type ML Model (annotated with red oval) and click Add to list (annotated with red arrow). Click Save.



53- Next, add Copy Assets node to the canvas with the objective of copying the selected model to preProd deployment space. To do so, expand the Copy section (annotated with red arrow) in the palette and drag/drop the Copy assets node (annotated with red rectangle) to the right of the Select Better Model node. Hover over or select the Select Better Model node so that the connection arrows appear; then select the connection arrow on Select Better Model node and drag it over the Cope assets-2 node to connect the two nodes.

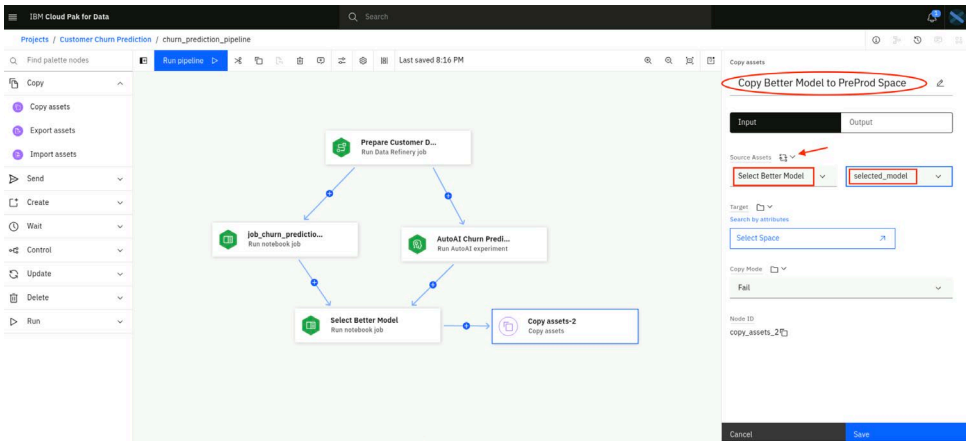


54- Double click the Copy assets node to open the details view for this node and provide required information. Change the title of the node to Copy Better Model to PreProd (annotated with red oval). For the Source Assets, specify the following:

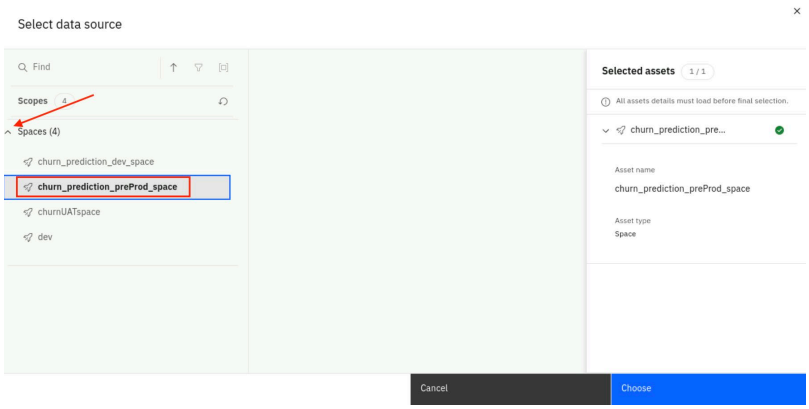
a. Selection method: Select from another node (annotated with red arrow)

b. Select node: Select Better Model

c. Select an output: selected_model



55- Next, click Select Space button to specify which space to copy assets to. On the space selection window, click the Spaces drop down (annotated with red arrow) and select the churn_prediction_preProd_space (annotated with red rectangle) and click Choose.

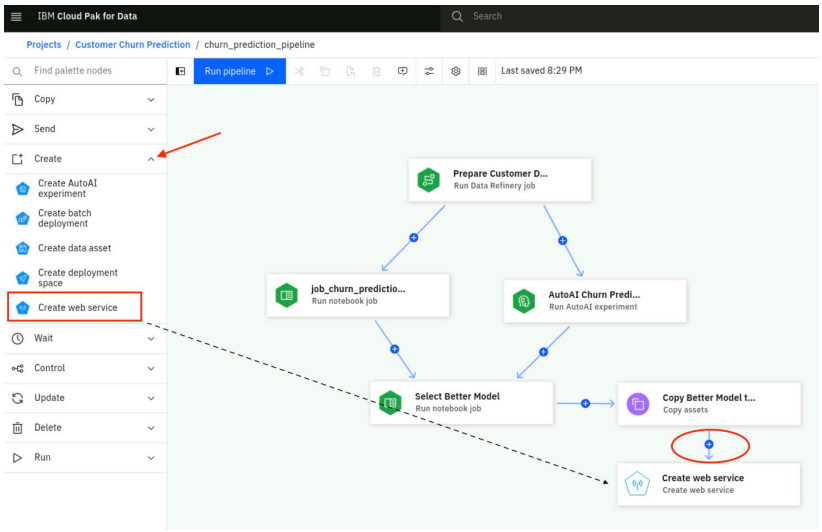


56- Lastly, update the Copy Mode to Overwrite (annotated with red rectangle).

Click Save to save all the configurations for the Copy Assets node.

57- The last node in this pipeline involves adding a node for creating a web service from the model in the pre-production deployment space so it can be accessible via REST API. To do so, expand the Create section in the palette (annotate with red arrow) and drag/drop the Create web service node (annotated with red rectangle) onto the canvas below the Copy Better Model to PreProd node. Hover over or select the Copy Better Model to PreProd node so that the connection arrows appear; then select the connection arrow on the top node and drag it over the bottom node so the connection

arrows on the other node appear and release to make the connection between the two nodes (annotated with red oval).



58- Double click the Create web service node to open the detailed view. Change the title to Churn Prediction Web Service. For the ML asset, specify the following:

a. Selection method: Select from another node (annotated with red arrow)

b. Select node: Copy Better Model to PreProd (annotated with red rectangle)

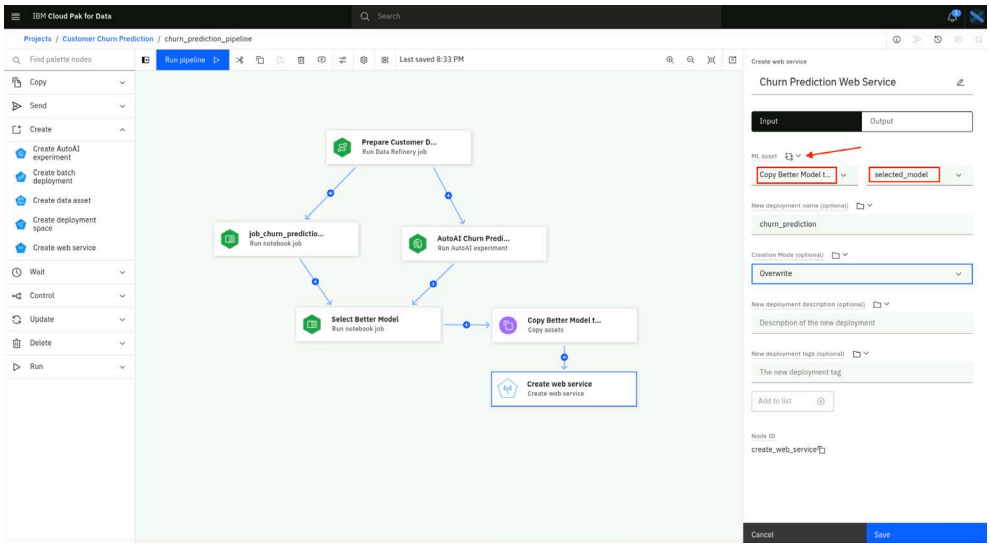
c. Select an output: selected_model (annotated with red rectangle)

Provide the New deployment name as churn_prediction (annotated with red oval).

For Creation Mode field, select Overwrite.

Optional Provide a description for the new deployment.

Click Save.



59- Now you have a complete pipeline which includes the following steps:

a. Run data refinery flow to join 3 data assets into one CUSTOMER_DATA_ready asset.

b. Execute a notebook job for training a PySpark model for churn prediction using the CUSTOMER_DATA_ready data set which is generated by the Data Refinery job.

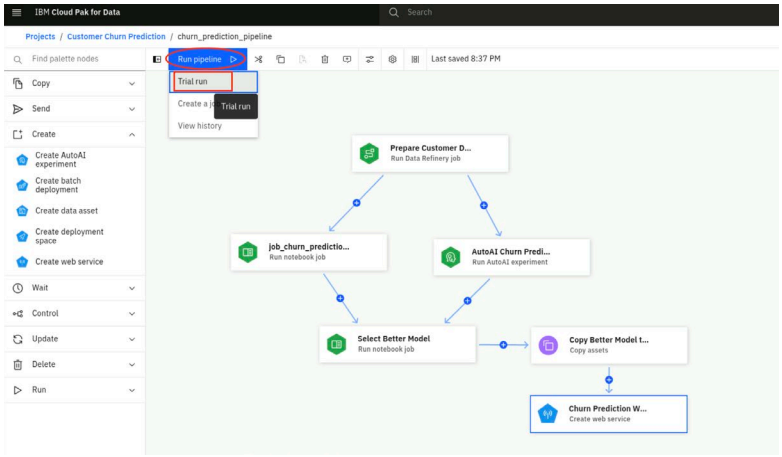
c. Run an AutoAI experiment using the CUSTOMER_DATA_ready data set.

d. Run another notebook to select the better model between the notebook model and the AutoAI model.

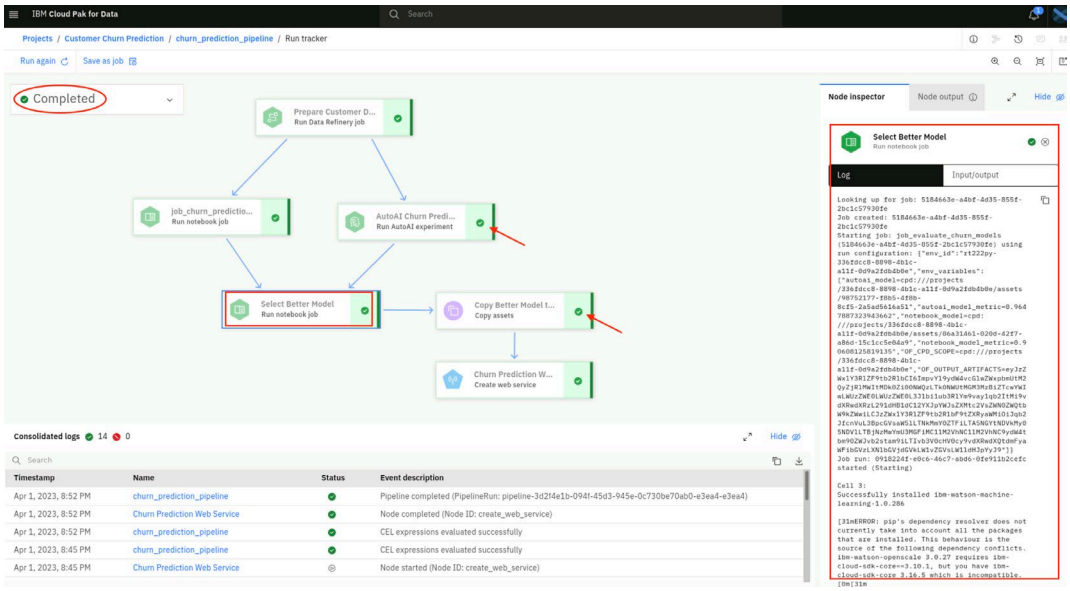
e. Copy the better model to preProd deployment space.

f. Create a Churn Prediction Web Service

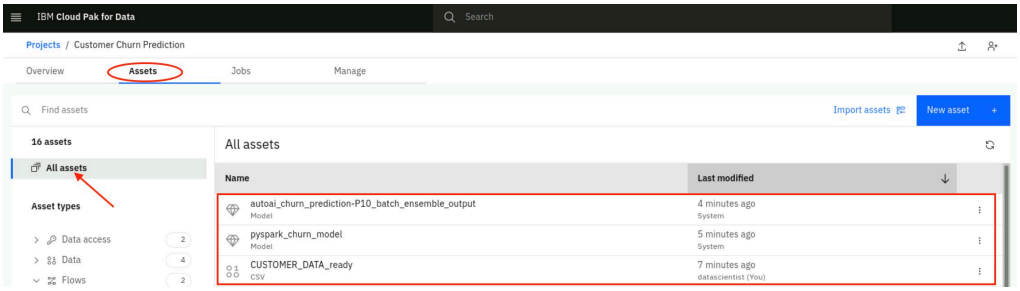
60- Click Run pipeline button (annotated with red oval) and select the Trial run option (annotated with red rectangle).



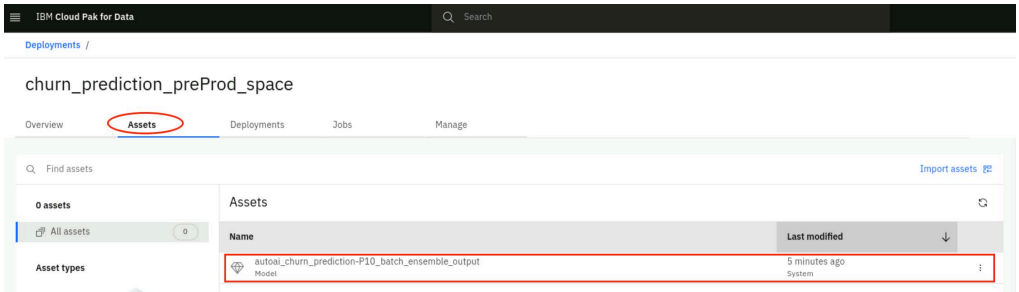
61- Wait for the pipeline to run through the different steps outlined above. It will take 3-5 minutes to complete. When the pipeline run completes, you should see the Completed status (annotated with red oval) and all nodes/stages of the pipeline having green checkmarks (annotated with red arrows) indicating successful completion. If you'd like to check the logs of any node, you can select that node and the logs will display in the log area to the right. For example, click the Select Better Model node (annotated with red rectangle) and you will see the logs in the log area (annotated with red rectangle).



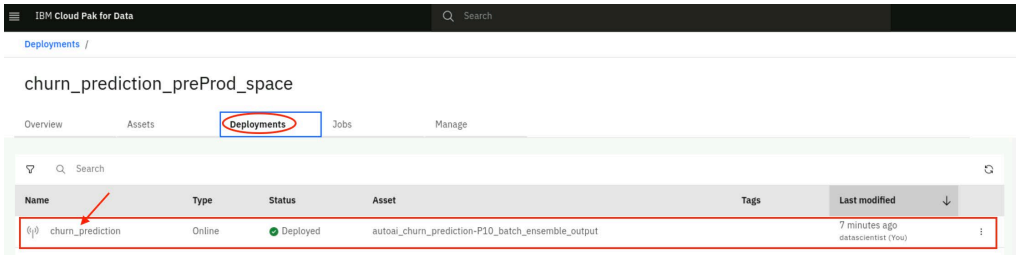
62- Navigate back to your Customer Churn Prediction project either by clicking the project name in the breadcrumbs or by clicking the Navigation menu, selecting All Projects and then selecting your Customer Churn Prediction project. On your Customer Churn Prediction project, click **Assets** tab (annotated with red oval) and select All assets (annotated with red arrow) to show all assets in the project. Confirm the recently created assets, a few minutes ago (annotated with red rectangle), the CUSTOMER_DATA_ready data asset which was generated by the Data Refinery flow triggered by the pipeline and the autoai_churn_prediction_*** model generated by AutoAI experiment run via the pipeline.



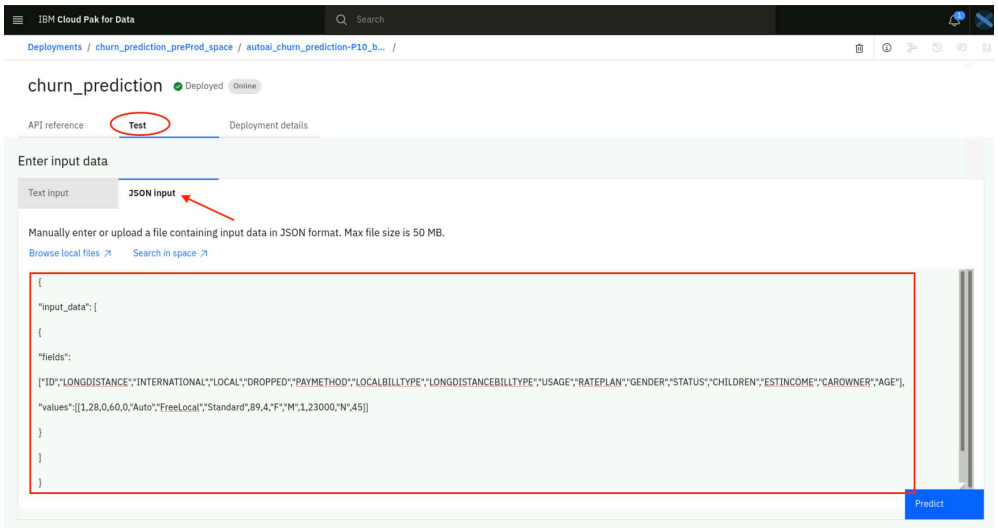
63- Next, navigate to the **churn_prediction_preProd_space** deployment space and click the **Assets** tab (annotated with red oval). Note the autoai_churn_prediction_*** model which was copied to the preProd deployment space by the Copy Better Model to PreProd node/stage in the pipeline.



64- Lastly, click the **Deployments** tab (annotated with red oval) in the churn_prediction_preProd_space deployment space and confirm the **churn_prediction** deployment (annotated with red rectangle) which was created by the Churn Prediction Web Service node/stage in the pipeline. Click the **churn_prediction** deployment (annotated with red arrow).



65- On the churn_prediction deployment page, click the Test tab (annotated with red oval). Select JSON input (annotated with red arrow) to provide the input payload in a JSON format. Provide a sample payload by copying it from AutoAI Payload Sample boxnote and click Predict.



66- Note the response (annotated with red rectangle) from the deployment model showing the likelihood of this customer to churn. Click the JSON view (annotated with red arrow) to see the response in JSON format or try the Table view (annotated with red oval) to see the response in Table format.

This concludes the first part of this module where you created the Watson pipeline step-by-step to automate the tasks done in earlier modules so they can be re-run and executed on-demand.

Lab B – Importing and running a pre-built pipeline

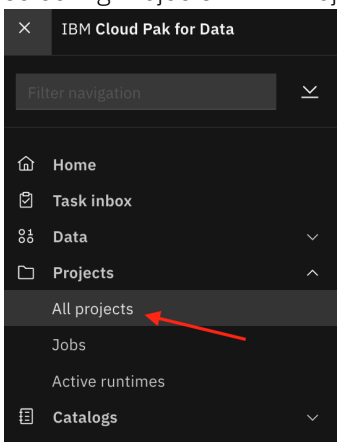
Prerequisites and setup

This lab is self-contained and does not have any pre-requisites other than the correct Cloud Pak for Data services being available.

Import the pipeline

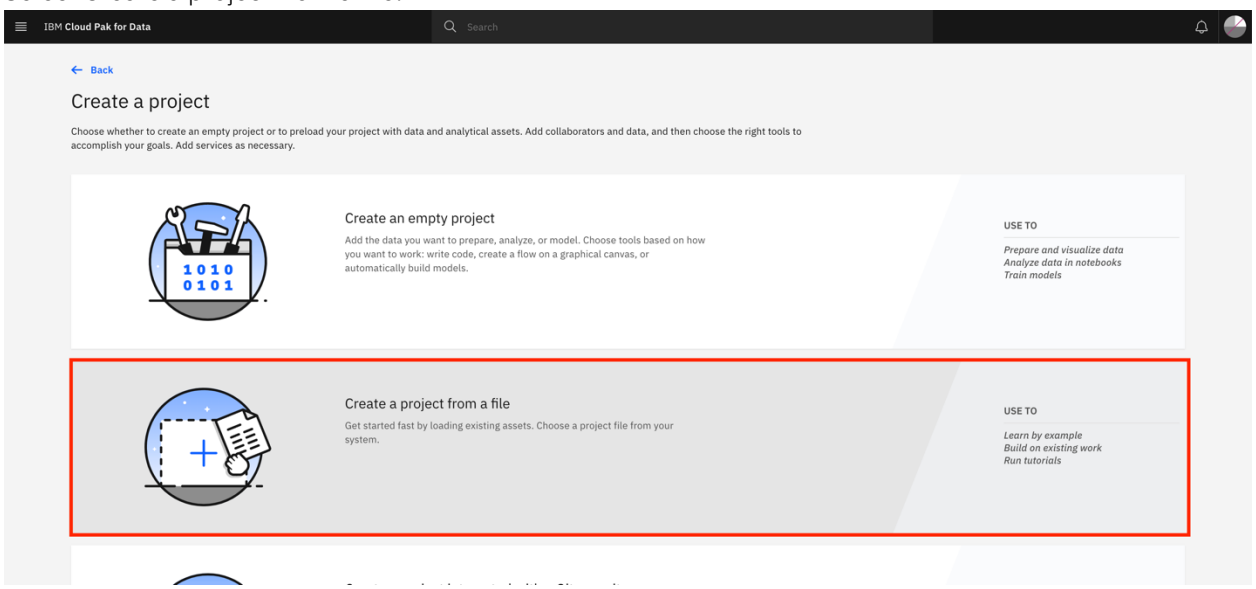
1- Log into Cloud Pak for Data as datascientist user.

2- Navigate to your Projects view by clicking on the Navigation menu (top left hamburger icon) and selecting Projects ➔ All Projects (annotated with red arrow).

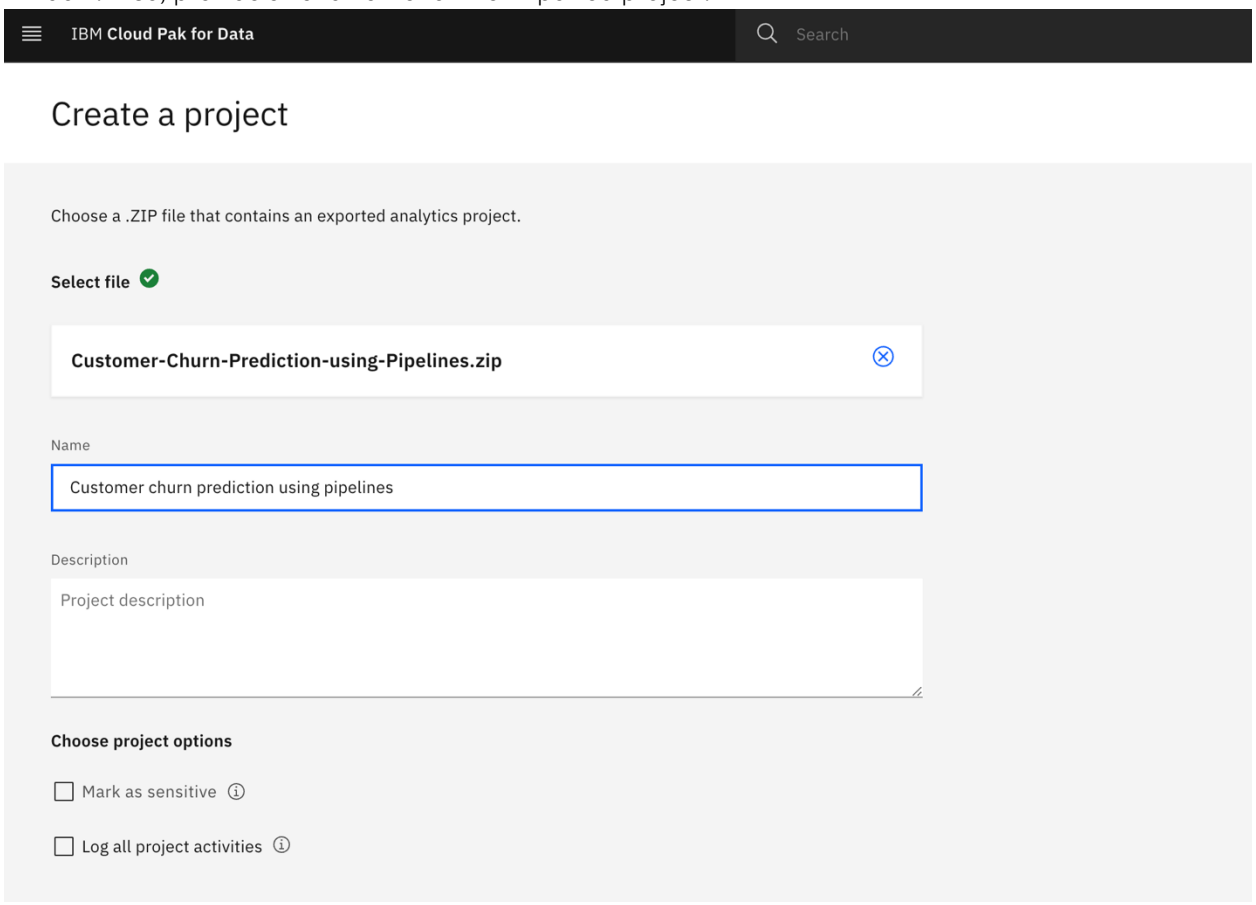


3- Create a new project by clicking the New Project button.

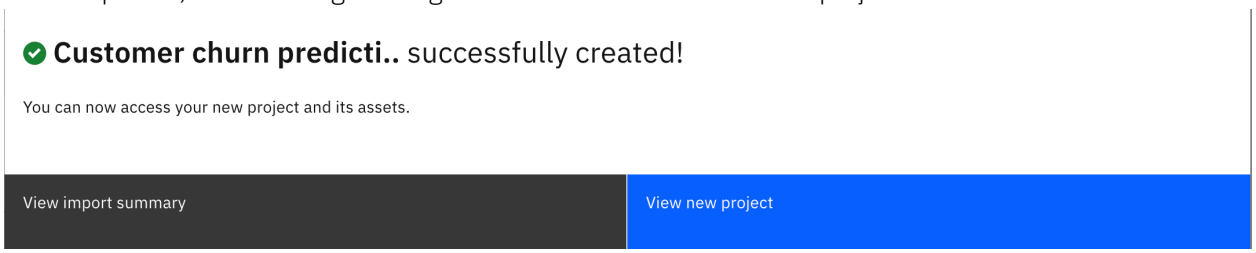
4- Select Create a project from a file.



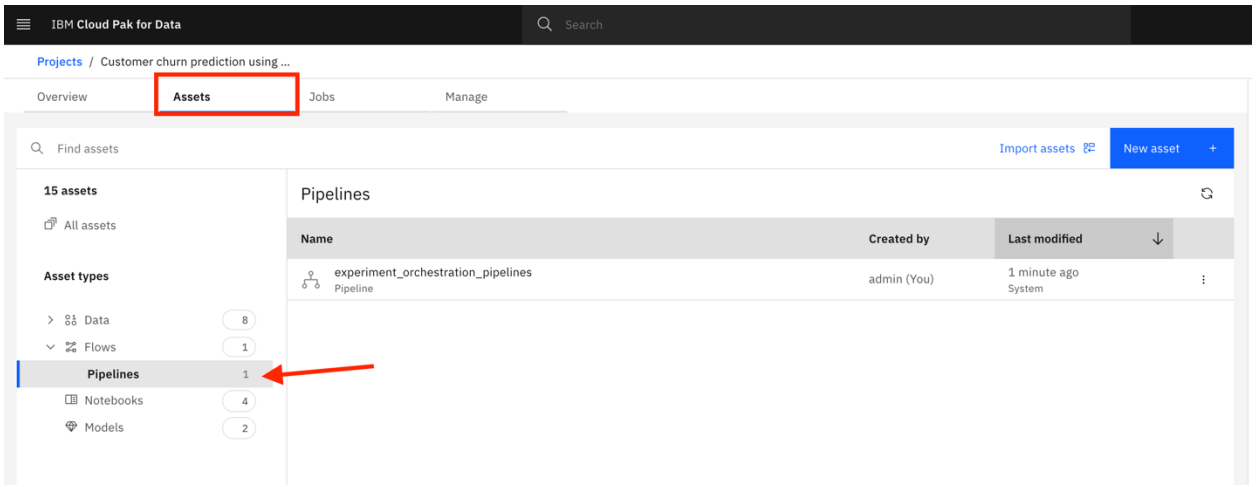
5- Browse for the “Customer-Churn-Prediction-using-Pipelines.zip” file or drag the file the browser window. Also, provide a valid name for the imported project.



Once imported, the following message will be shown. Click View new project.

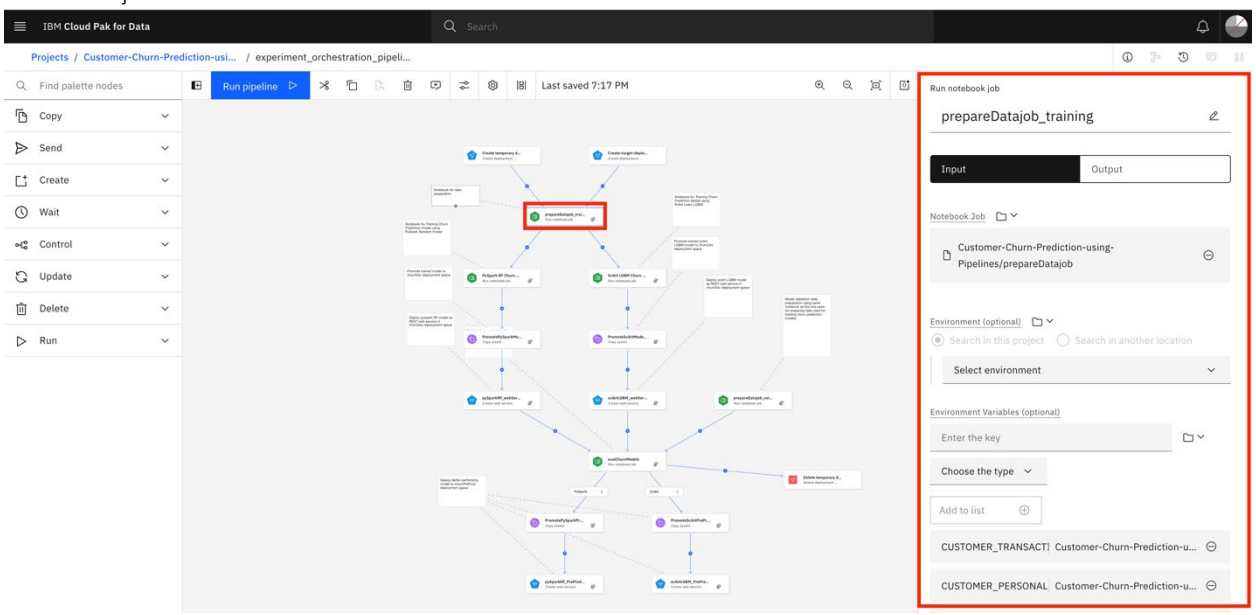


6- Navigate the project assets and select Flows → Pipelines in the navigation menu. There are quite a few assets in this project: notebooks, data files and models. For this exercise we only need to focus on the pipeline.



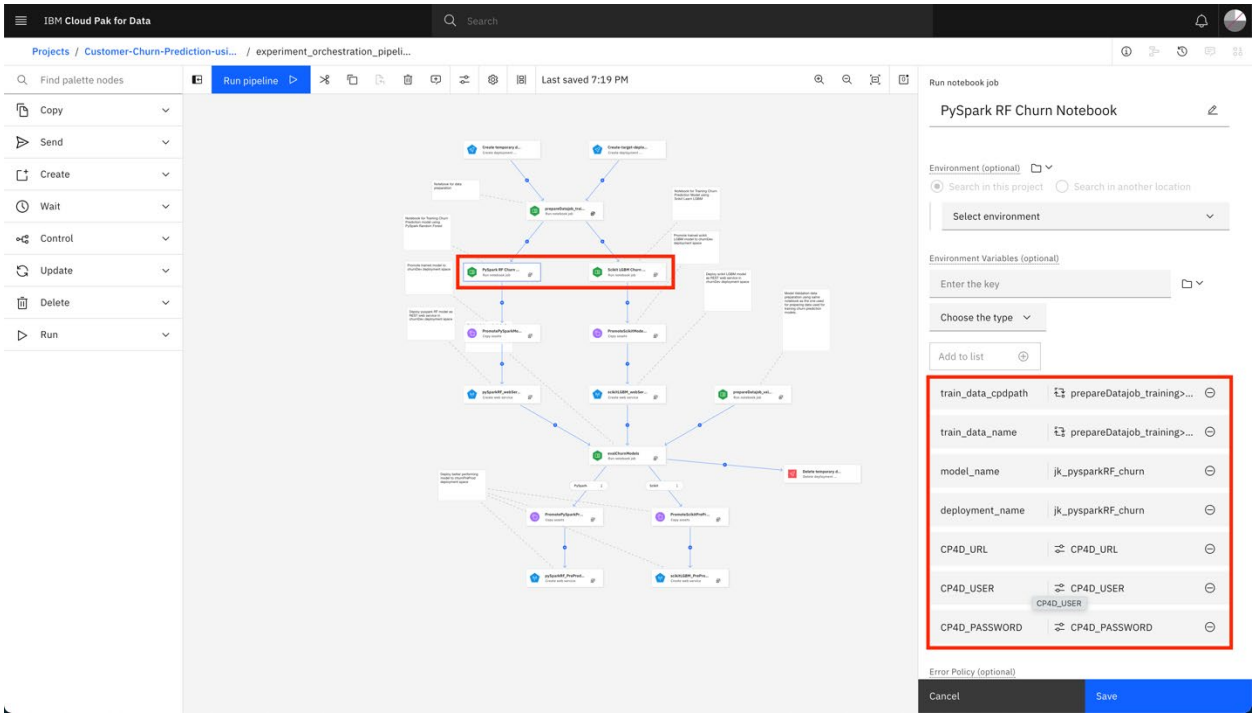
Click on the experiment_orchestration_pipelines flow.

7- You will see a pipeline consisting of many nodes. Just below the top nodes which create the deployment spaces, you will find the prepareDatajob_training node, which runs a notebook that prepares the data sets for training the models. Double-click on the node to see the details of the Run notebook job action.



You will notice that a few variables such as CUSTOMER_TRANSACTION_DATA, CUSTOMER_PERSONAL_INFO, and CUSTOMER_CHURN_LABELS are passed to the notebook as environment variables.

8- In the next step, two models are trained in parallel, one PySpark Random Forest model and the other a SciKit Learn LGBM model. The goal of the pipeline is to compare the performance of these two models and deploy the one that has the highest accuracy. While this pipeline shows only two notebooks, it illustrates the capability for experiment(s) orchestration where multiple data scientists can each work on their own notebook/solution for the same use case using different model architectures and then the various generated models get evaluated to choose best performing one.

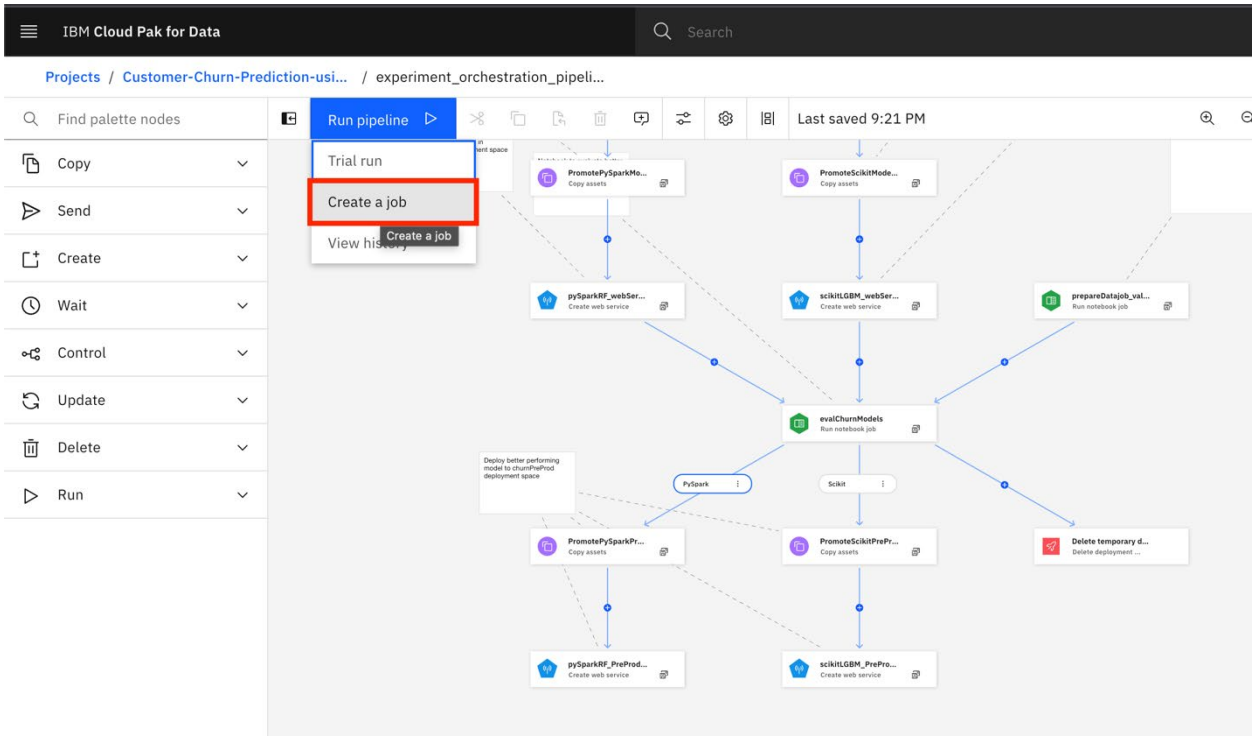


Again, if you double-click on the PySpark RF Churn node you will see the details of the notebook that is being run and the parameters passed to the notebook as environment variables. In this case, also the Cloud Pak for Data URL, the user and password are required. These will be specified when you run the pipeline.

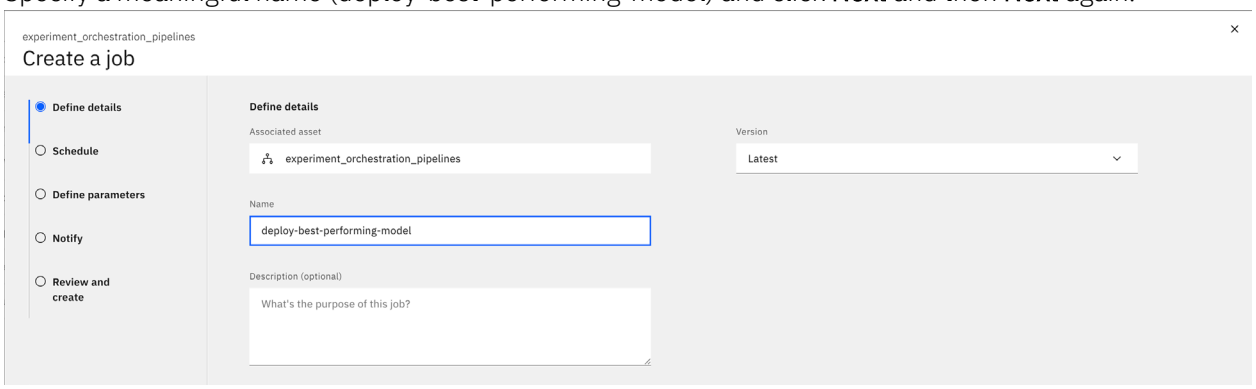
You can check each of the nodes to understand what action it takes. To clarify certain steps, notes have been added into the pipeline. The notebooks that train the models save the model into the project, then they are deployed as online models (web services) in the temporary deployment space that was created at the beginning of the pipeline.

The evalChurnModels notebook compares the performance of the two models, respectively the PySpark model and the SciKit learning model. The model with the better performance is returned in the bestModel user variable of the pipeline, which then triggers the copying of the model into the specified deployment space (dev is the default) and the subsequent deployment of the model.

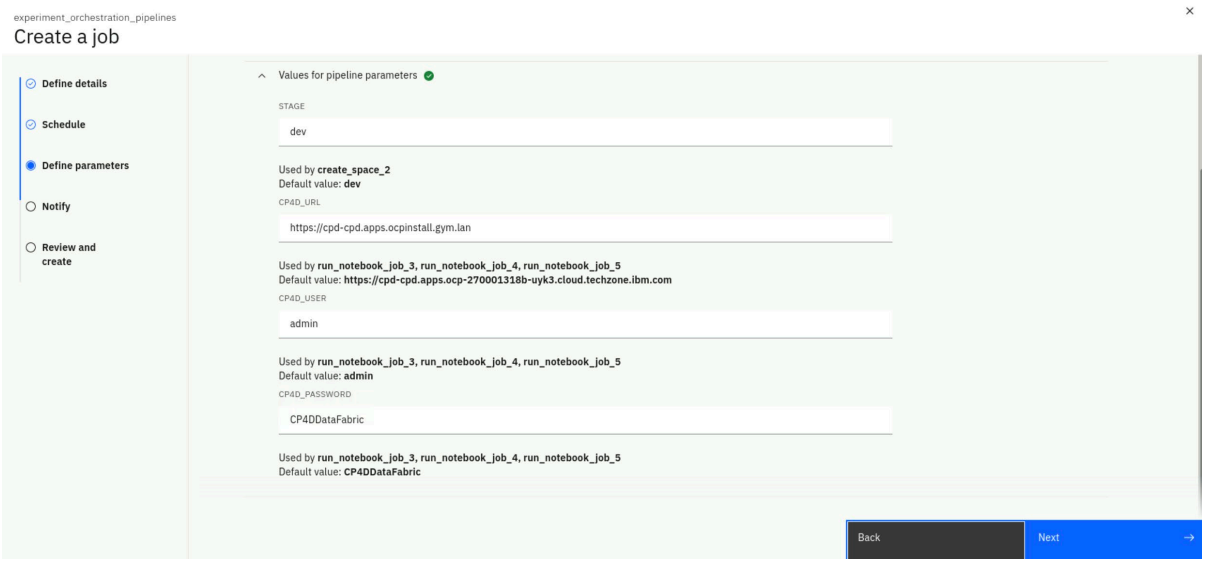
9- Once you've checked the different nodes, run the pipeline by clicking the Run pipeline button. Choose Create a job.



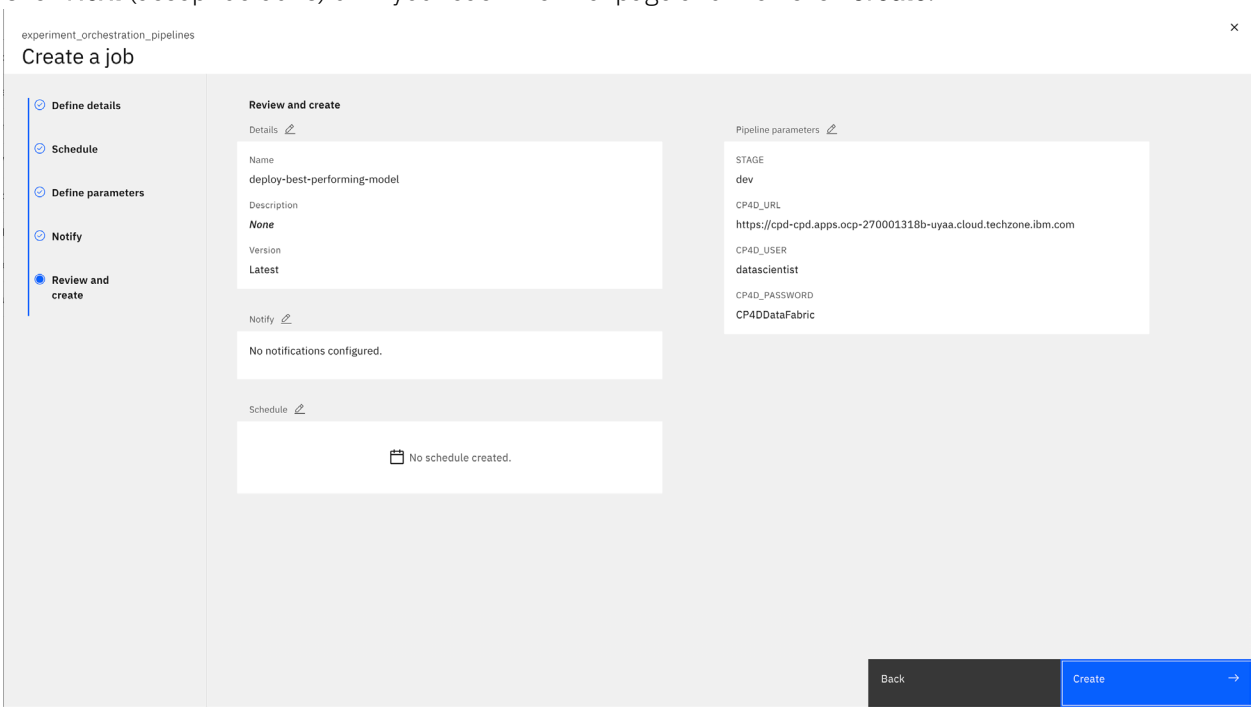
10- Specify a meaningful name (deploy-best-performing-model) and click **Next** and then **Next** again.



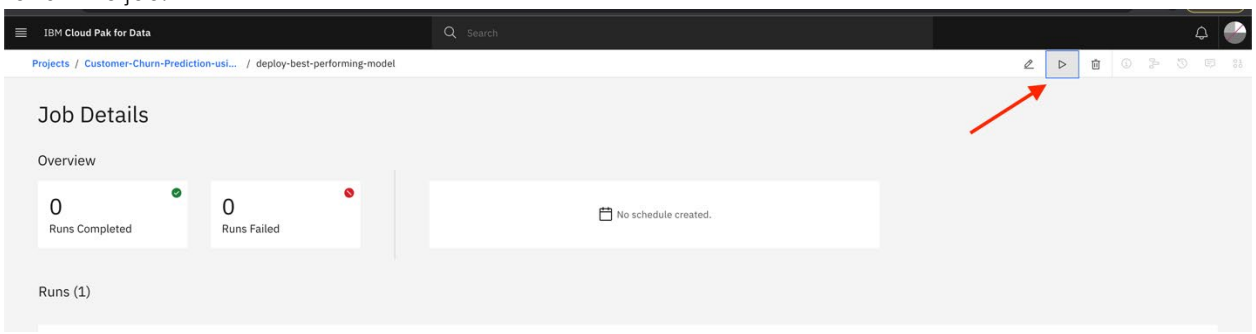
11- Specify the URL of your Cloud Pak for Data installation and the user and password. Then click **Next**.



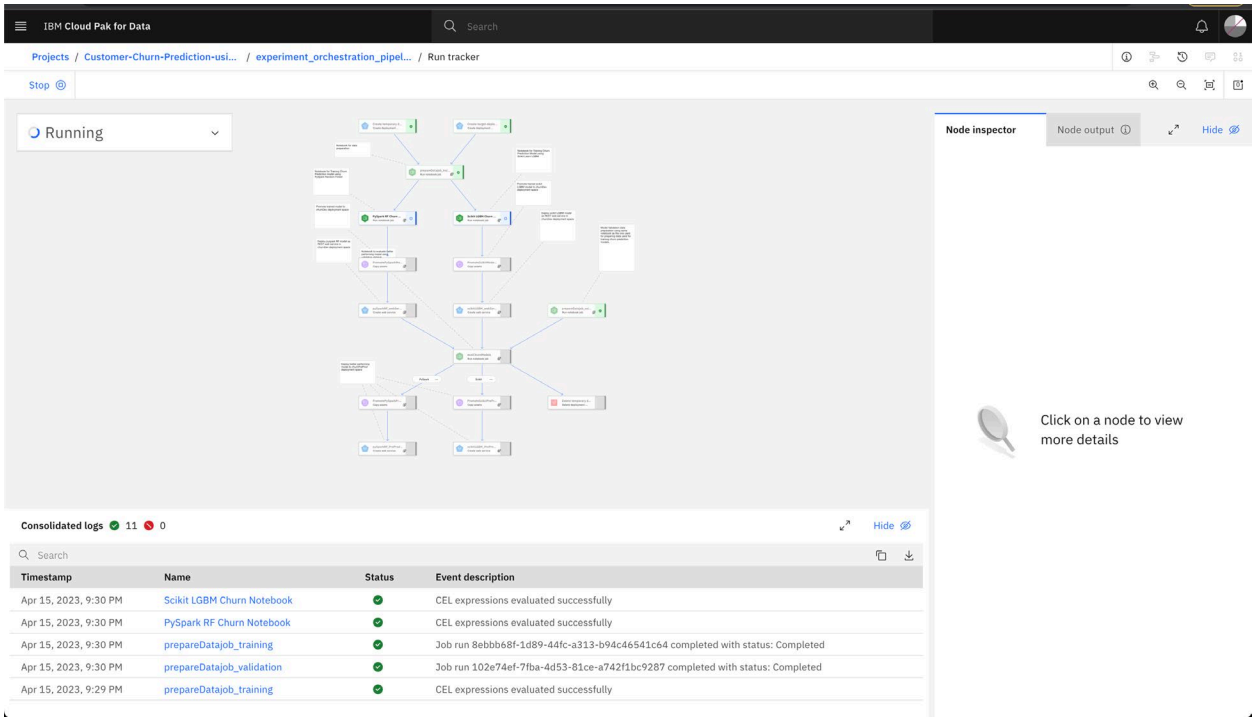
12- Click **Next** (accept defaults) until you reach the final page and then click **Create**.



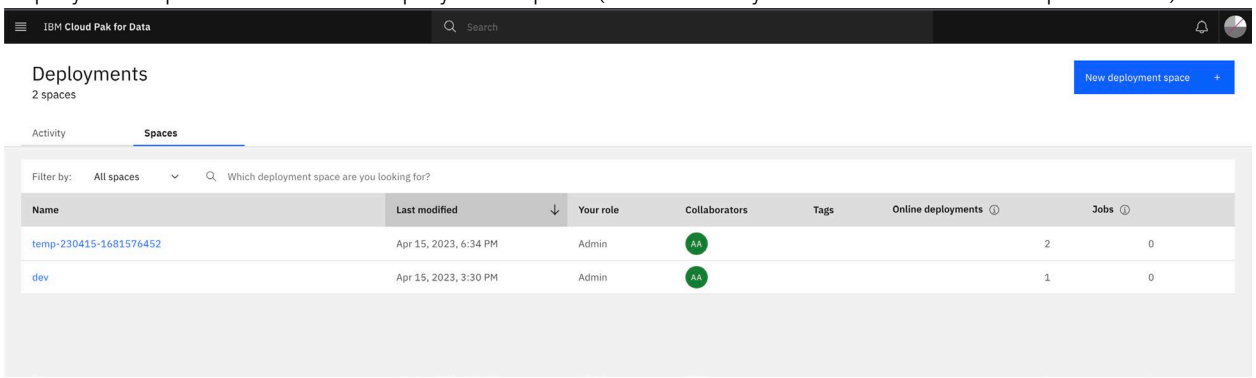
13- Once the job has been created, click the Play button (annotated with red arrow) at the top of the page to run the job.



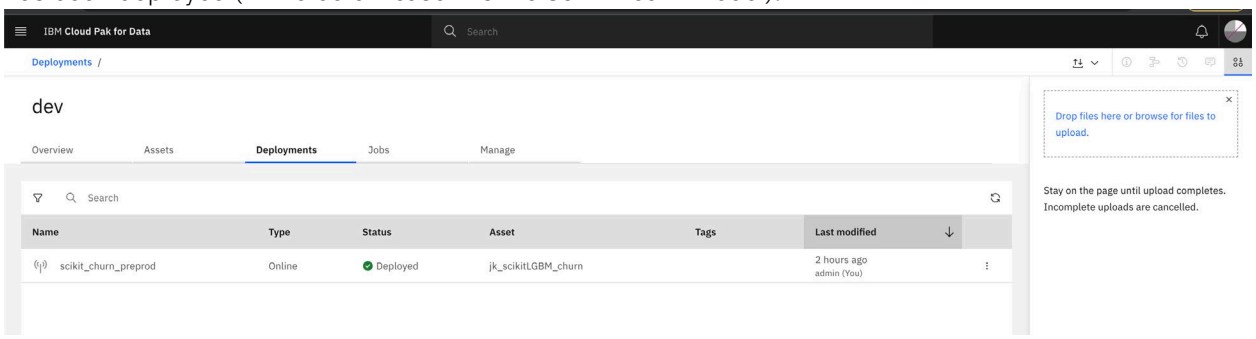
14- You can monitor the pipeline and click on a node that is running or has finished to view the logs. The job will take 5-10 minutes to complete.



15- Once finished, click on the hamburger menu and select Deployments. You will see the temporary deployment space and the dev deployment space (or whatever you chose for the STAGE parameter).



16- Click on the dev deployment space and navigate to deployments. You will see that one of the models has been deployed (in the below case it is the SciKit Learn model).



Summary

In this lab, you've seen how to leverage Watson Pipelines to automate the process of data preparation using Data Refinery, training AI models using different techniques (notebooks and AutoAI), selecting best performing models and creating an online deployment of such models. You could create a job of this pipeline which can be triggered from any automation tooling or could be scheduled to run periodically. Also note that since the input to Data Refinery was virtualized data, then as the source data changes, that gets captured automatically when the Data Refinery flow is run.

Other common scenario for pipelines include:

- ⇨ Comparing challenger models to deployed models: Following a similar pattern, you can compare new challenger models created either using new data or new algorithms against model deployed in production and if results are better, can notify data science lead so they can review results and decide whether to deploy new model.
- ⇨ Automating the training of different models using different notebooks / algorithms and selecting best performing models.