Azure Machine Learning Tutorial

Avneet Kaur
Negar Sadreddini
Satyam Vatts

Table of Contents

AZURE MACHINE LEARNING ***3***

Overview **3**

Registration **3**

Azure for Students Subscription 6

Machine Learning Service **7**

Creating Workspace 9

Machine Learning Studio **14**

Selecting the Correct Workspace 14

Compute Targets **15**

Compute Instance 15

Compute Cluster 16

AML Designer **18**

Dataset Creation 18

Machine Learning Pipeline 20

Run the pipeline 26

Deployment of the Pipeline 29

Automated ML **33**

Regression Model for USA-Housing dataset 34

Classification Algorithm Based on MNIST Dataset 40

Notebooks **49**

Development Phase 49

Deployment Phase 53

AZURE MACHINE LEARNING

Azure is a cloud platform offered by Microsoft that has more than 200 products and cloud services. Most of the companies in Canada use Microsoft ecosystem consisting of OS, Office, Power BI etc. Thus, Azure becomes an obvious choice for cloud provider. Although, AWS is the largest and leading cloud service provider, the cloud computing concepts explained in this tutorial are applicable to AWS as well. There are differences in terms of terminology and interface, the operations and services are similar in functioning.

We have focused on Azure Machine Learning as part of this tutorial to lay out the initial groundwork required for students to build, configure, and deploy their machine learning models to cloud platform.

Overview

The AML offers enterprise grade service for end-to-end machine learning lifecycle. In this tutorial we are going to cover some of these services that can help students get comfortable with environment and explore other services at ease.

- Registration: Getting a free Azure account and Azure for Student subscription
- Machine Learning Service: Initial Step and Configuration
- Machine Learning Studio:
    - Selecting the Correct Workspace
- Setting up Compute Infrastructure
- AML Designer: Drag-Drop ML Pipeline
- Automated ML: No Code Auto Model Selection & Deployment
- Notebooks: IDE on Cloud with Continuous Development & Deployment

Registration

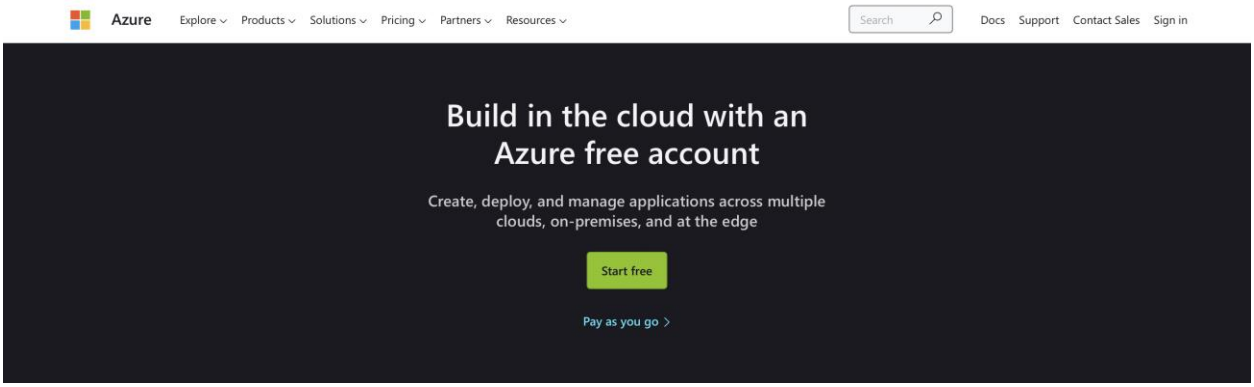
Registering for Azure cloud services is easy and fast. You get a free account with $200 Credits for normal subscription and $100 free credits for student subscription.

Prerequisites:

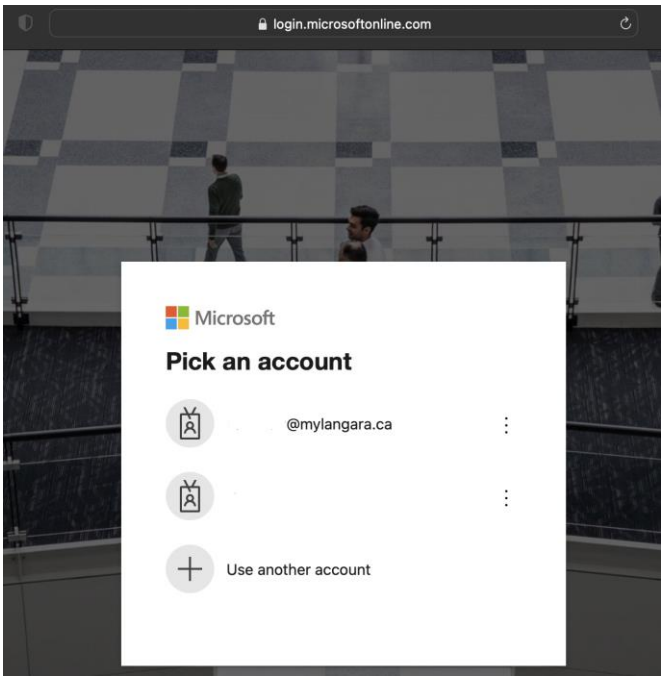
- Your college O365 account credentials
- Credit/Debit Card: Required for Normal Subscription

Azure Free Account: Normal Free Subscription

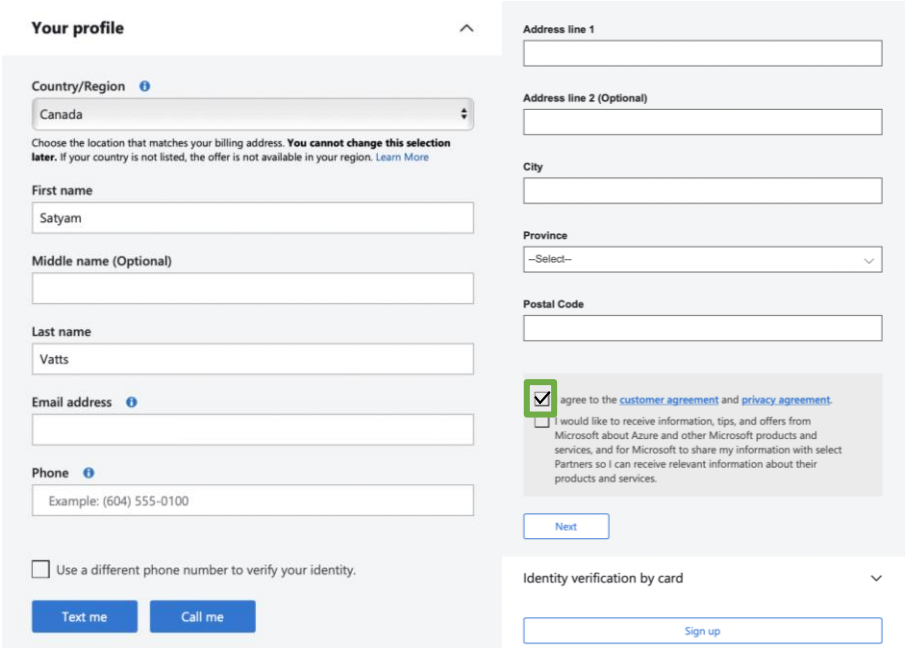
Step 1: Go to Microsoft Azure Page and Click on "**Start Free**"



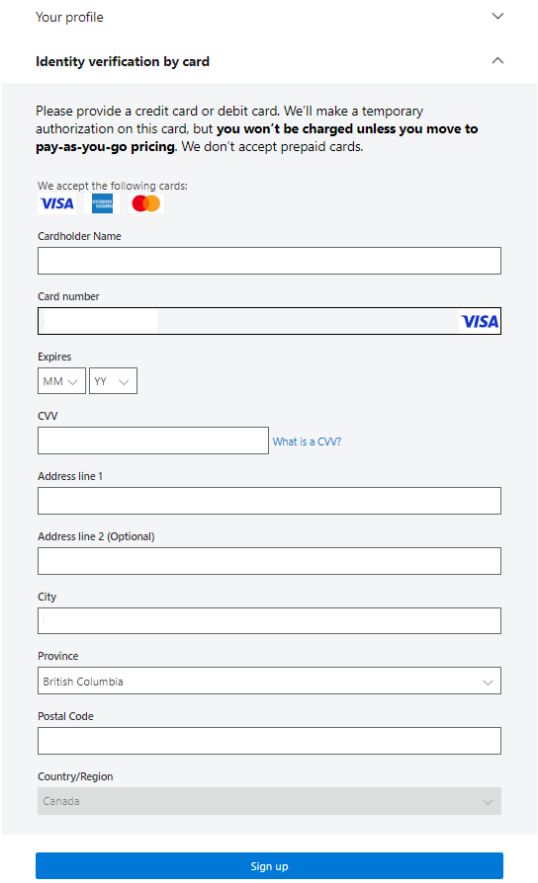
Step 2: Login using your college O365 account credentials



Step 3: You will get a Sign-Up form if you do not have a pre-existing Azure account. (In case you already have an Azure account, skip to next section)
Fill up form with required details. You also need to verify your phone number. Use any of the method provided “Text Me” for SMS verification or “Call Me for voice verification.

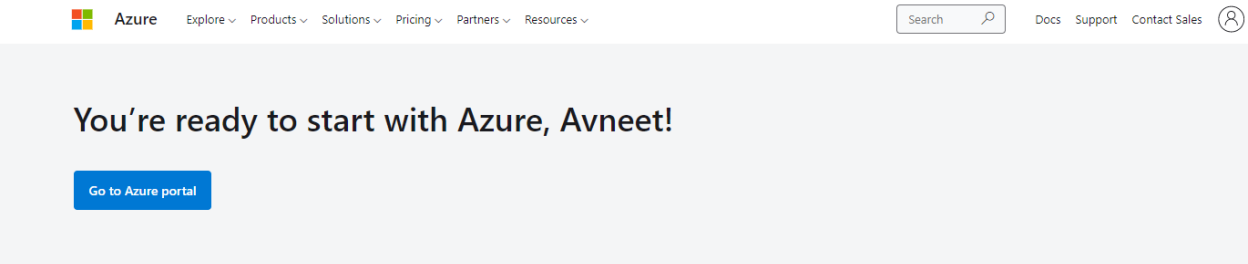


After clicking on Sign Up, you will be prompted to enter Payment Details. Do not worry about it. There are no deductions unless you use over your free credits, or your subscription expires.



After entering payment information, click on Sign Up.

That's it, you have your free Azure Account.

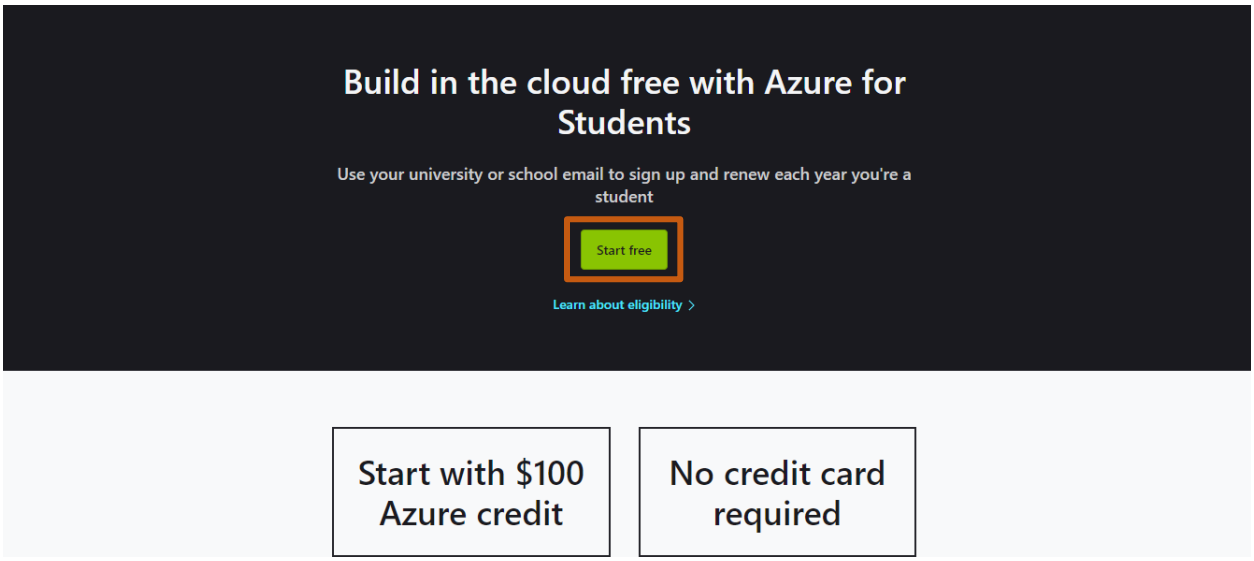


On the Welcome Page, Click on Go to Azure Portal

Azure for Students Subscription

The above section creates a default subscription for the Azure account that provides $200 credits for 1 month. You can also sign up for Student subscription if you have a valid School Email ID. Follow the below steps to get Azure for Students Subscription (*this is required for completion of this tutorial)

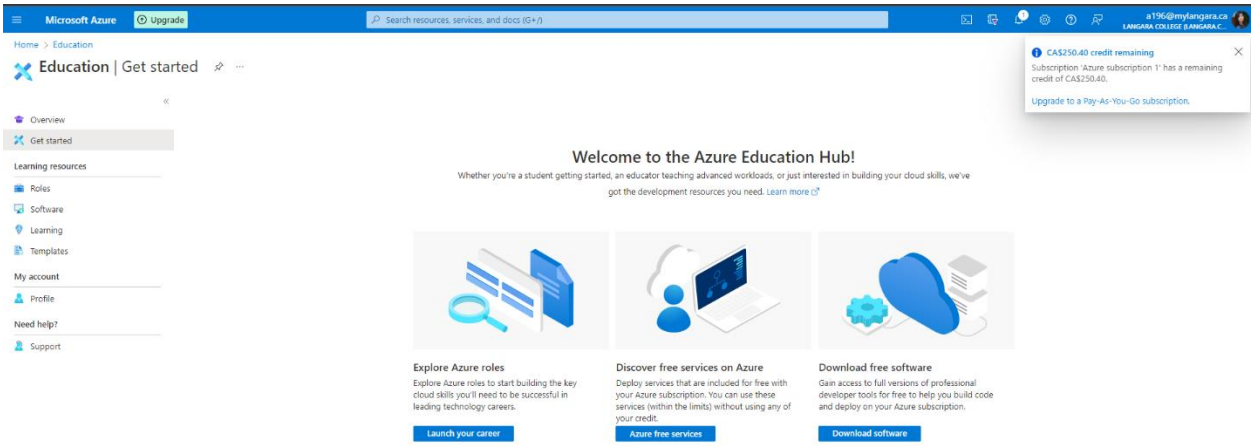
Step 1: Go to Azure for Students page on Azure



Step 2: Click on Start Free and Accept the terms & conditions.

If you are logged into your School Account, you will automatically be logged in. If not, login using your school account to Sign Up.

Once signed in, you will land up on Education page of Azure Portal.



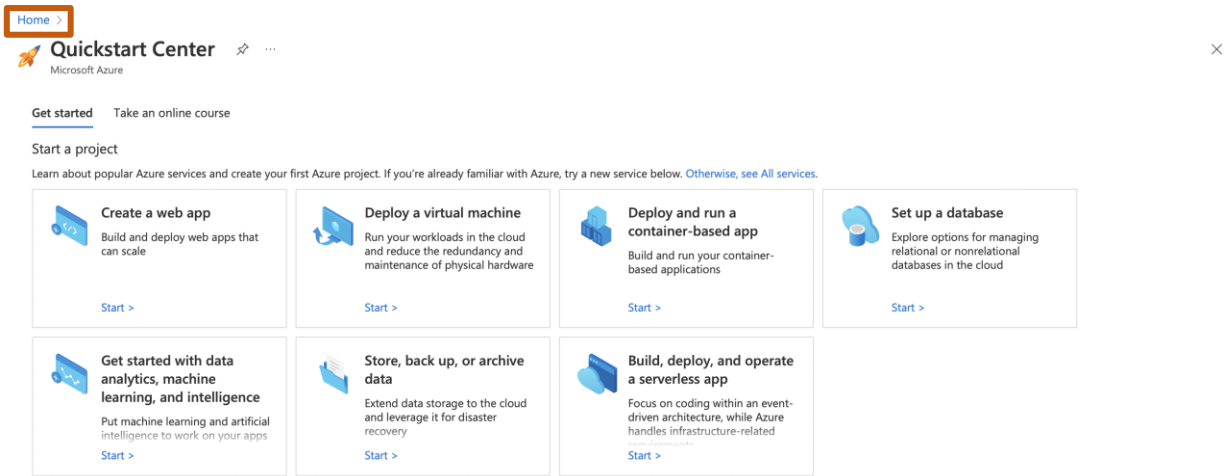
Explore this page to get the most out of your students' subscription.

Machine Learning Service

The Azure Platform has a variety of services to perform various tasks. In this tutorial, we will explore the Machine Learning service also known as AML (Azure Machine Learning).
AML is an extensive service that includes different methods of developing data preprocessing, data analysis, and machine learning pipelines. It also includes integration of Compute platforms to deploy the trained models and utilize them as an API or Web Service.

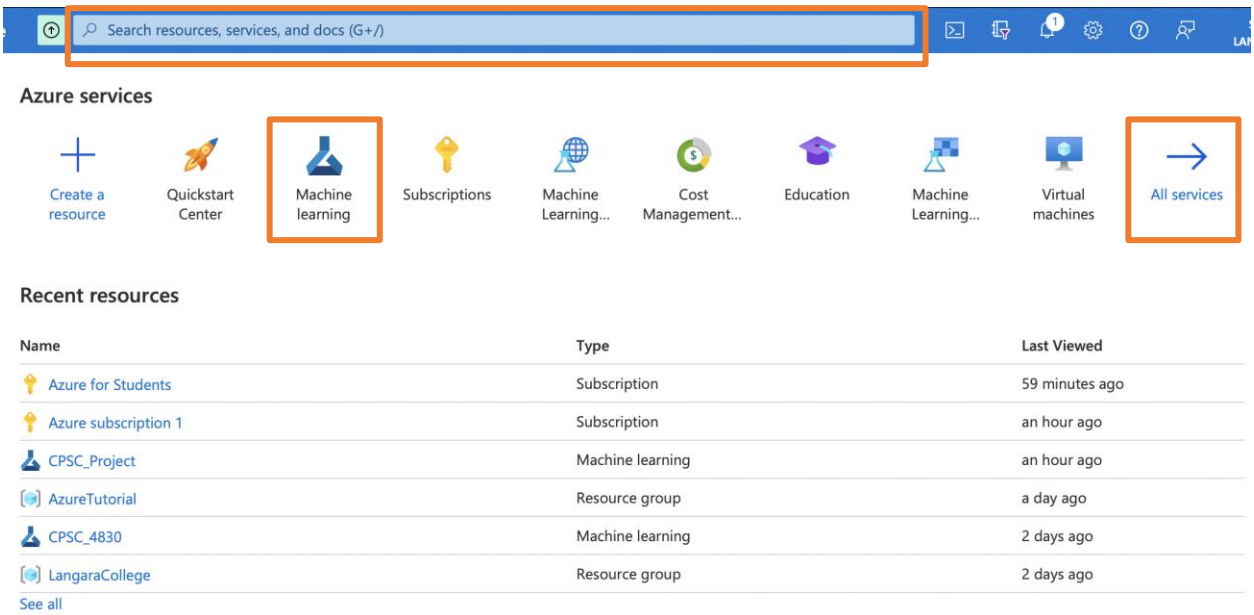
Let's open the AML service to get started. From the previous section, you are now logged on to your Azure Portal.

If you are logging in for first time, you may land up on the QuickStart Center.



Here you can check out various projects to work on for developing web application, VMs, setting up databases etc.

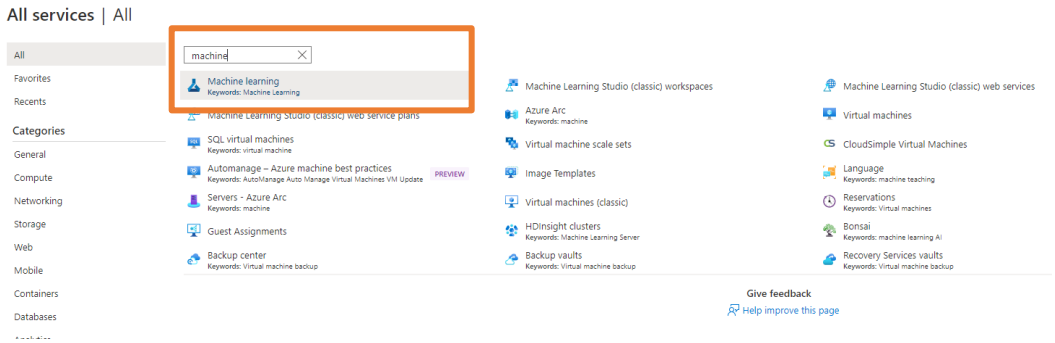
Click on Home from the top right corner to arrive at Homepage of Azure Portal.



Now, to open AML, you can:

- Search for Machine Learning from Search Bar at the top.
- Click on Machine Learning icon if already being shown

- Click on All Services and scroll in the list of all available services, and select Machine Learning

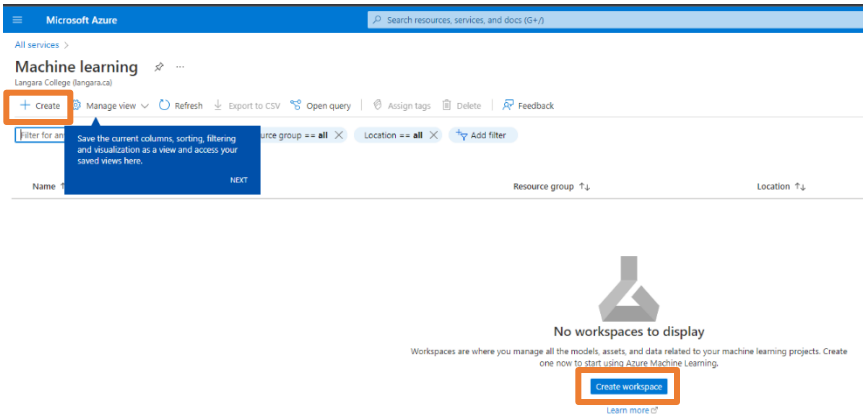


Machine Learning service requires user to create a Workspace to work in. This workspace helps in keeping different domains of an enterprise separate when dealing with machine learning applications across different segments of an organization.

If you already have a workspace, skip to “Selecting the Correct Workspace” section.

Creating Workspace

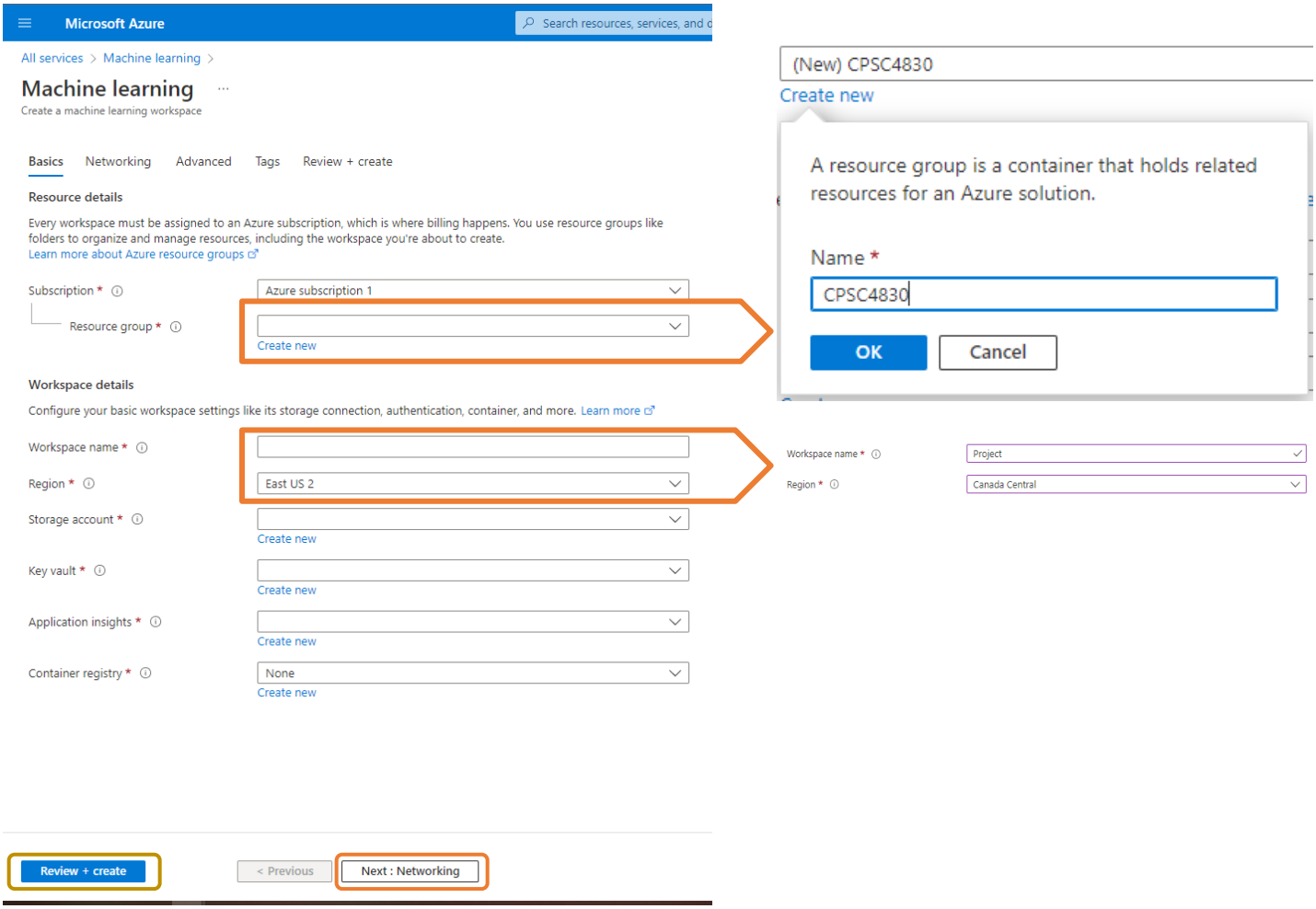
On the Machine Learning homepage, Click on Create Workspace.



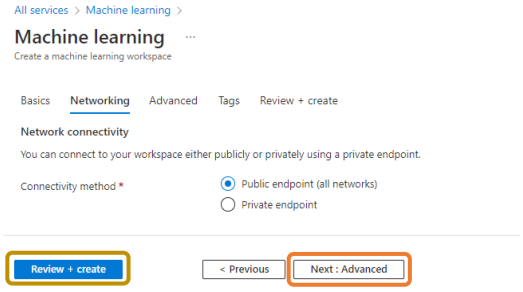
The workspace creation requires you to specify the Azure subscription you would like to use for the workspace. You can either use the default Azure Subscription you get when you create a free account or use your Azure for Students subscription.
For this tutorial, you need to select Azure for Students subscription. (If you do not have an Azure for Students subscription, go to this section to get one.)

- Select Azure for Students subscription.

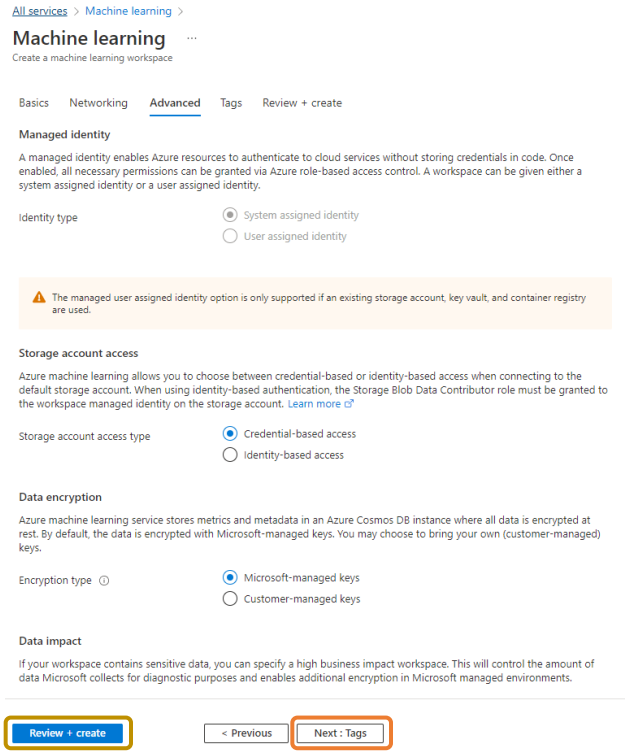
- Provide a Resource Group by clicking on “Create New”
- Give a relevant name to your workspace.
- Select a Region. Region selection is crucial to acquiring VMs and Clusters. Make sure to use US or Canada regions only.
- Everything else gets created automatically. Click on Next: Networking. (You can skip additional configurations by clicking on Review + create)



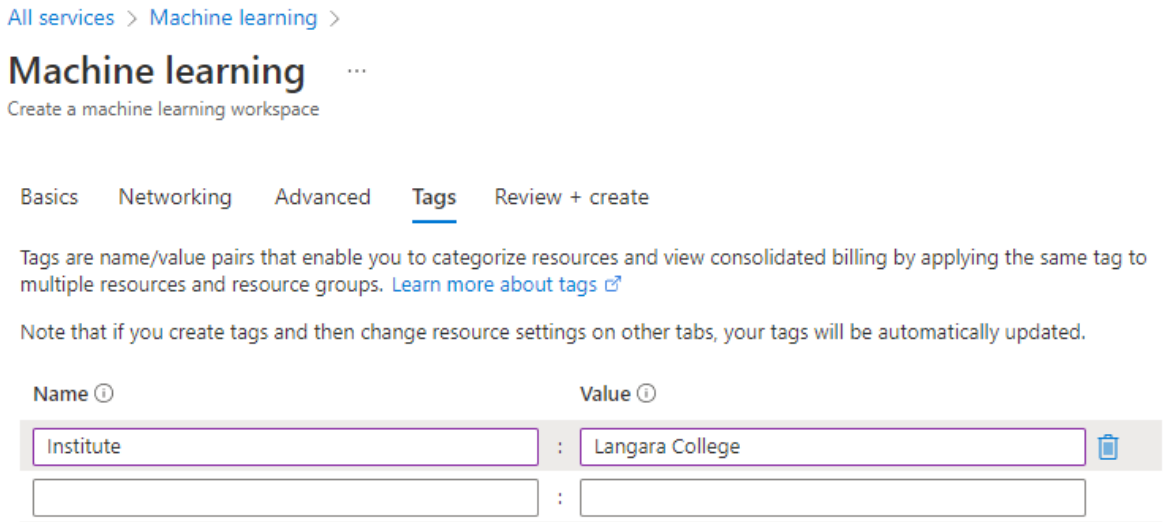
Keep the default setting of Public Network Selected. Click on Next: Advanced



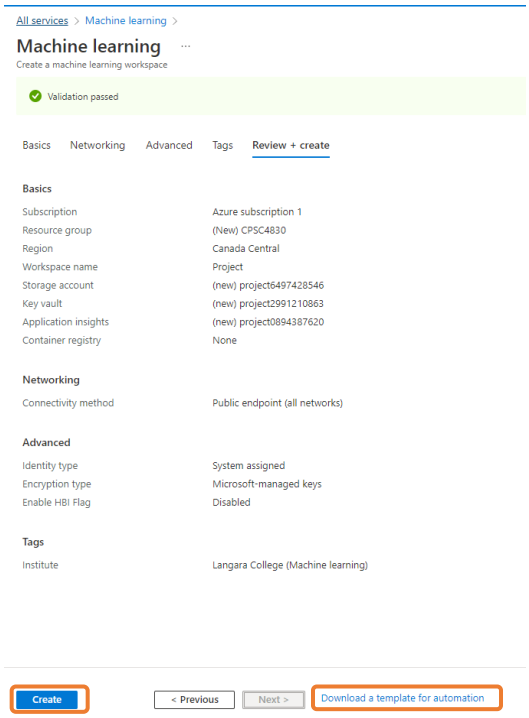
You can configure advanced settings related to Access, Encryption etc., to make your workspace more secure. For this tutorial, we leave the default settings and click on Next: Tags.



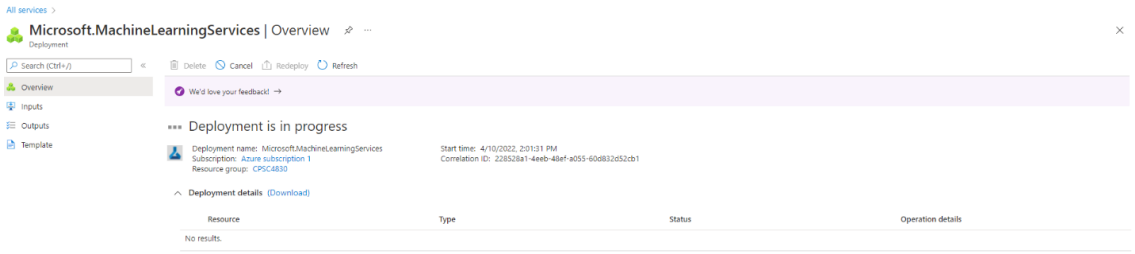
Tags are used to identify properties of a Workspace defined as metadata. These are helpful to identify, search and utilize the workspace. For example, we have provided the tag of the college to identify this workspace as belonging to Langara College tasks.



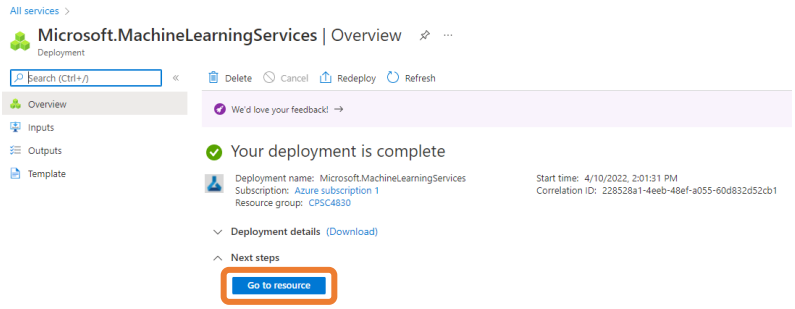
Finally, Click on Review + Create and then Create to create your workspace. You can download a template of settings configured in the workspace dialog. This can be utilized to automate the process for multiple workspace creation via code.



The workspace creation involves deployment of various components such as resource, registry, containers etc. After clicking create, the deployment of these resources will begin. Wait for process to complete.

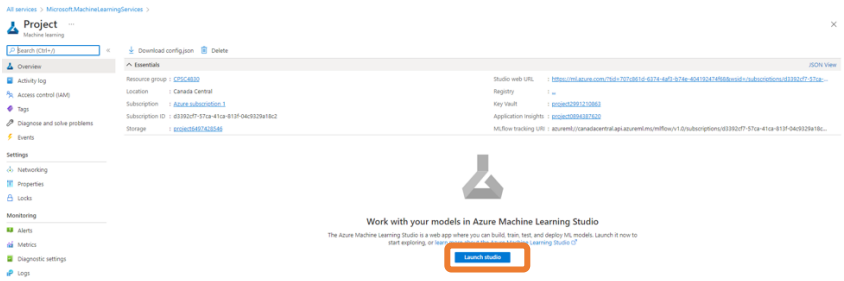


Once Deployment is completed and successful, click on “Go to Resource” to open your machine learning workspace.

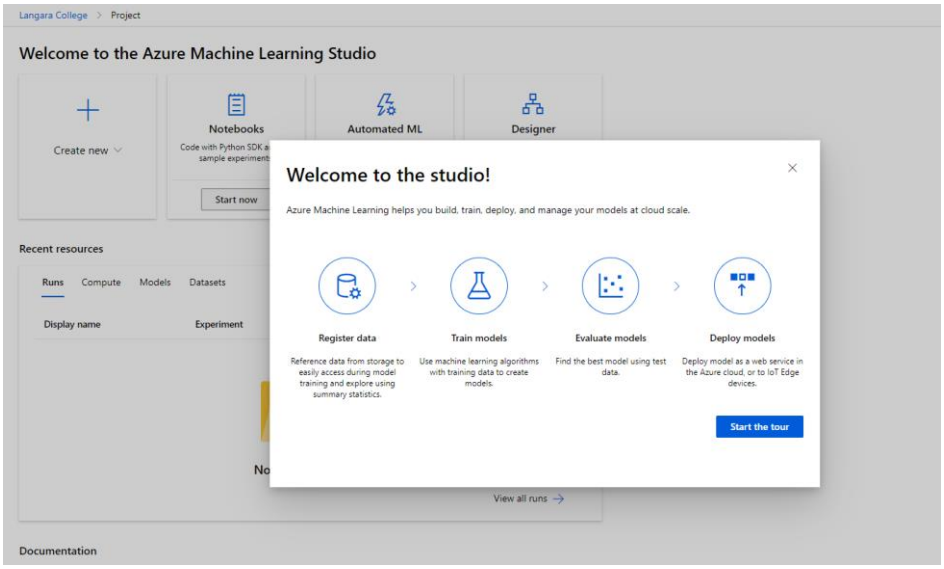


Machine Learning workspace has a variety of options to configure and control. Explore these options to find out other additional advanced configuration as per use case.

Click on Launch Studio to launch Machine Learning Studio in the workspace.



Congratulations! you have successfully configured a Machine Learning workspace.
Now let’s do the action in the studio.

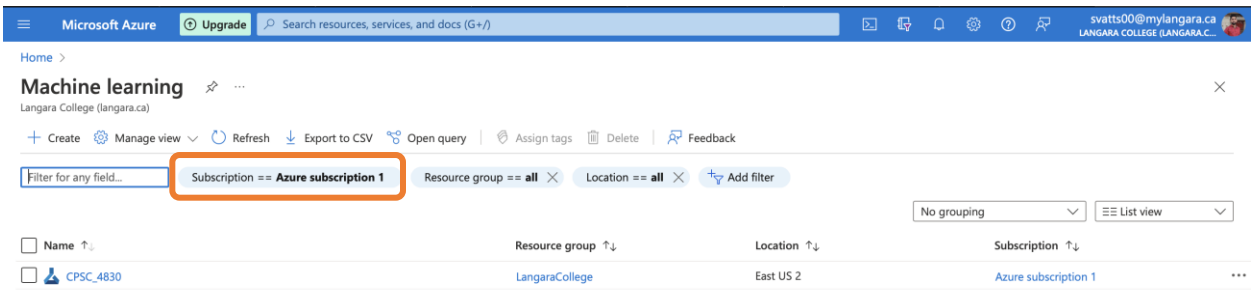


Machine Learning Studio

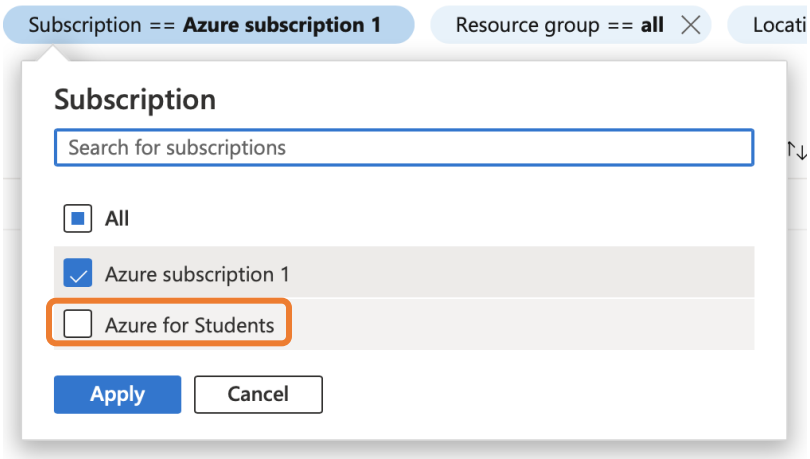
Selecting the Correct Workspace

If you have more than one subscription, make sure to select resource under the Azure for Students subscription for this tutorial. If it is not visible on Machine Learning service, select “All” in subscription filter to select the correct workspace

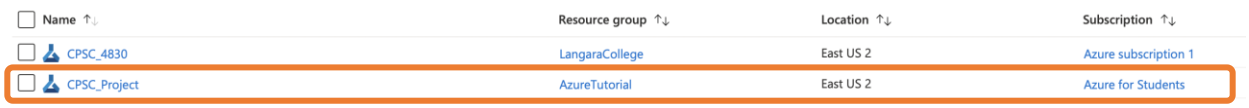
Step 1: Go to Machine Learning Service.



Step 2: Click on Subscription filter and select Azure for Students subscription.



Step 3: Select the Machine Learning Workspace under Azure for Students.



Compute Targets

A compute target is a designated compute resource or environment where we can train our machine learning models and then deploy them. In AML Studio, we can create four kinds of compute resources:

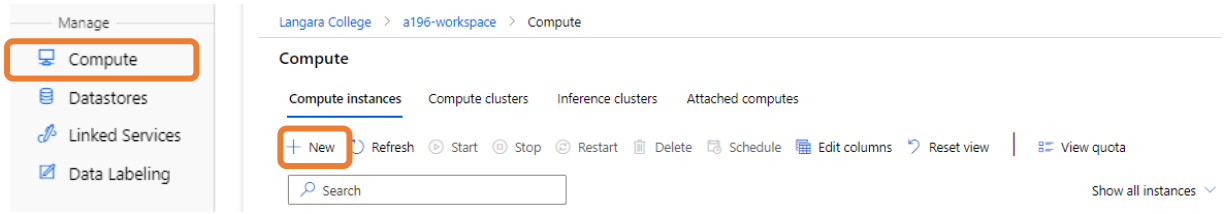
- **Compute Instances**: Development workstations that data scientists can use to work with data and models.
- **Compute Clusters**: Scalable clusters of virtual machines for on-demand processing of experiment code.
- **Inference Clusters**: Deployment targets for predictive services that use your trained models.
- **Attached Compute**: Links to existing Azure compute resources, such as Virtual Machines or Azure Databricks clusters.

In this tutorial we'll explain how to create two of them which we'll be using – Compute Instances and Clusters.

Compute Instance

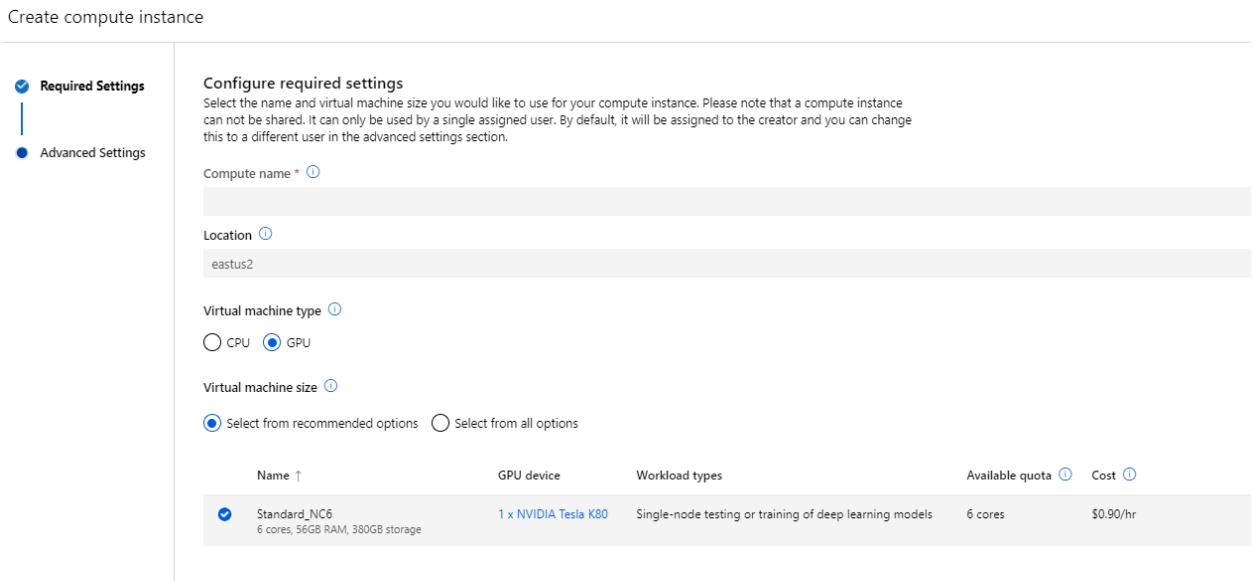
We'll create a GPU enabled instance for training our models. Go to Azure Machine Learning Student workspace to get started.

Step 1: Click on Compute under Manage section from the side Menu.



Step 2: Under Compute Instances tab, click on New to create a new compute instance

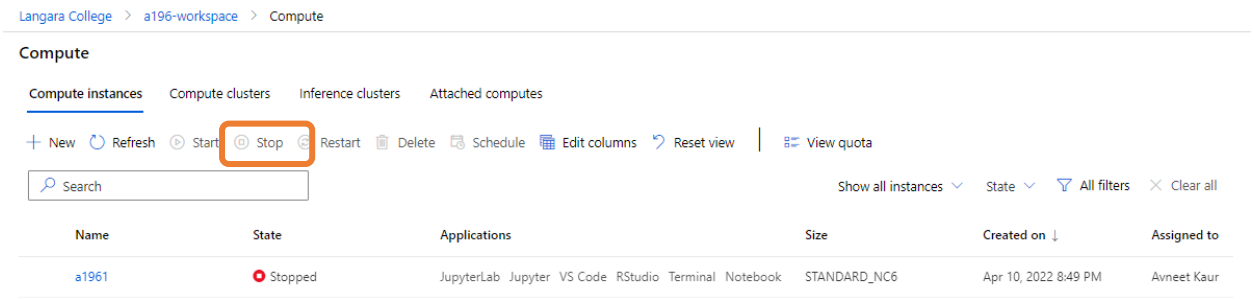
Step 3: Enter a compute instance name, location and VM type as shown below:



Step 4: Select a GPU based compute instance as shown above for training Deep neural networks. Since the subscription allows up to 6 cores, make sure to select an appropriate instance. You can select CPU based compute instance for normal Machine Learning tasks.

Step 5: Click Next to configure any Advanced Settings if needed. Finally click Create.

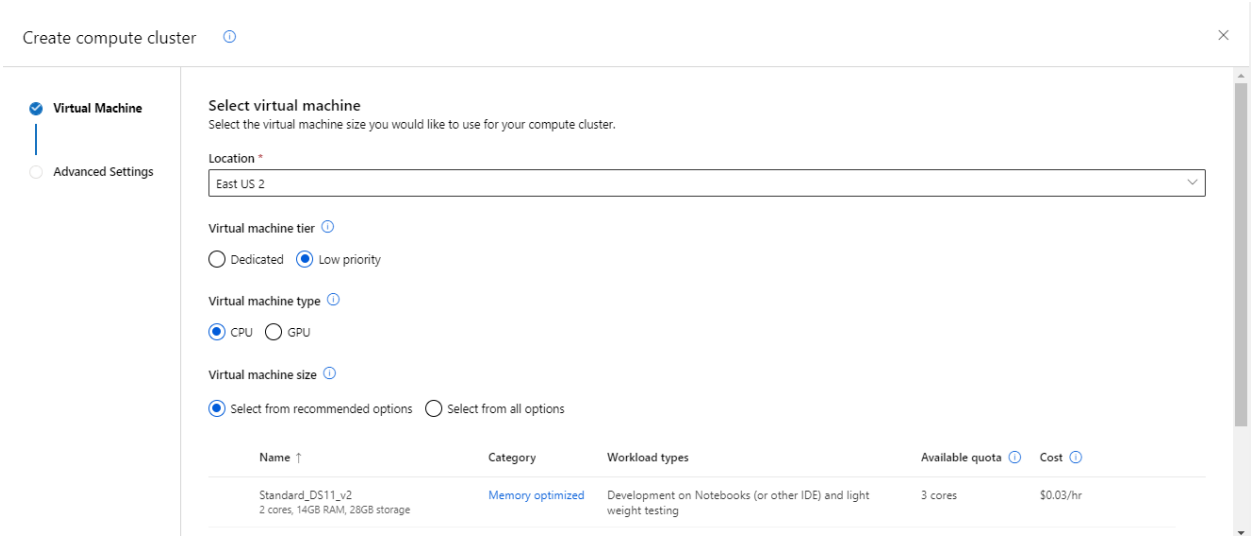
The instance creation takes about 2-5 minutes. Once created, the instance will start Running automatically. Make sure the instance is in Stopped state until and unless we are running the training of the model.



Compute Cluster

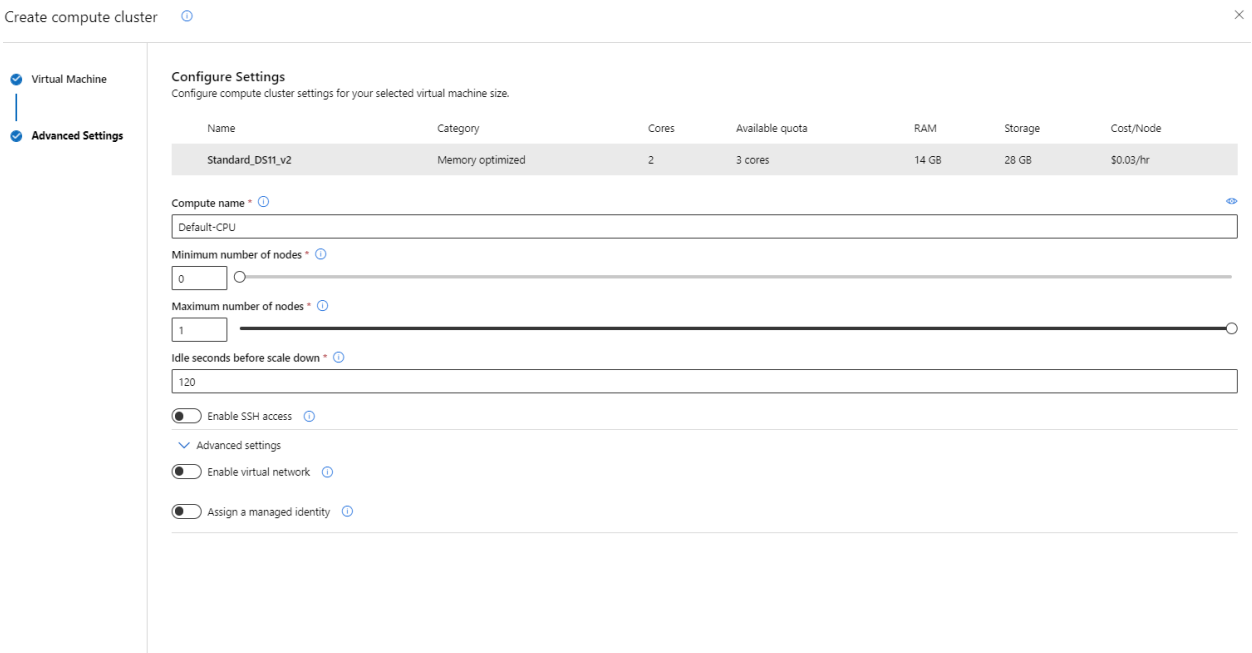
Step 1: Click on New under Compute clusters tab.

Step 2: Make the following selections according to the requirements as shown below:

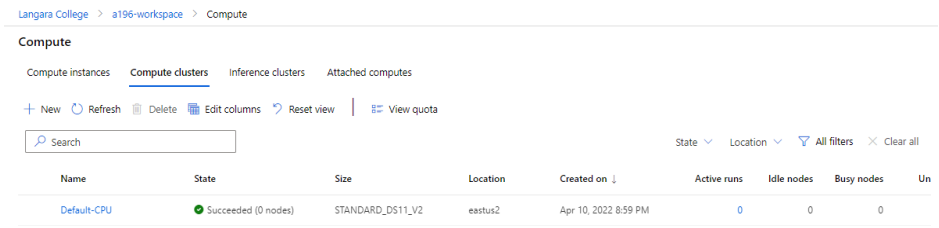


Step 3: Click on Next to proceed to Advanced Settings.

Step 4: Fill in the details like Compute Name, Min/Max nodes as shown below:



Step 5: Click on Create.



If you are using Low Priority nodes in the cluster which are provisioned while running, you do not require to explicitly stop the clusters. They are scalable and on demand.

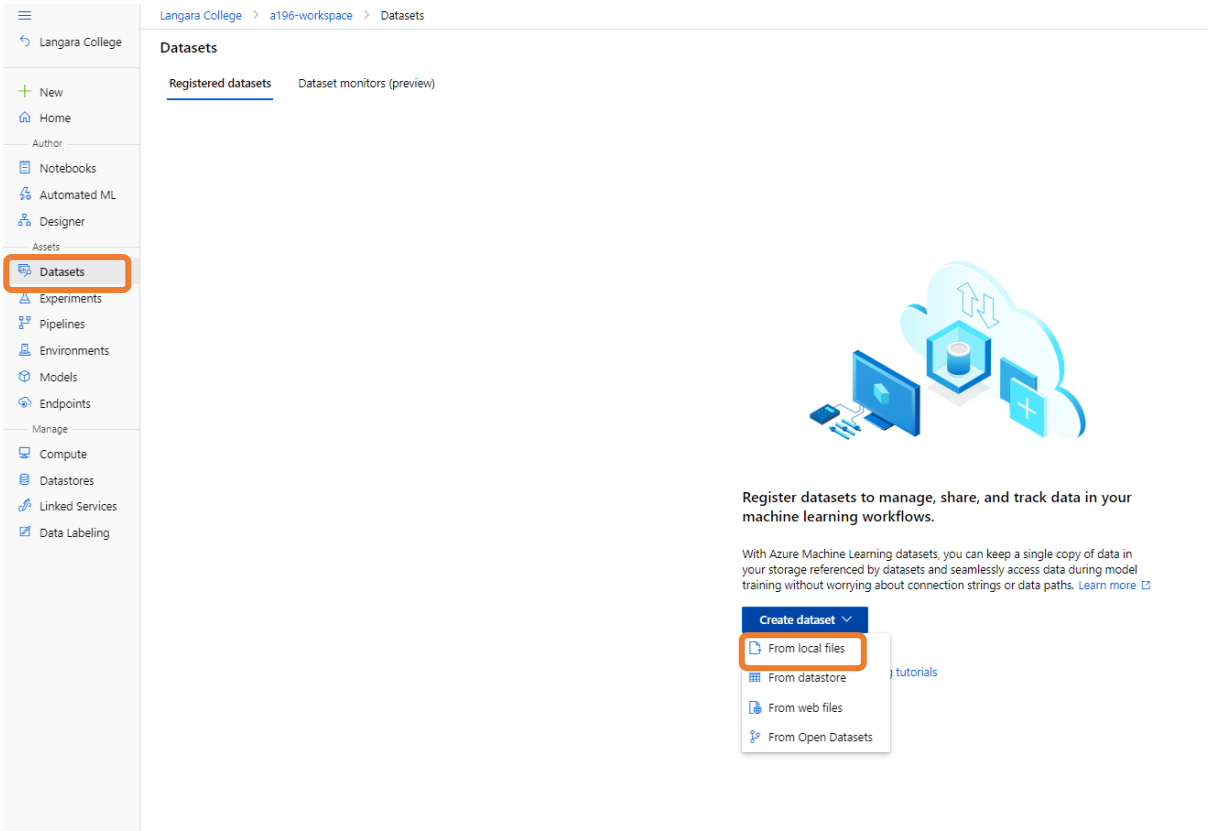
Well Done! You have successfully set up the compute infrastructure needed for your Machine Learning Tasks.

AML Designer

In this section, we'll be training a deep neural network using PyTorch for Multiclass Image Classification of Simpson's characters. The dataset is publicly available on Kaggle. Since it's a huge data, for our tutorial we only select 5 characters - apu_nahasapeemapetilon, comic_book_guy, edna_krabappel, kent_brockman, nelson_muntz.

Dataset Creation

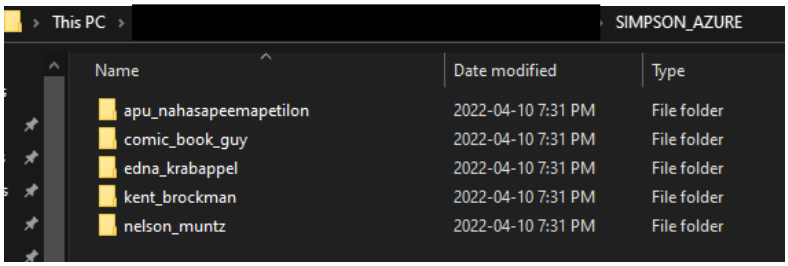
Step 1: Go to Datasets under Assets Section and click on Create Dataset.



Step 2: As shown above we have 4 ways of uploading the datasets – we select **Local Files** as we are using our own custom dataset. You can import the datasets from datastore (datasets on Azure Cloud), web files (using URL of the source dataset), or use Open Datasets available in the tabular format on Azure Cloud

Some important tips for preparing data:

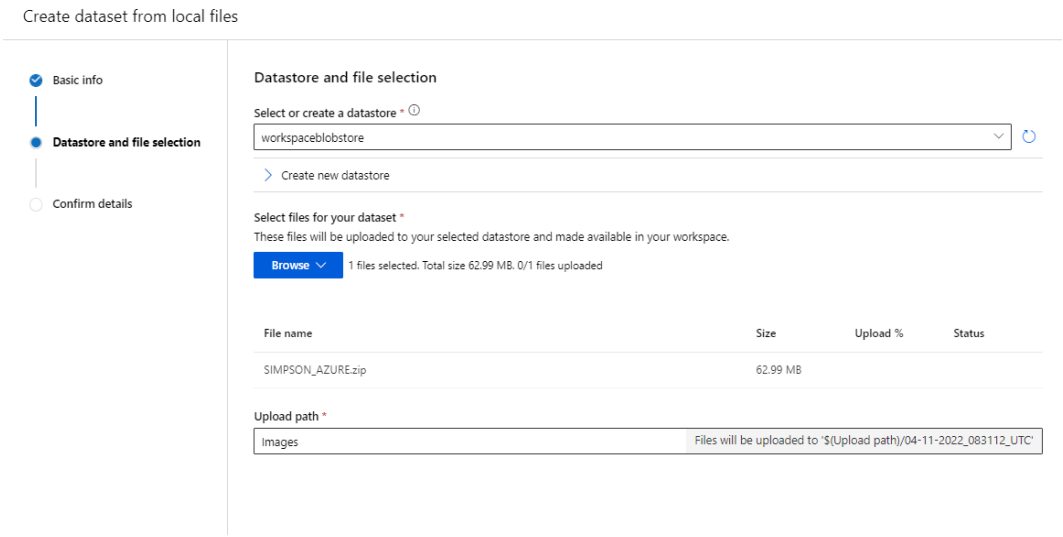
For image datasets, we need to upload the files in a particular format. In your local system prepare your dataset in such a way that inside the main folder, each subfolder represents a label/class that contains images of the corresponding class as shown below:



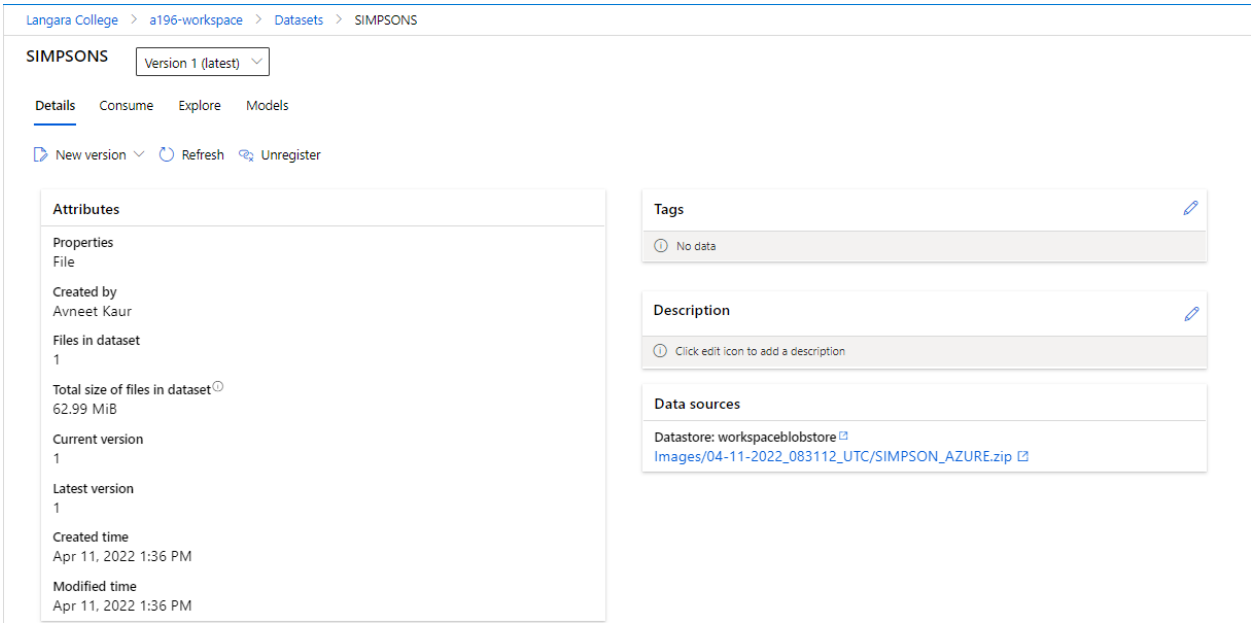
The number of images in each class can vary. Now, zip the parent folder (SIMPSON_AZURE).

Step 3: Fill in the basic information as shown below and click on Next:

Step 4: Select the default datastore. For selecting files, Click Browse and upload the zipped folder. Give any name for Upload path and click Next.



Step 5: Confirm the details and click create. Once created, you can open it to view Details.

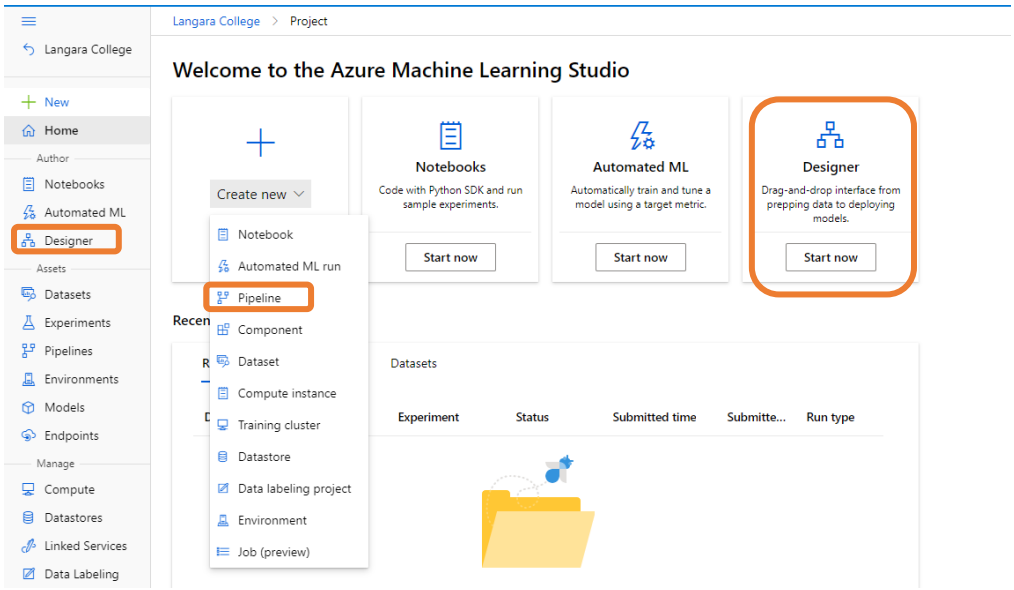


Kudos! Our dataset is ready, and we can finally begin with creating our Pipeline.

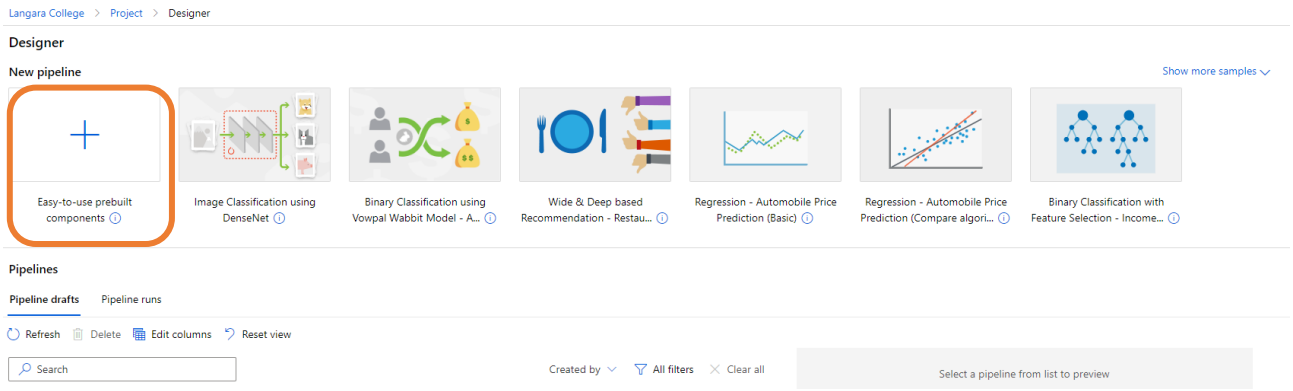
Machine Learning Pipeline

Azure provides an effortless no code interface to create a complete training as well as inference pipelines in no time. We will start with creating a training pipeline for our image dataset.

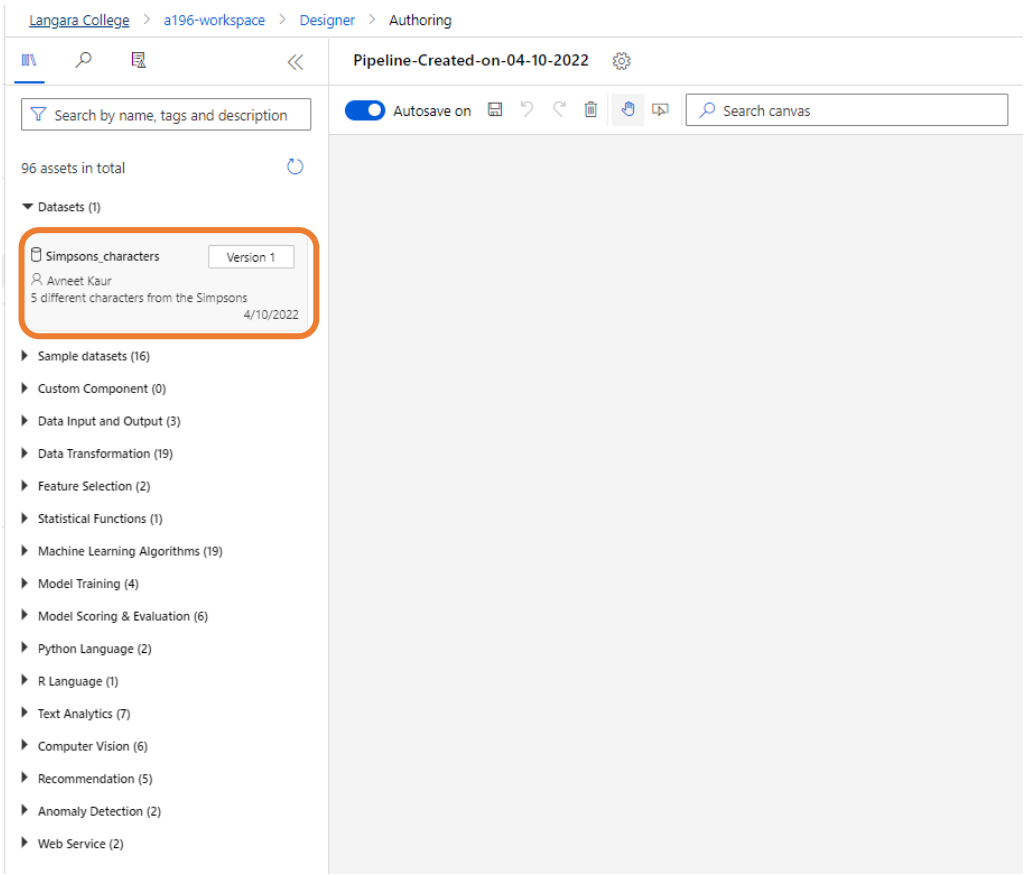
Step 1: From you ML studio you can go to designer in 3 ways as shown:



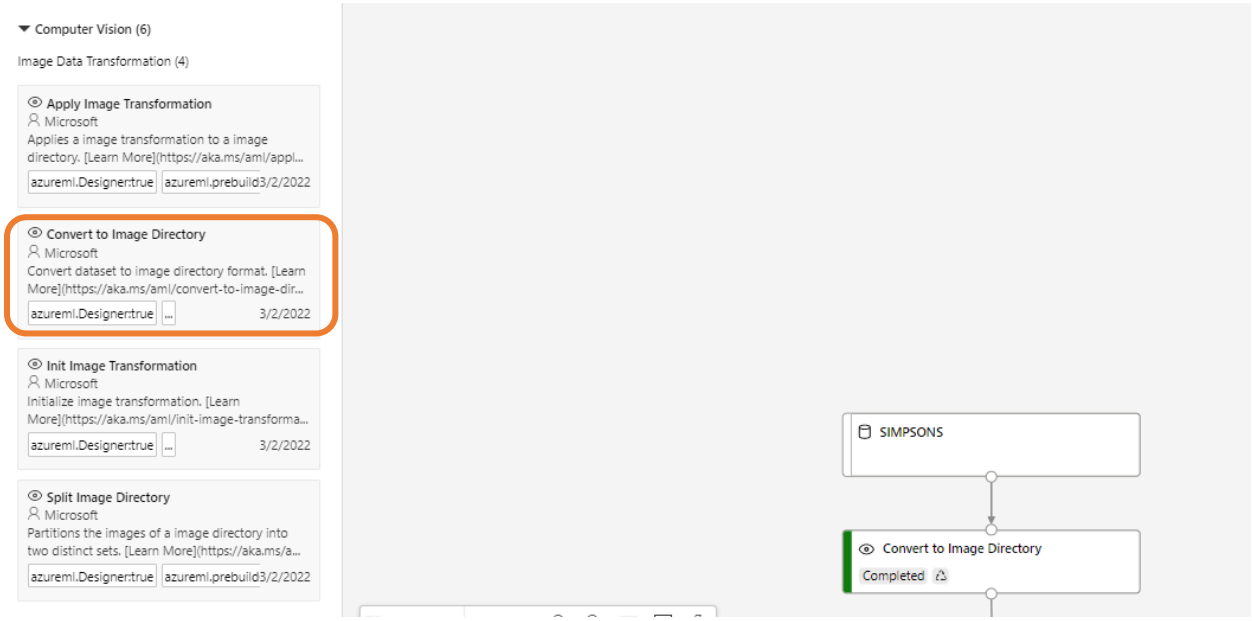
Step 2: You can explore sample pipelines provided by Azure for reference. To create your own, Select on “Easy-to-use prebuilt components”



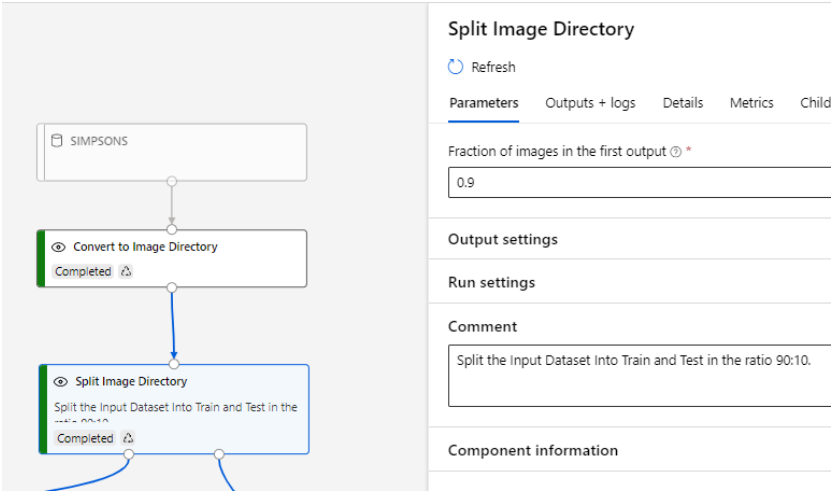
Step 3: You can explore various prebuilt operations related to different data analytics, machine learning, NLP, and Neural Network tasks. Let’s begin with loading the dataset. Drag and drop the dataset module on the canvas.



Step 4: To consume the zip file, we need to convert it into an image directory. Select Convert to Image Directory from *Computer Vision -> Image Data Transformation* and drag it to the canvas.



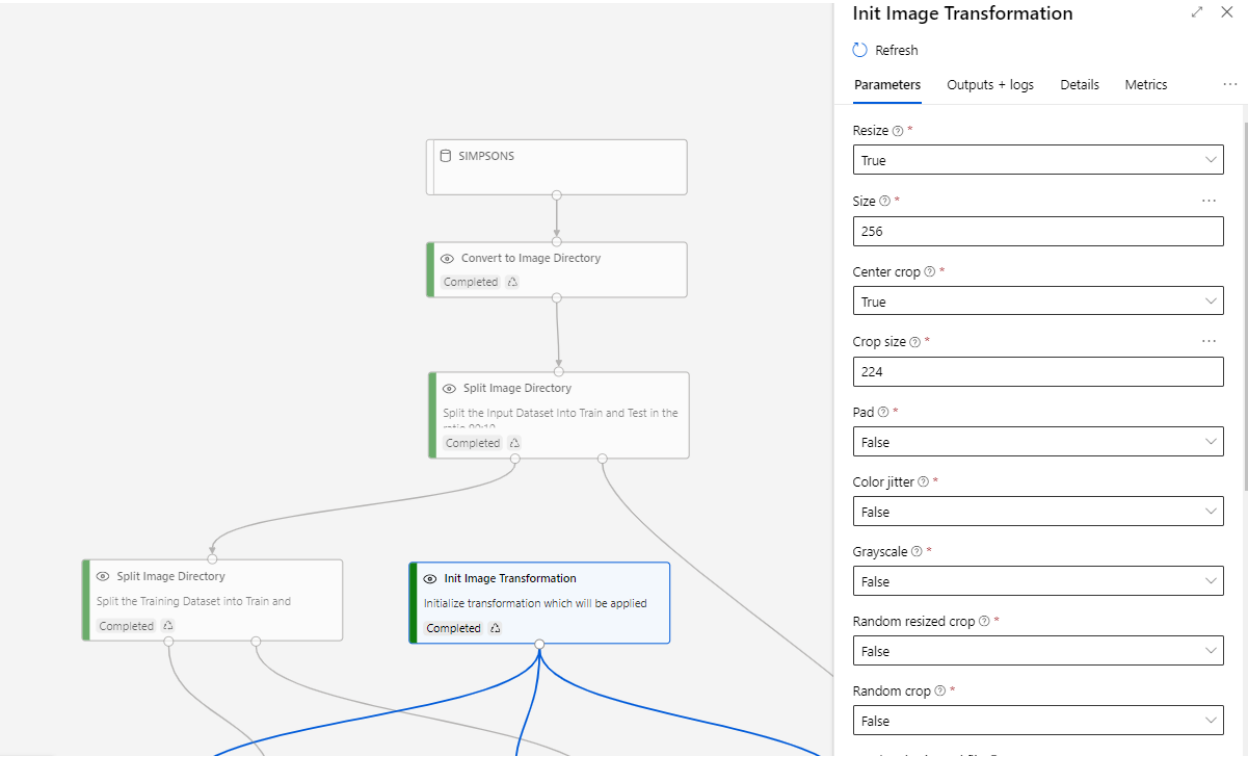
Step 5: Drag Split Image Directory on the canvas from *Computer vision -> Image Data* Transformation. Set the parameters as shown. This module splits the Input Dataset into Train and Test in the ratio 90:10.



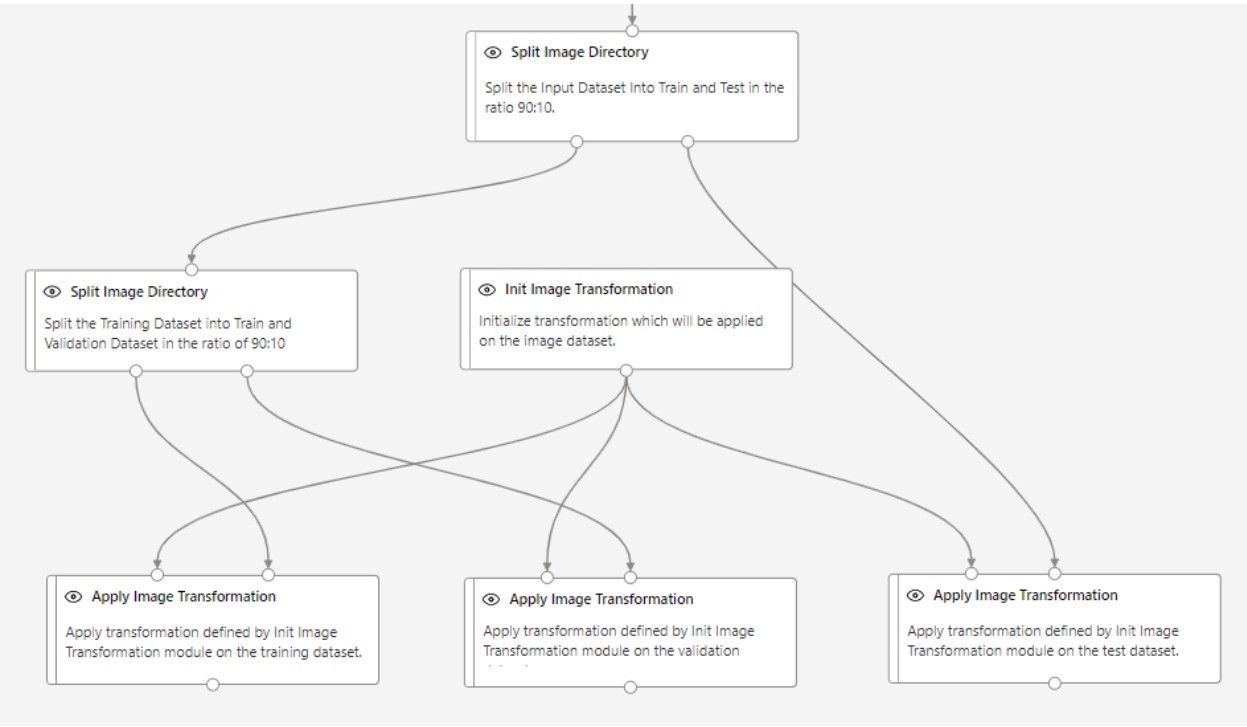
Step 6: Add another Split Image Directory module splits the Train Dataset into Train and Validation sets in the ratio 90:10.

NOTE: Use Docker = False in Run Settings of all modules

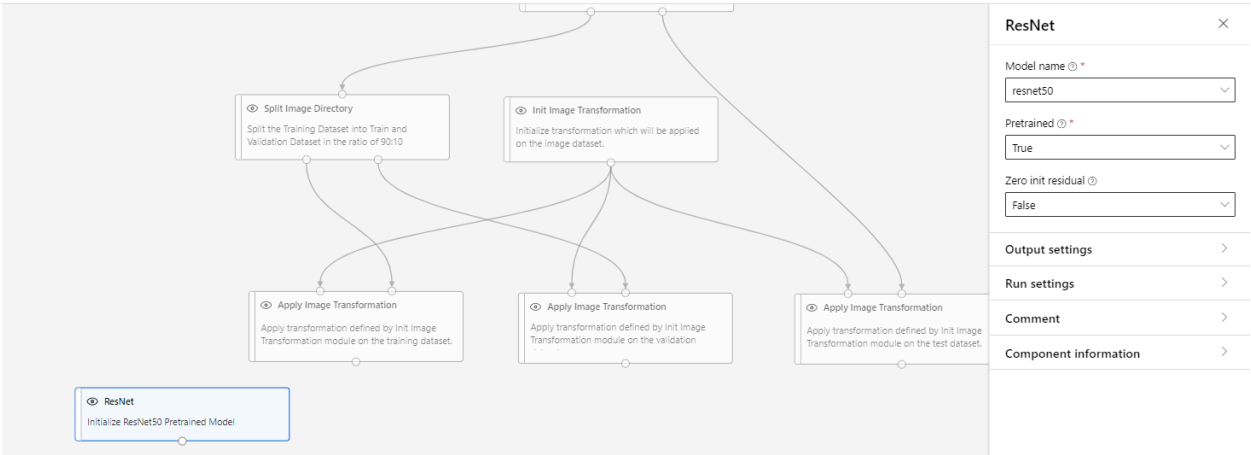
Step 7: Add Init Image Transformation component to initialize various image transformations such as Resizing, center cropping, random flipping etc.



Step 8: We need to apply these transformations to Train, Test and Validation datasets:

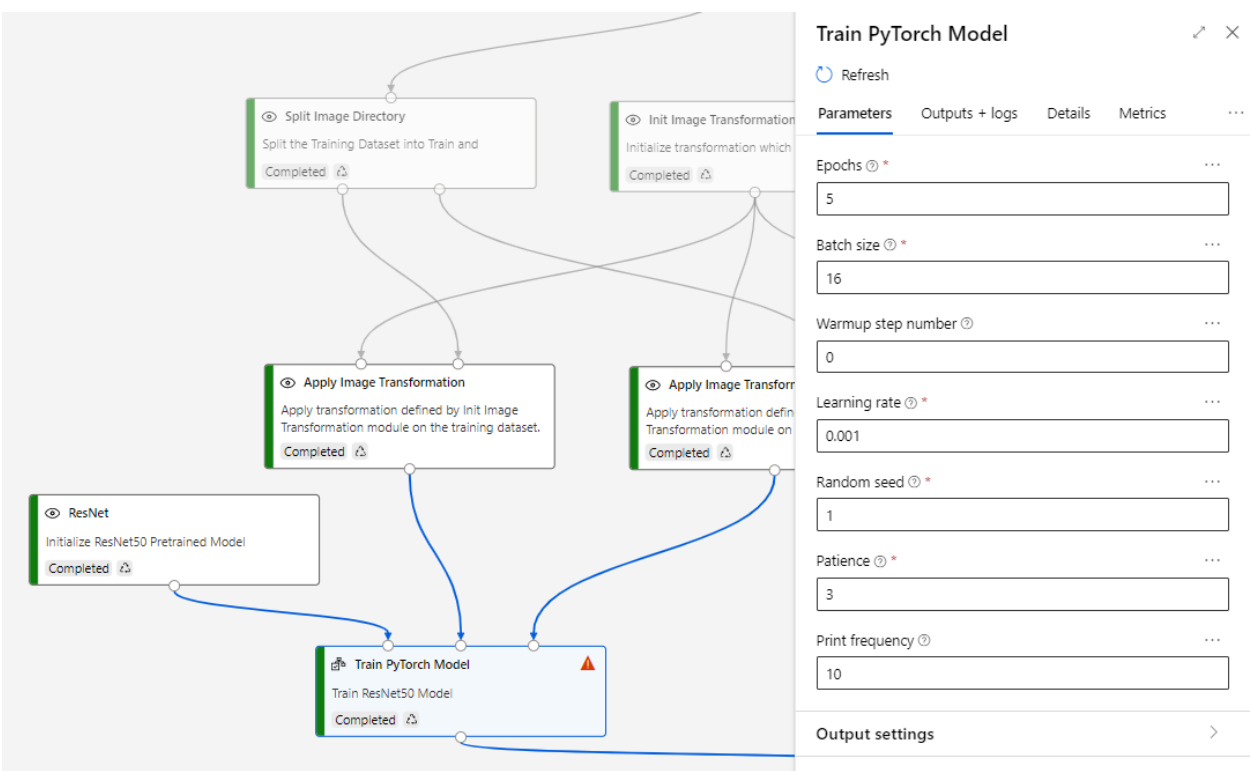


Step 9: We are using ResNet50 model for this problem. Grab the ResNet module from *Computer Vision -> Image Classification*



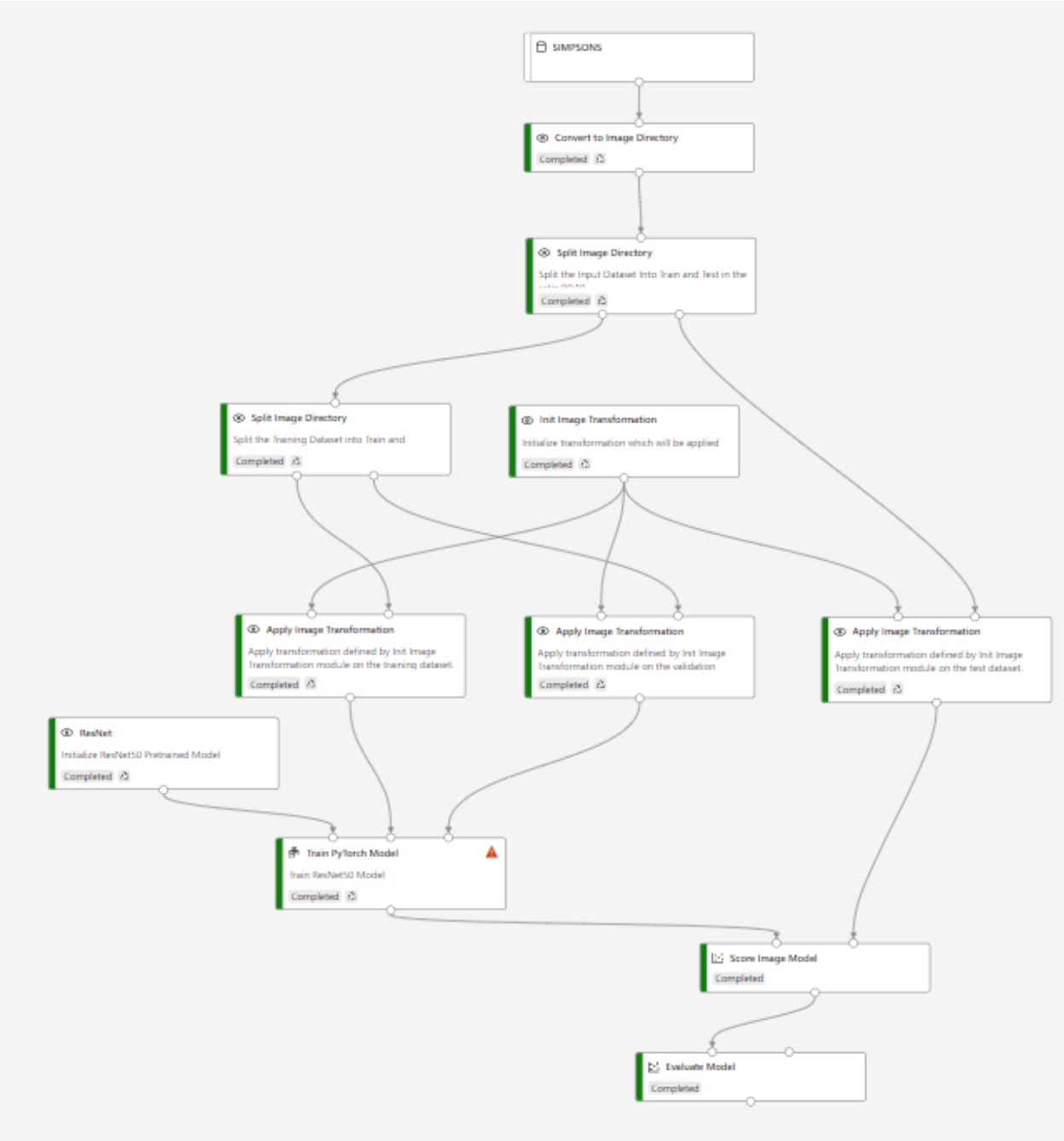
Step 10: Select Train PyTorch Model module (as ResNet module is developed in PyTorch on Azure) from Model Training and drop to the canvas. You can also tune hyper parameters from the parameter settings as shown. Make Sure to select GPU compute instance from Run Settings for only this module for faster processing.

**For all other modules select CPU or default compute instance*



Step 11: Finally add Score Model and Evaluate Model modules to capture model performance metrics.

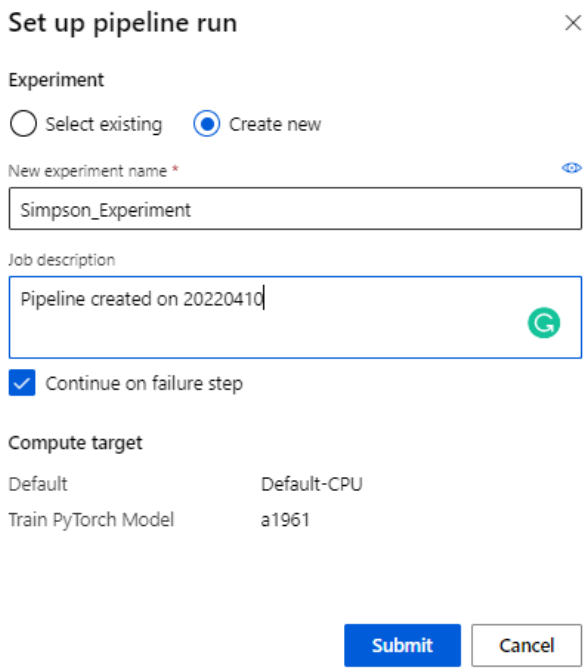
Final pipeline looks like as shown below:



Voila!! You have succesfully developed an image classification pipeline without writing a single line of code!

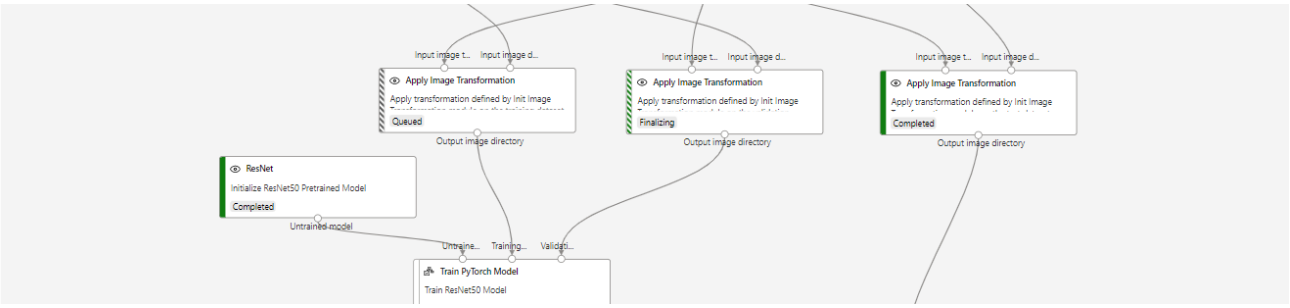
Run the pipeline

For running the pipeline just click on Submit button on the top right corner. Before submitting, make sure your compute instances are in Running state. On clicking Submit, the below popup occurs to setup the pipeline run.

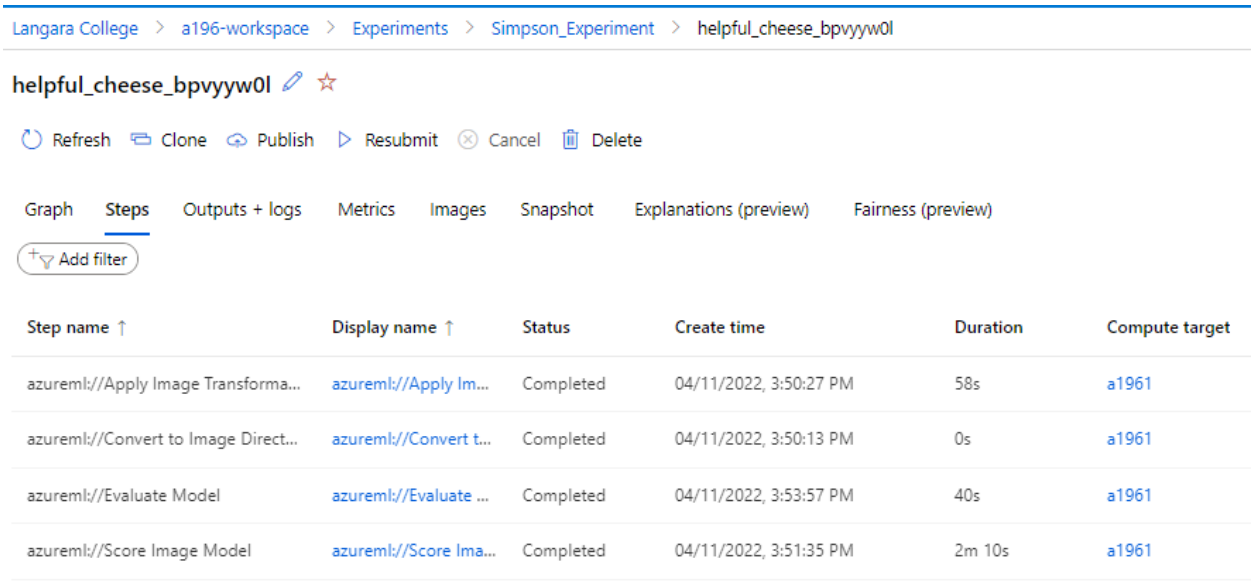


Experiment is a collection of trials used to validate a user's hypothesis. As we currently have no existing experiments, we create a new one, with the name Simpson_Experiment. Fill in the details as shown above and submit and wait for the experiment run to complete.

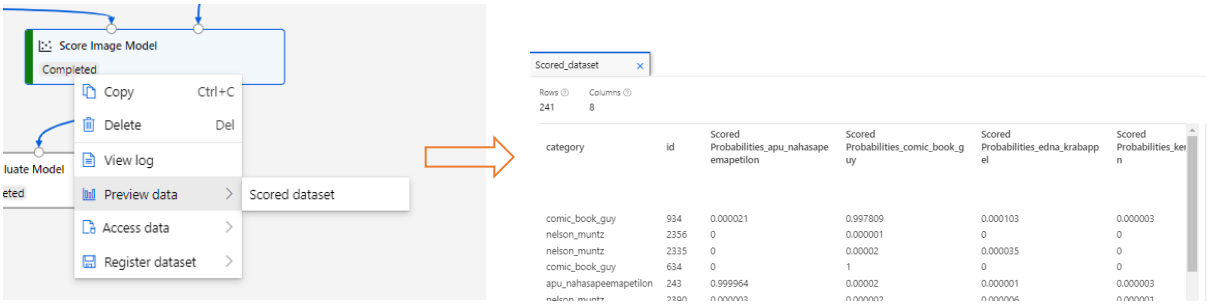
As the Experiment runs, we can view the modules transitioning from different states such as Queued, Running, Finalizing and Completed.

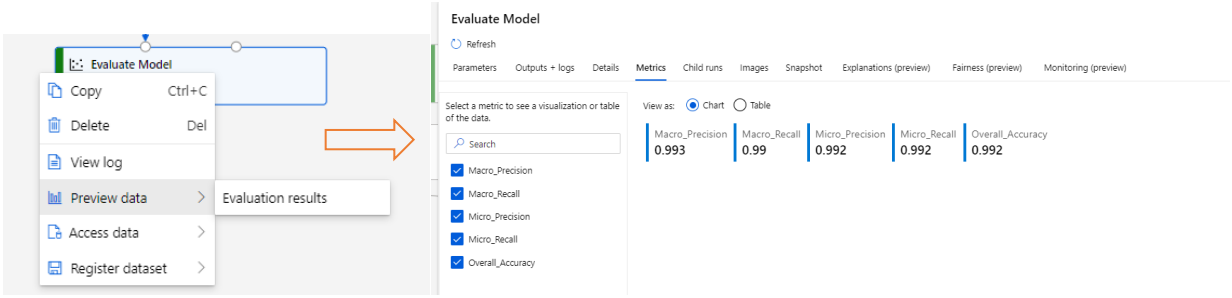


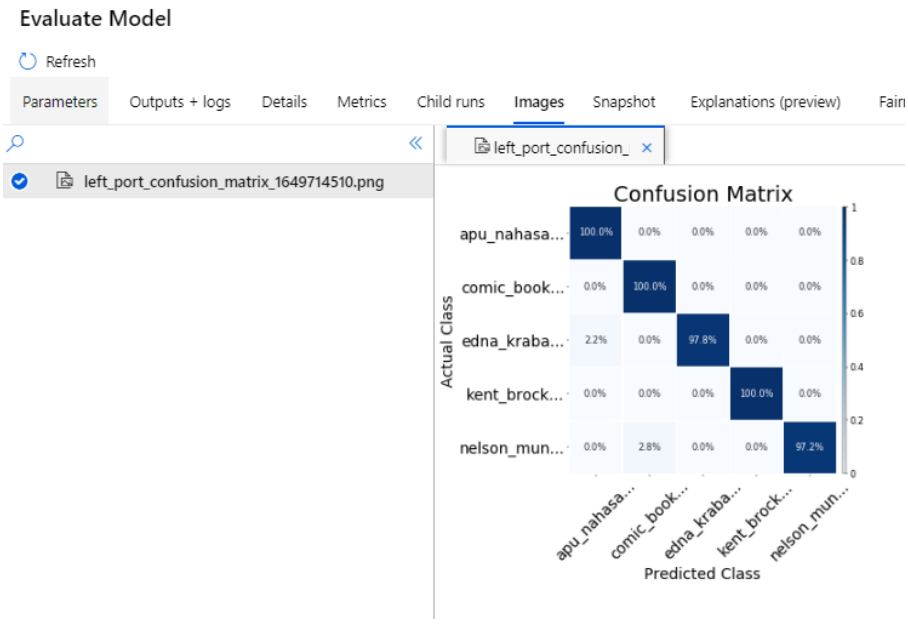
We can also navigate to *Pipelines -> Select the current Pipeline -> Steps* to view the status, duration etc. of each module in the pipeline run:



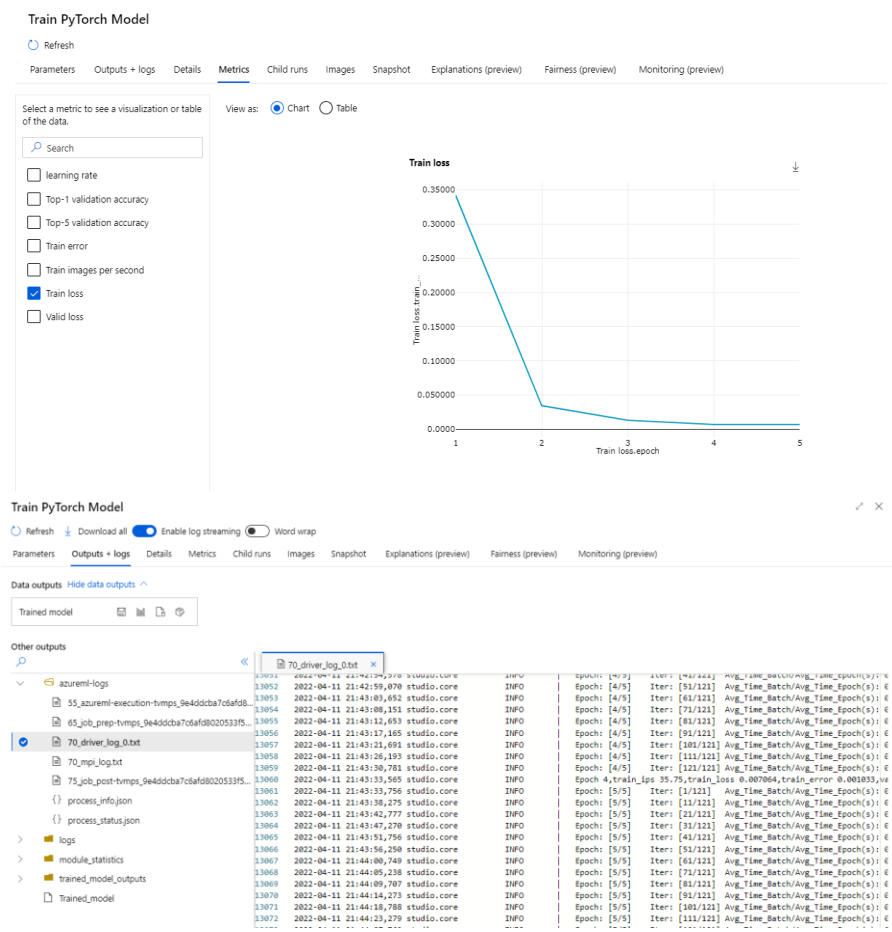
Once the pipeline run successfully completes, we can see the results through Score Image Model and Evaluate Model modules by clicking right and selecting following options:







We can also right click on the Train PyTorch Module to inspect logs and visualize performance metrics:

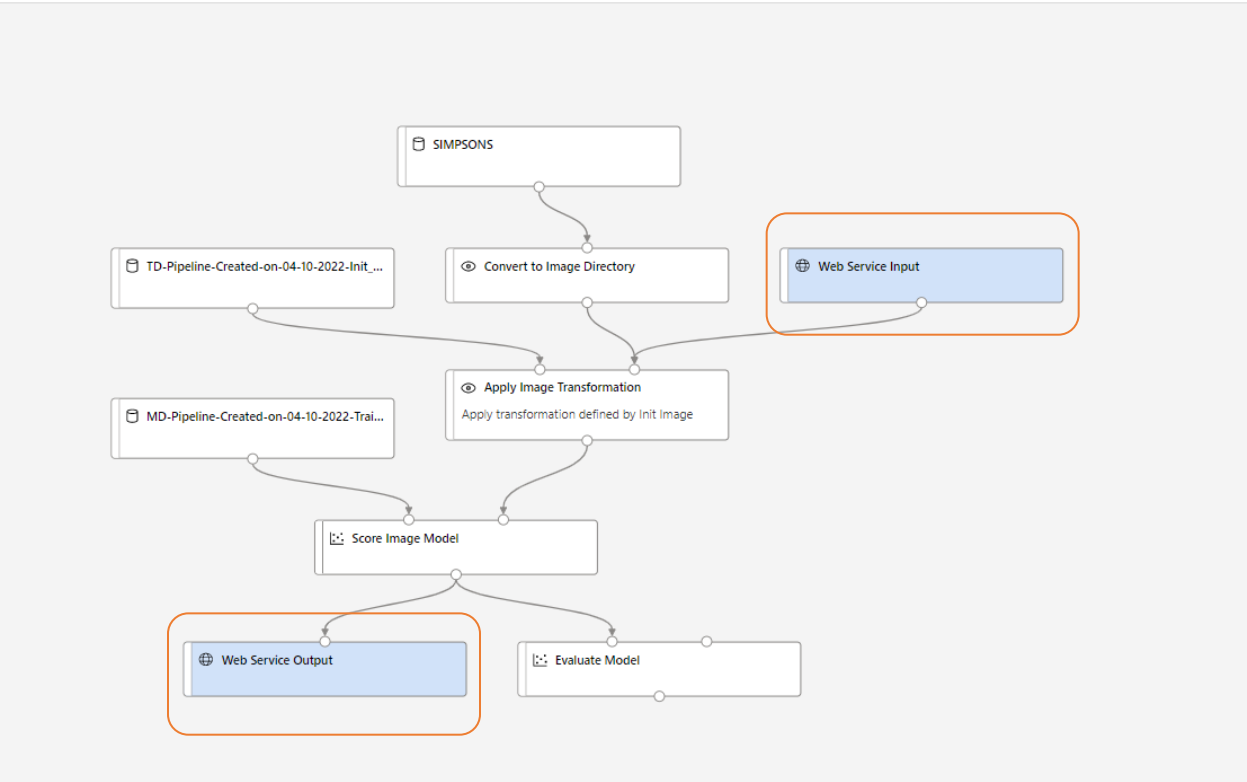


Deployment of the Pipeline

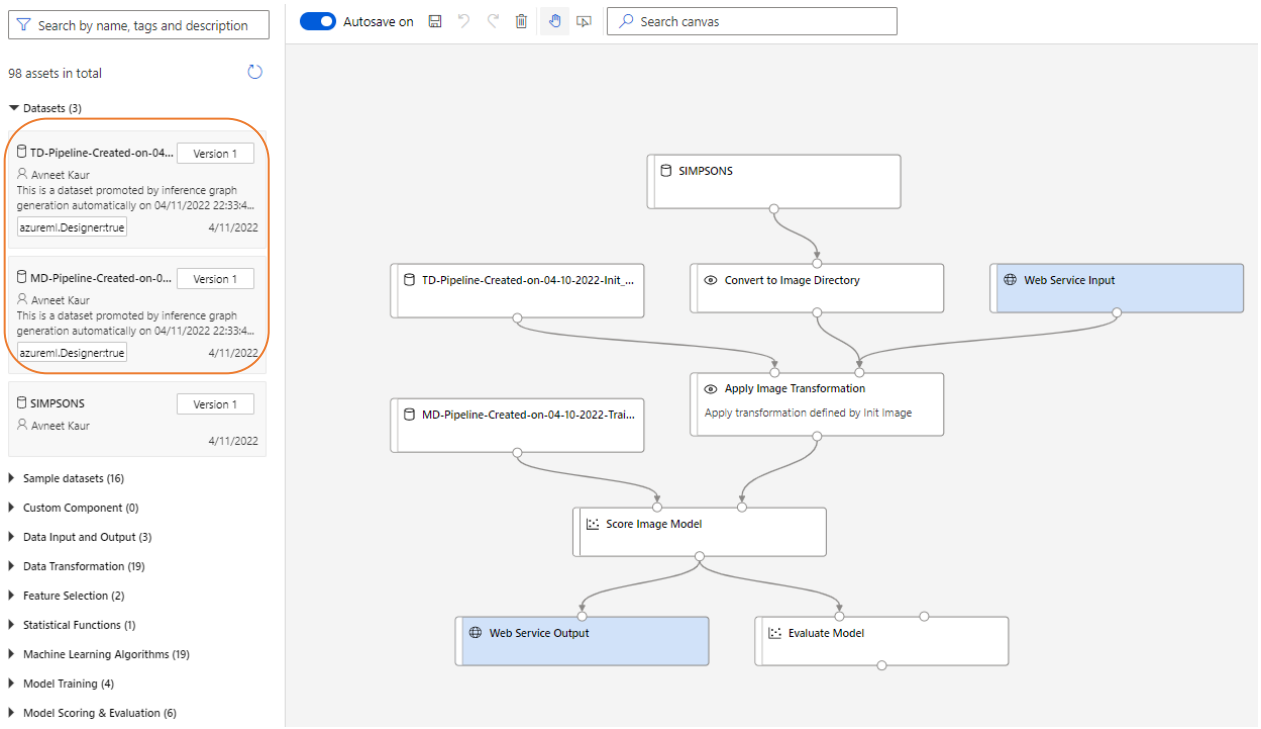
The model developed using the above pipeline (Training pipeline) gets registered on Azure portal. It is saved as a pickle(.pkl) file on the cloud so that it can be used for carrying out predictions. The deployment is effectively using the saved model through an access point called as an 'Endpoint' in Azure ML environment. An endpoint provides a way to access the model through internet via APIs.

Step 1: To deploy the pipeline, we need to convert training pipeline into a real time inference pipeline. Click on Create Inference pipeline and select Real time Inference pipeline.

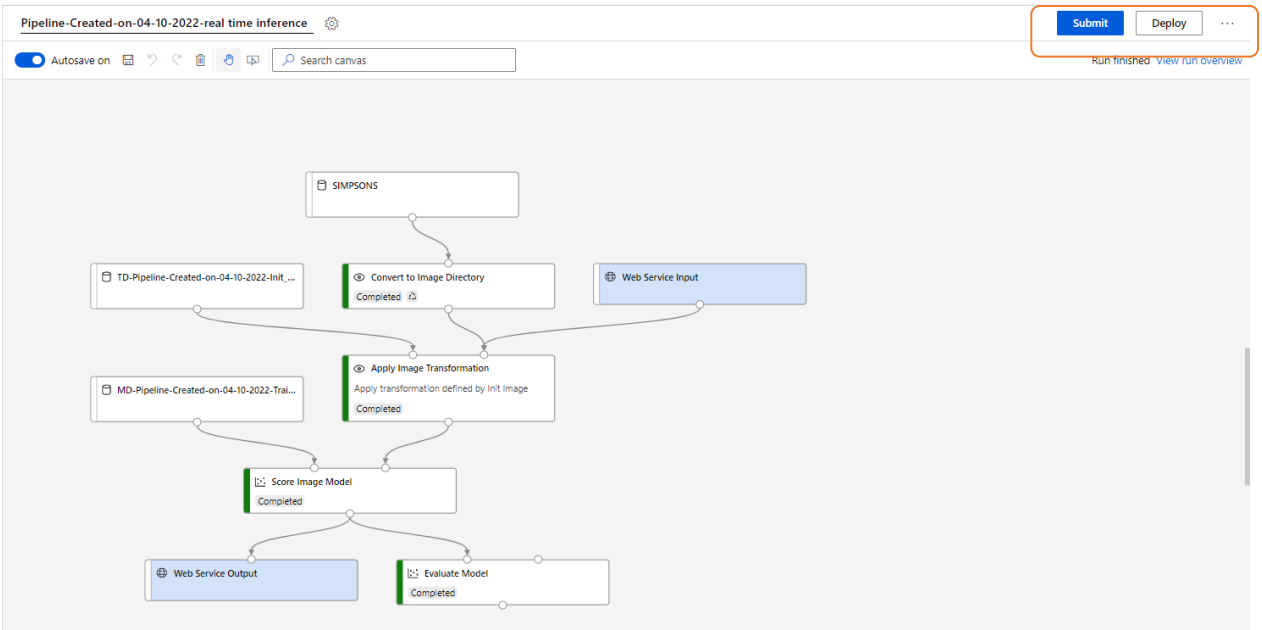
Step 2: The below pipeline gets automatically created. The Web Service Input provides a way for end users to submit the raw data to the model. Web Service Output gives out predictions based on input data.



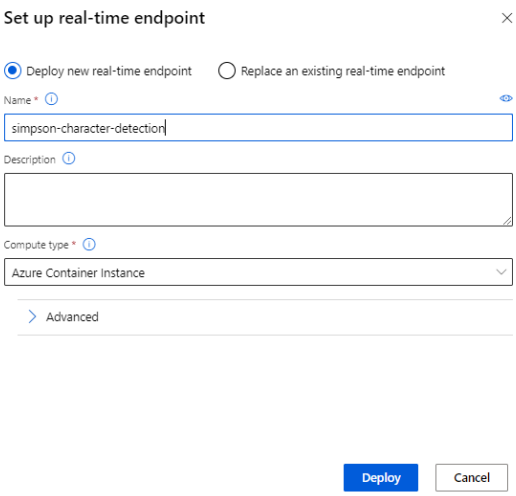
Two new datasets appear under Datasets section. These are effectively modules from training pipeline compiled as a single component.

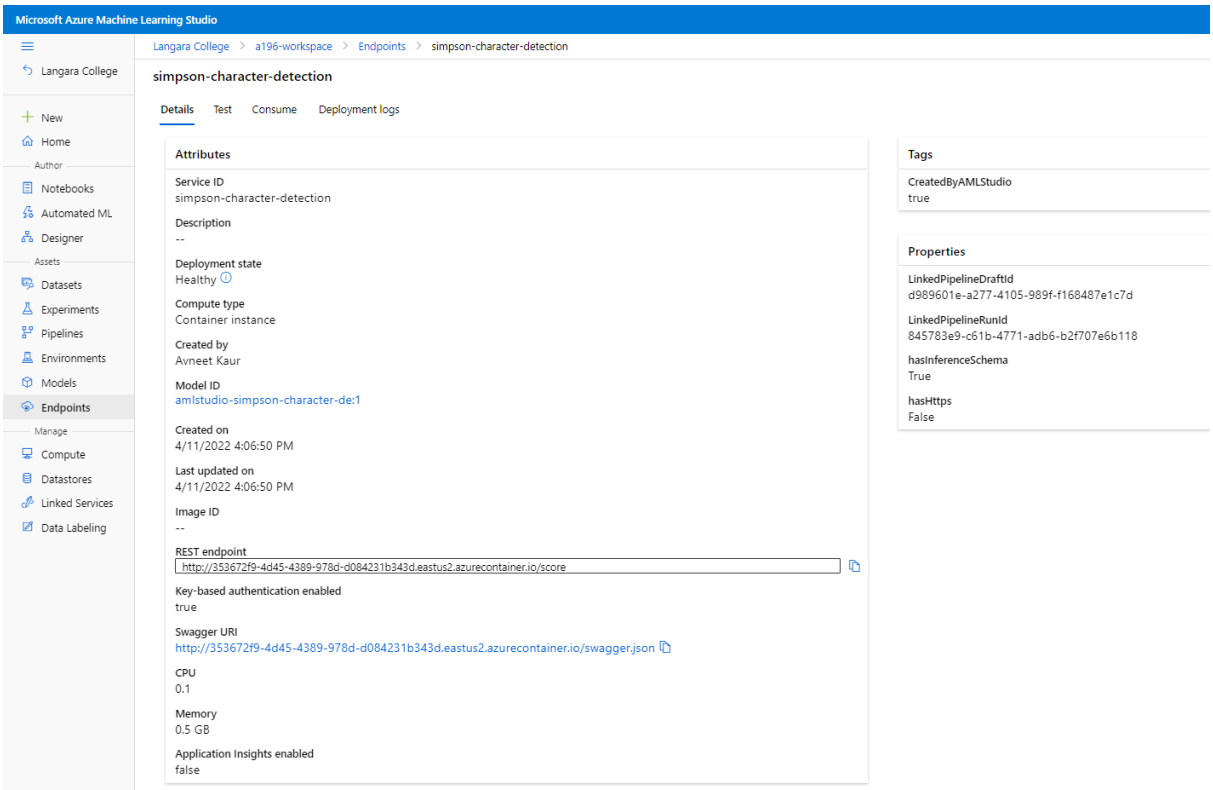


Step 3: Click on Submit to run the inference pipeline. Once completed click on to Deploy to initiate Deployment



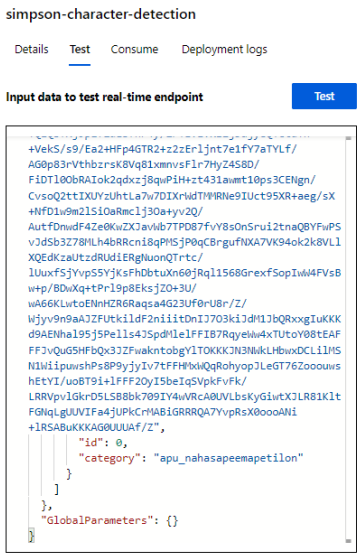
Step 4: On clicking Deploy, a popup to setup a real time endpoint appears. Since we do not have an existing endpoint yet, we create a new one as shown below. Select the compute type on which the model will be deployed as a service. There are two options: Azure Container Instance and Azure Kubernetes Service. We have selected ACI for our model. Click on Deploy once you fill the details. It will take some 15- 20 minutes to complete the deployment.





The deployment state will be in transitioning/unhealthy state until the deployment is completed. Wait for it to become healthy.

Step 5: Test the real time endpoint by navigating to the Test tab under Endpoints section. Click on Test to view the predictions.



Congratulations! You have now successfully deployed an image classification model on Azure Cloud. And it is giving the correct predictions!

Automated ML

Automated ML is an immensely powerful tool of Microsoft Azure that helps everyone to build Machine Learning models without even coding!

It is helpful for both who does not have any coding experience and those who are expert in coding and building machine learning models such as data scientists.

It saves tons of time and energy for the latter group and enables the first group to build accurate and powerful models that they even can store and use them later the new datasets.

Unlike the feature "Notebook" which people with a background in coding can use, it has some limitations in terms of applications. In the table below, you can find the comparison between the Python SDK and Automated ML:

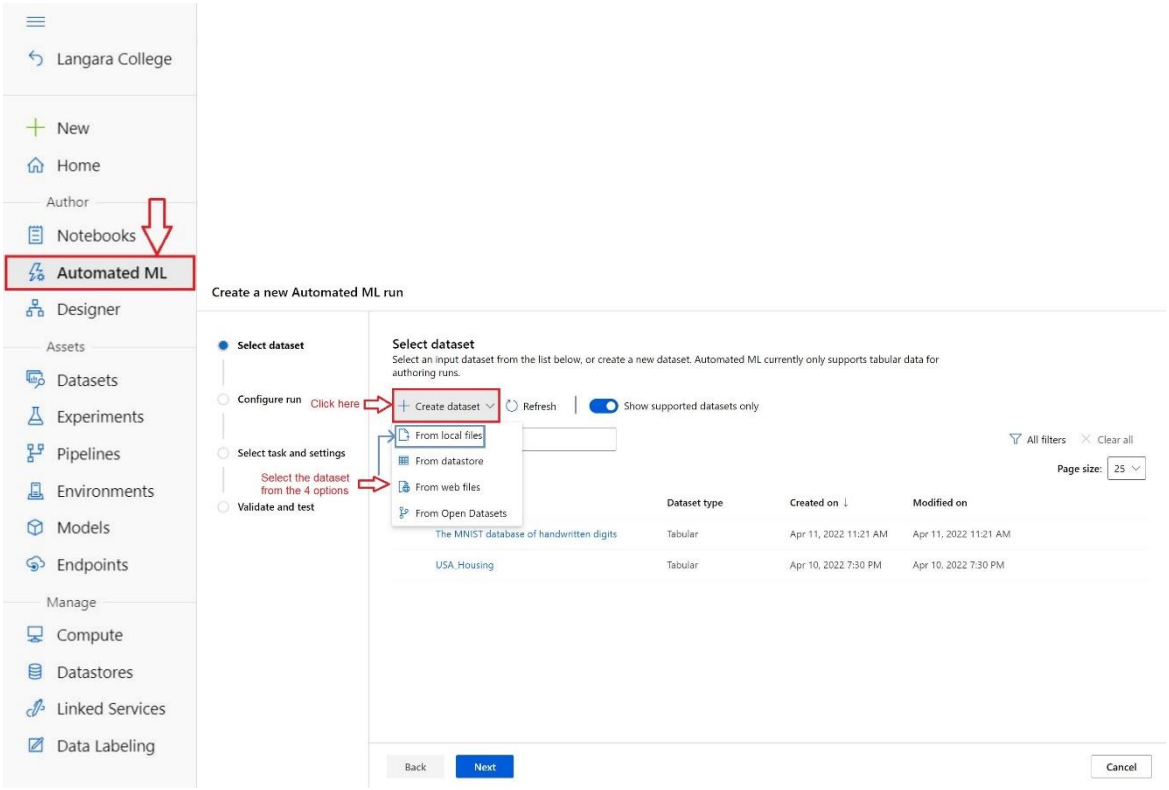
	The Python SDK	The studio web experience
Split data into train/validation sets	✓	✓
Supports ML tasks: classification, regression, & forecasting	✓	✓
Supports computer vision tasks (preview): image classification, object detection & instance segmentation	✓	
Optimizes based on primary metric	✓	✓
Supports Azure ML compute as compute target	✓	✓
Configure forecast horizon, target lags & rolling window	✓	✓
Set exit criteria	✓	✓
Set concurrent iterations	✓	✓
Drop columns	✓	✓
Block algorithms	✓	✓
Cross validation	✓	✓
Supports training on Azure Databricks clusters	✓	
View engineered feature names	✓	
Featurization summary	✓	
Featurization for holidays	✓	
Log file verbosity levels	✓	

In this project, we built 2 Machine Learning models using 2 different datasets. The first one is a regression model to predict housing price in USA and the second one is a classification model using the popular MNIST dataset to recognize handwrites.

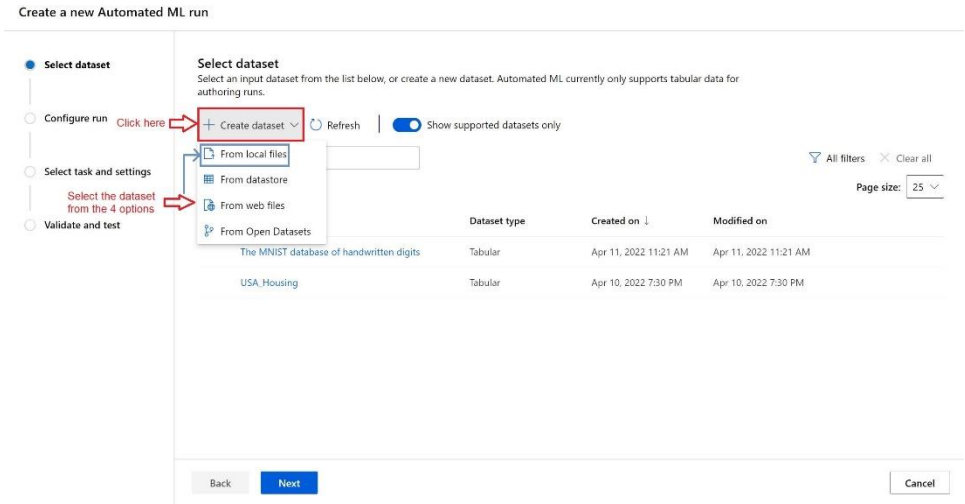
In the images below, you can follow the steps to create the models from scratch:

Regression Model for USA-Housing dataset

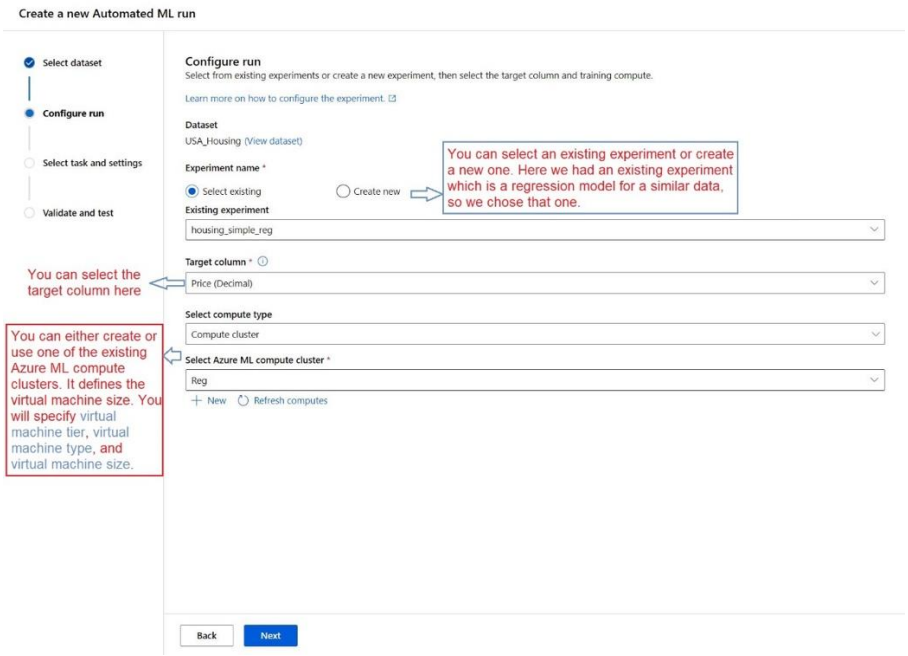
The very first step in to select Automated ML from the left pane then select “Create dataset”:



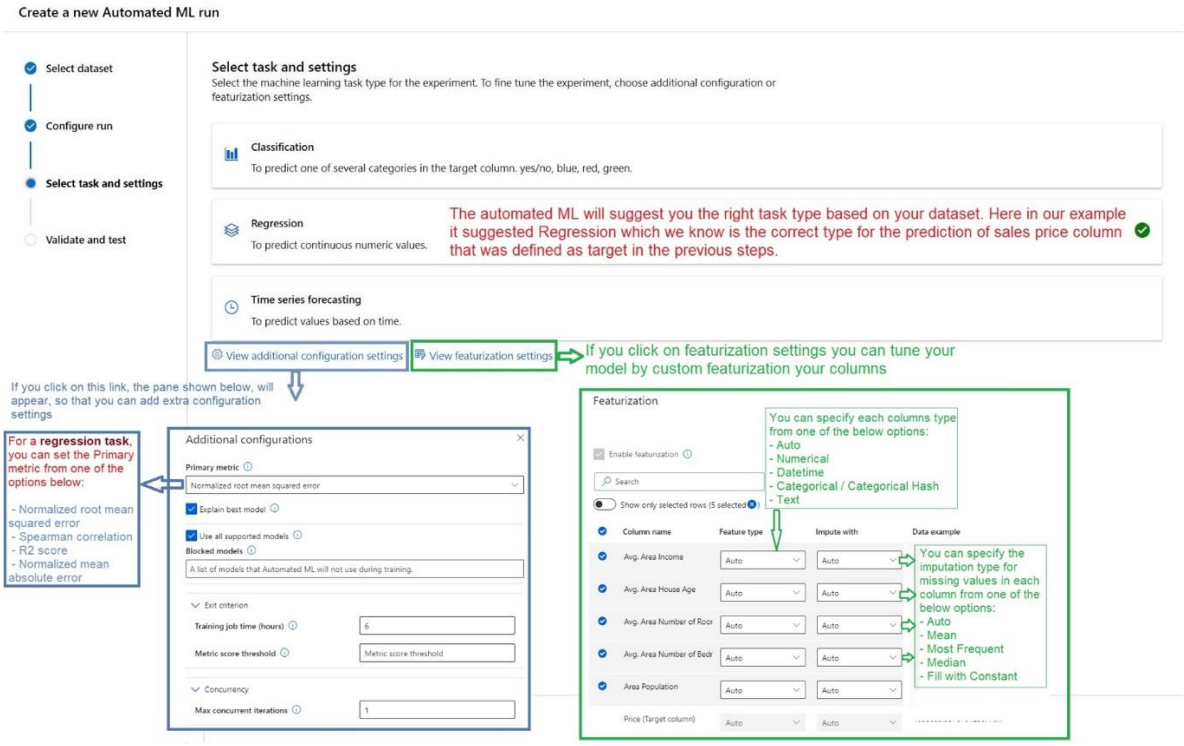
After clicking on Create dataset, then you can choose your dataset from one of the 4 options. We upload the USA_Housing dataset from our local computer:



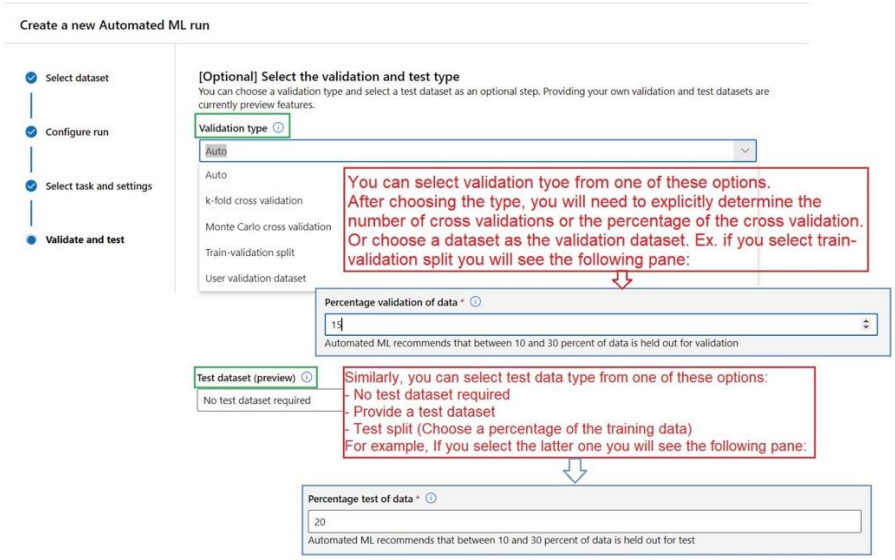
After you select the dataset, you need to specify configure run. The image below shows the steps need to be done in this step:



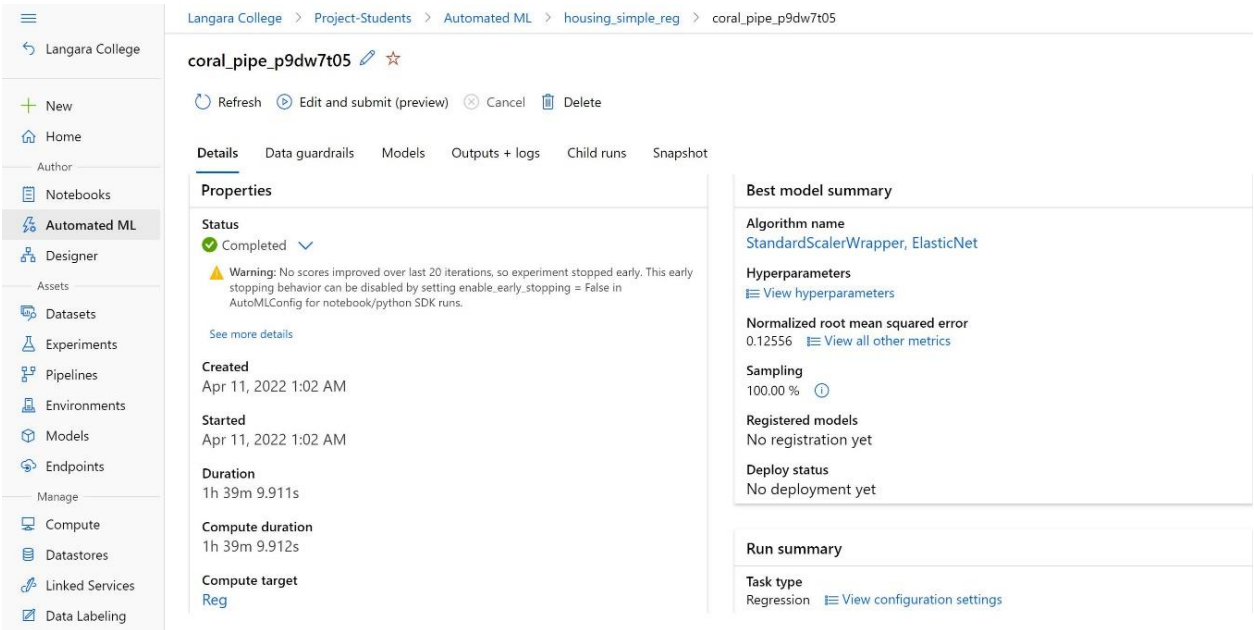
The next step shows an interestingly intelligence of this tool. After selecting the dataset, it automatically suggests the right task for the dataset and even recognise the general configurations. But you can still make improvements or any changes you desire:

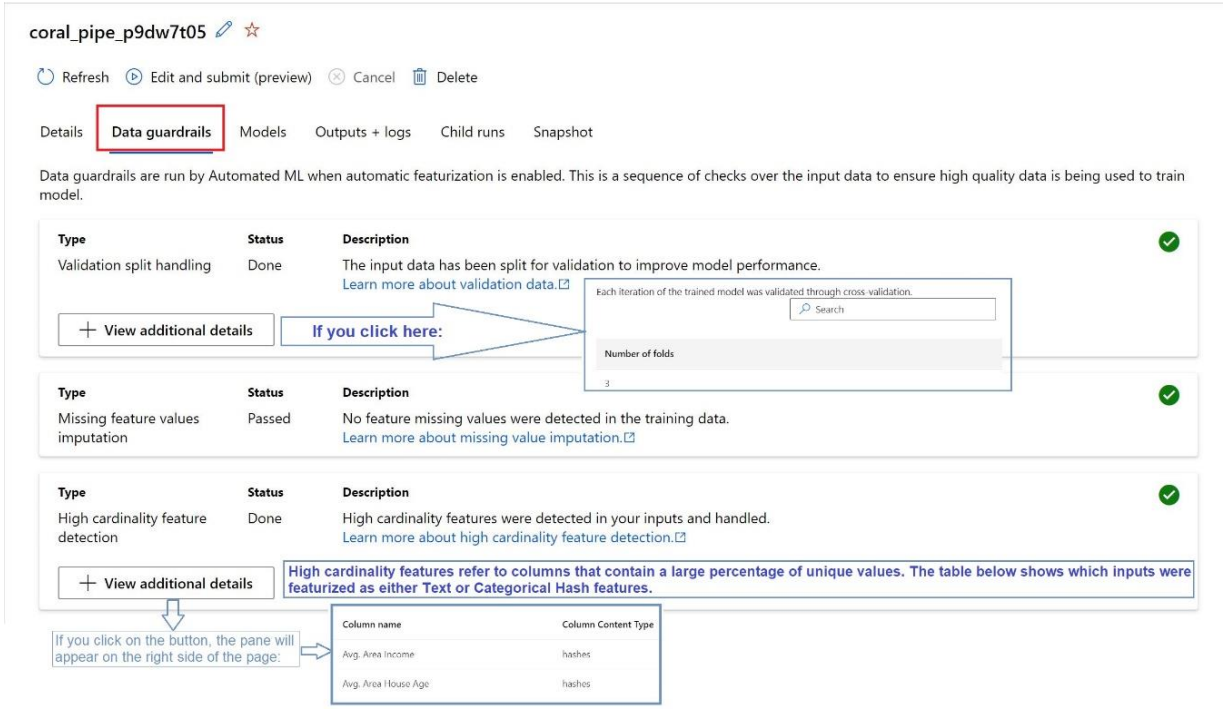


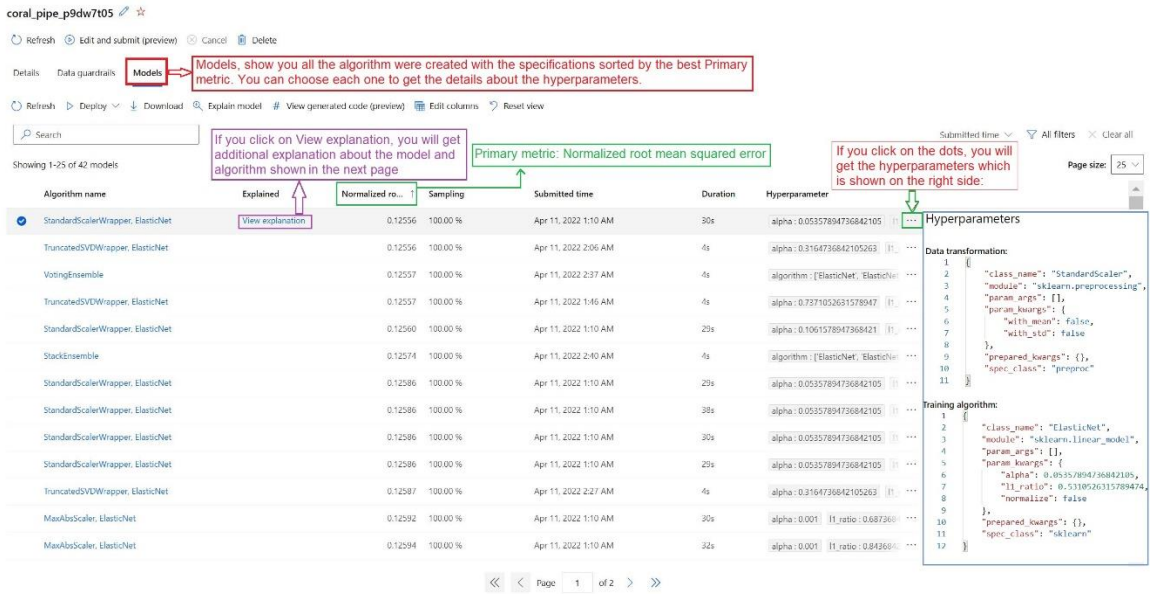
The next step is an optional step for tuning test and validation sets. You can either skip it or customize it to get the ideal result. If you don't want to make changes in this step it will create validation set automatically:



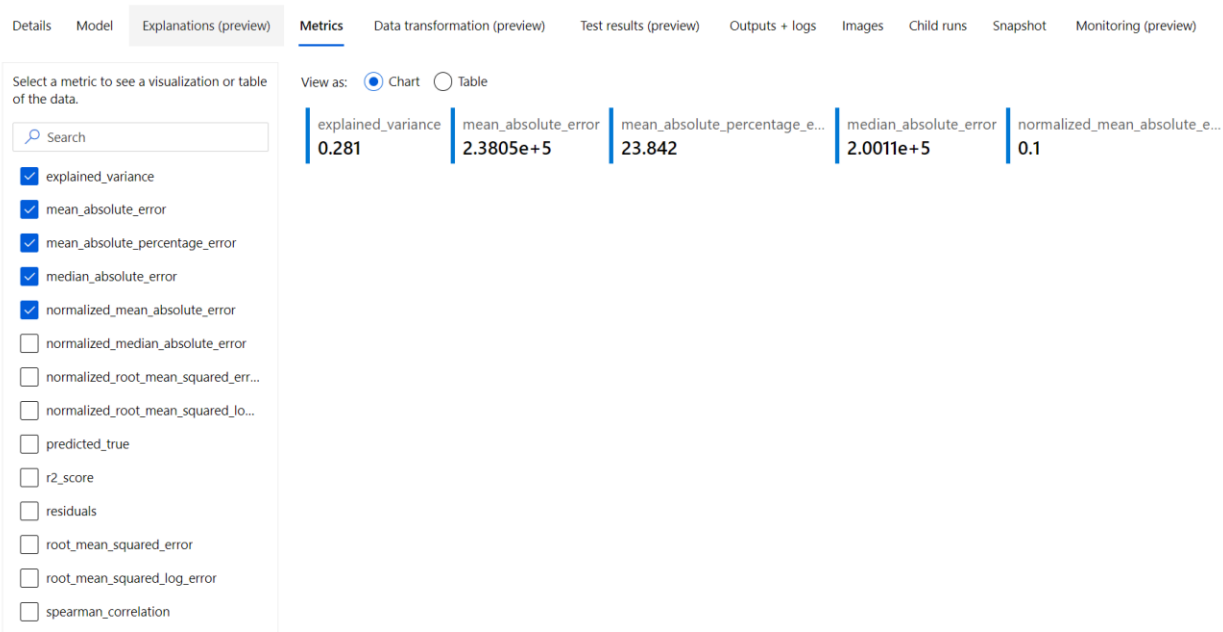
After the last step, then the model will begin creating. It will take a long for the model to be created. When the status is "Completed" then our model Machine Learning algorithm is ready. You can explore it by going through the tabs: 1- Details. 2- Data guardrails. 3- Models. 4- Outputs + logs. 5- Child runs. 6- Snapshot. We will review the first 3 tabs. But there are many details that you can go deeper to investigate. This is an interactive model reporting tool so that you can select any feature you would like at a time.



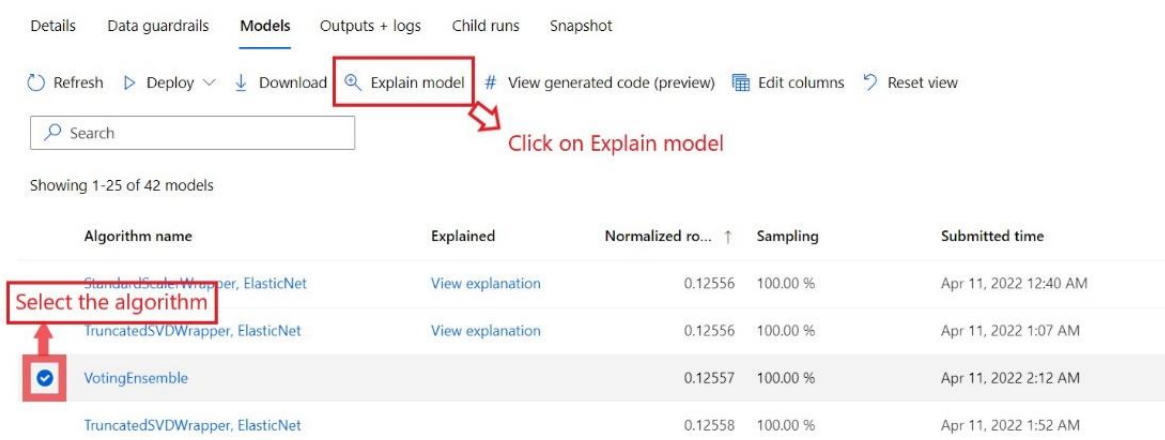




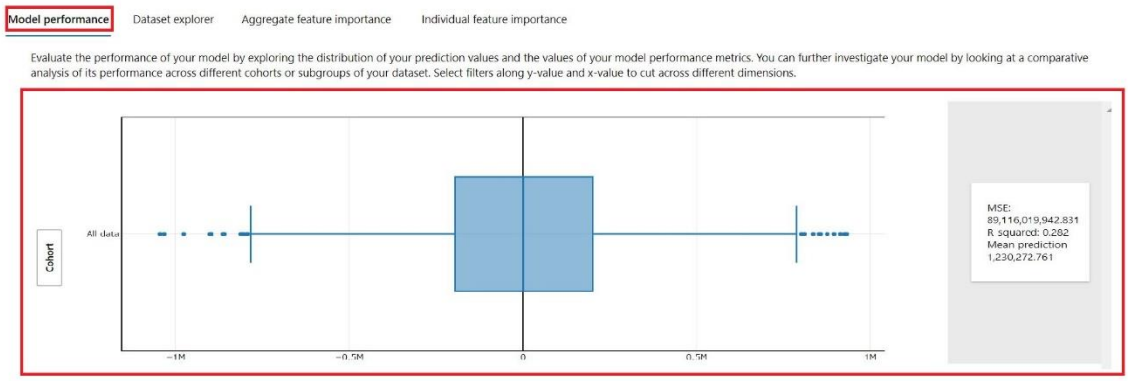
If you select one of the algorithms, you can go deeper into that algorithm to explore it better. For example, you can check all other metrics for that specific algorithm. In the picture below we selected the best algorithm to check the metrics. You can also select which metrics you want to examine.

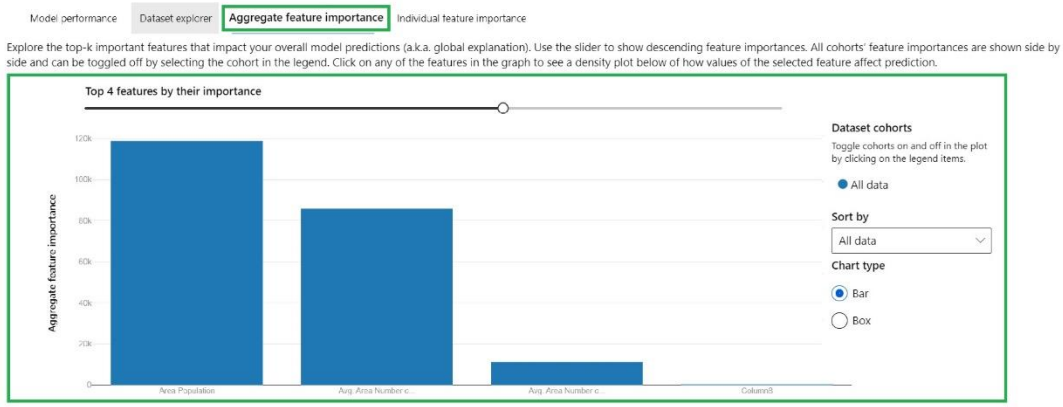


You can check the “View explanation” for all the models. To do that, you will only need to select the model and click on the explain model on the toolbar which is above the list of the models.

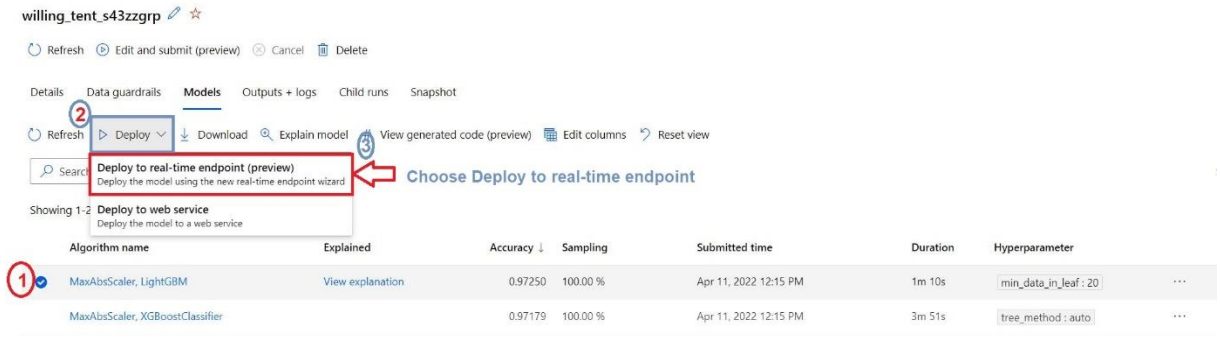


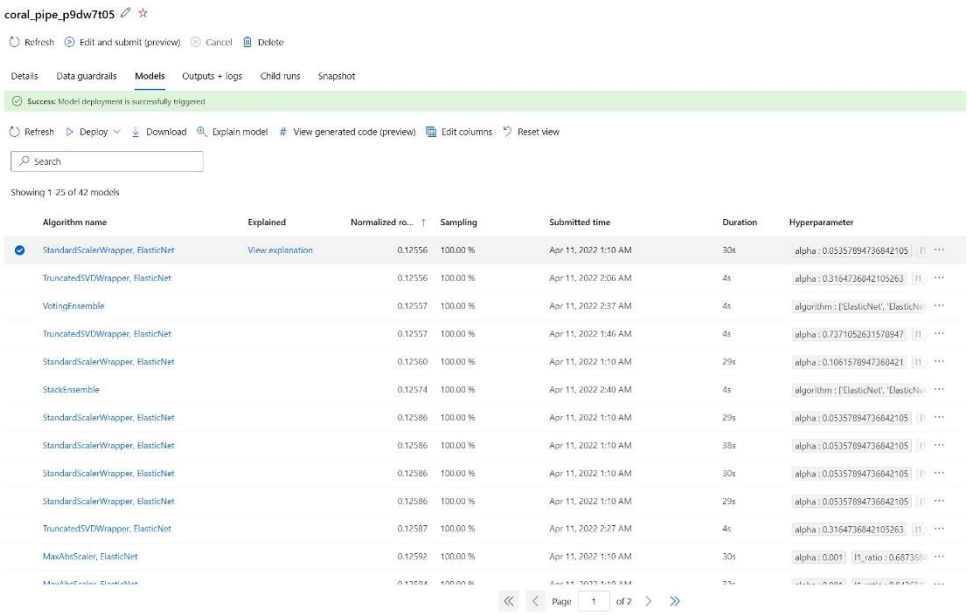
When you click on “View explanation,” then you will see the overall performance of that algorithm based on the distinctive features of data. For example, you can find out the most key features of the selected algorithm with their weight of importance. This is an interactive tool, so you can specify the number of prominent features by dragging the slider on top of the bar plot.





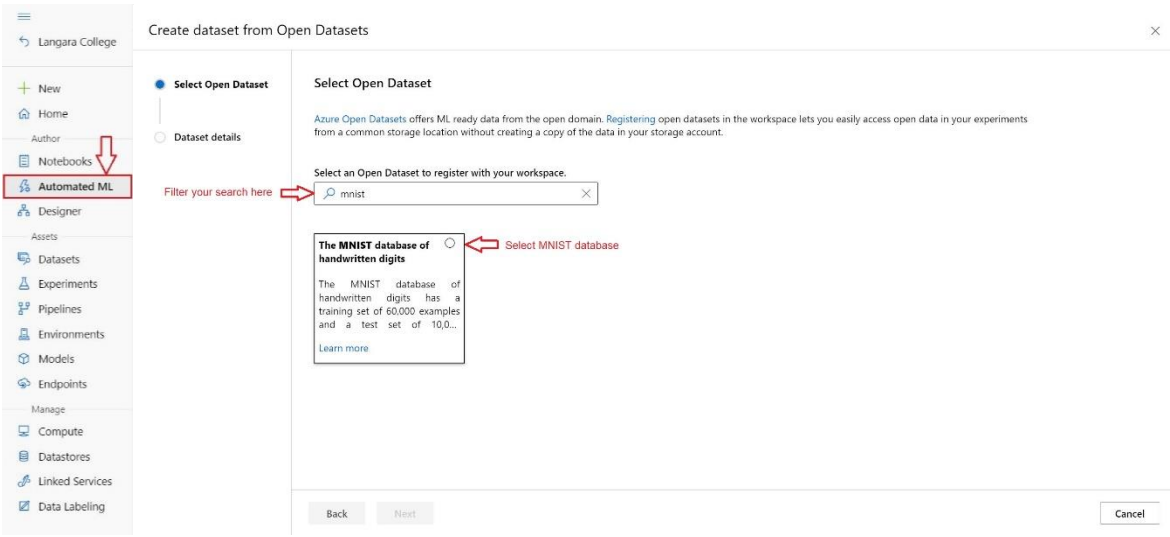
The next step is model deployment. The images below show the process for that step by step:



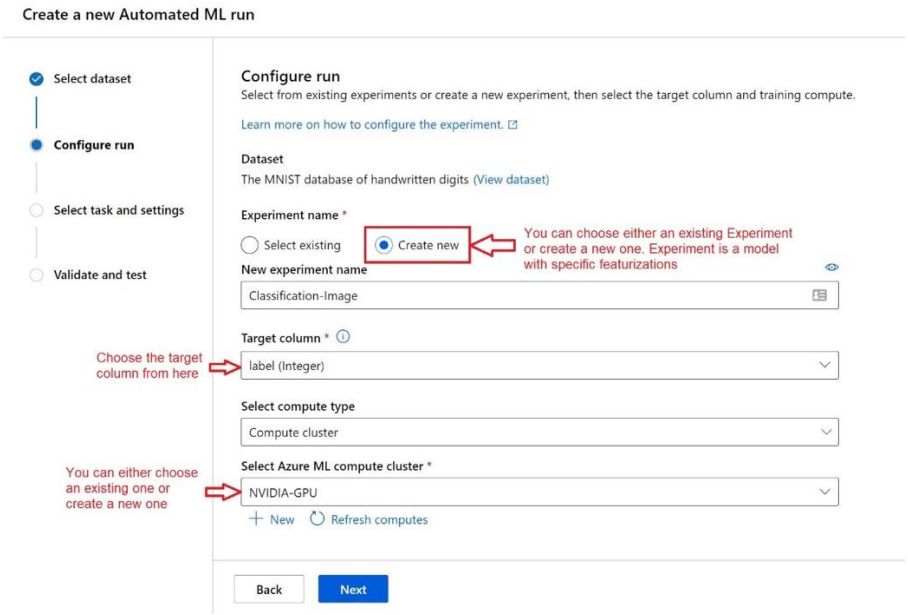


Classification Algorithm Based on MNIST Dataset

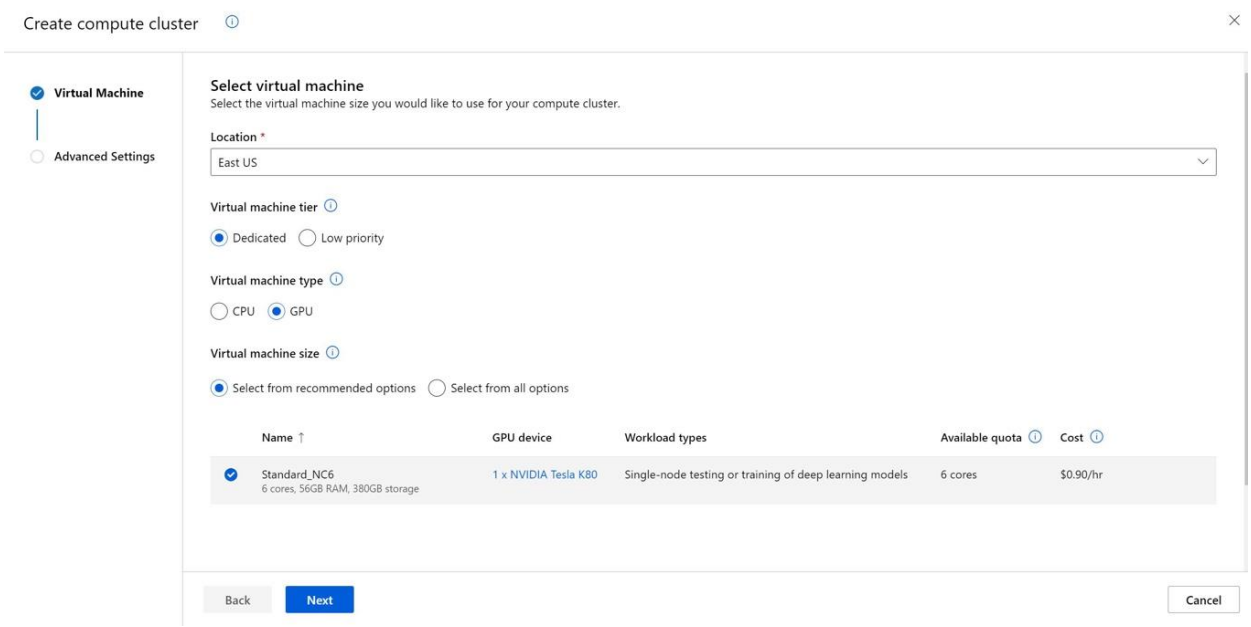
Similarly, we run the same steps for the other dataset to create a classification algorithm to recognize numbers from handwritten images. The difference between MNIST dataset and USA-Housing data is that MNIST is an open dataset that you can directly import from Microsoft Azure without needing to download or upload it. The images below, show the process of classification model creation:

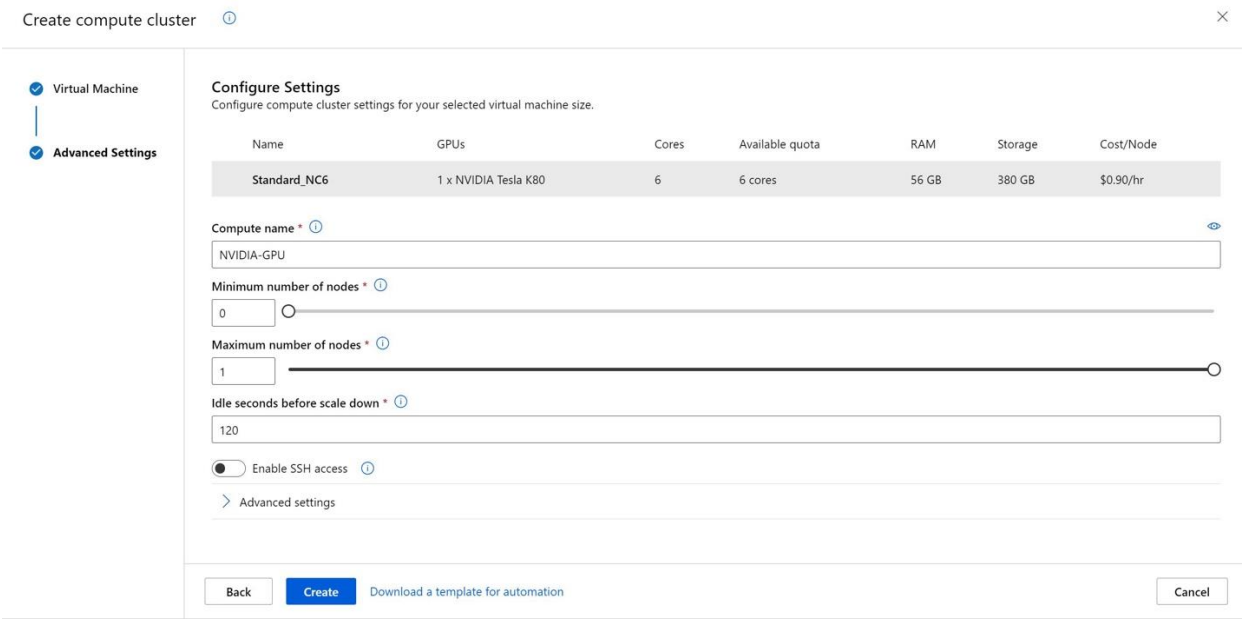


We go through the same steps as before, to create the Automated ML for a classification algorithm:

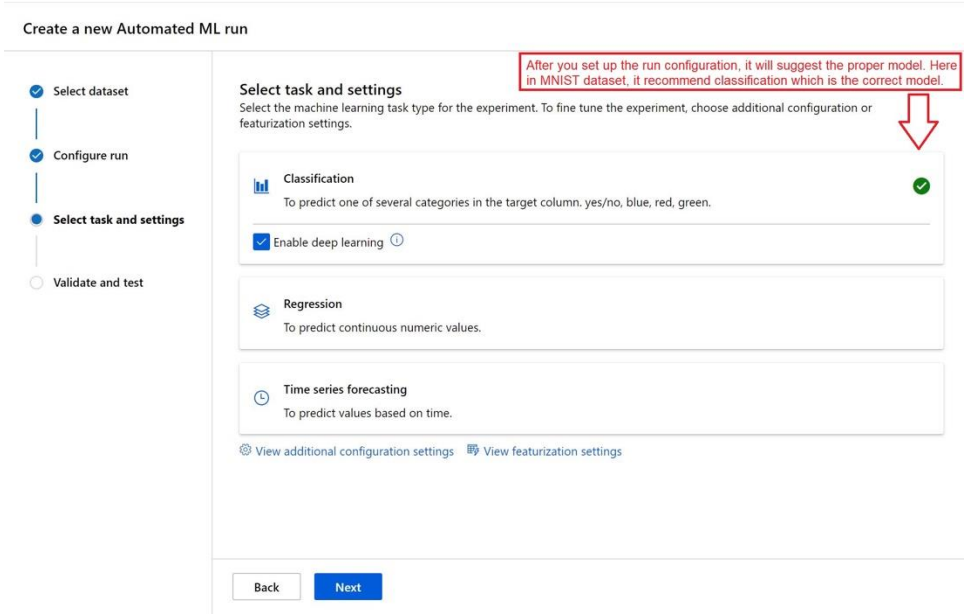


Because the image classification problem needs more advanced algorithms, to accelerate the process of model creation, we chose GPU for the compute cluster:

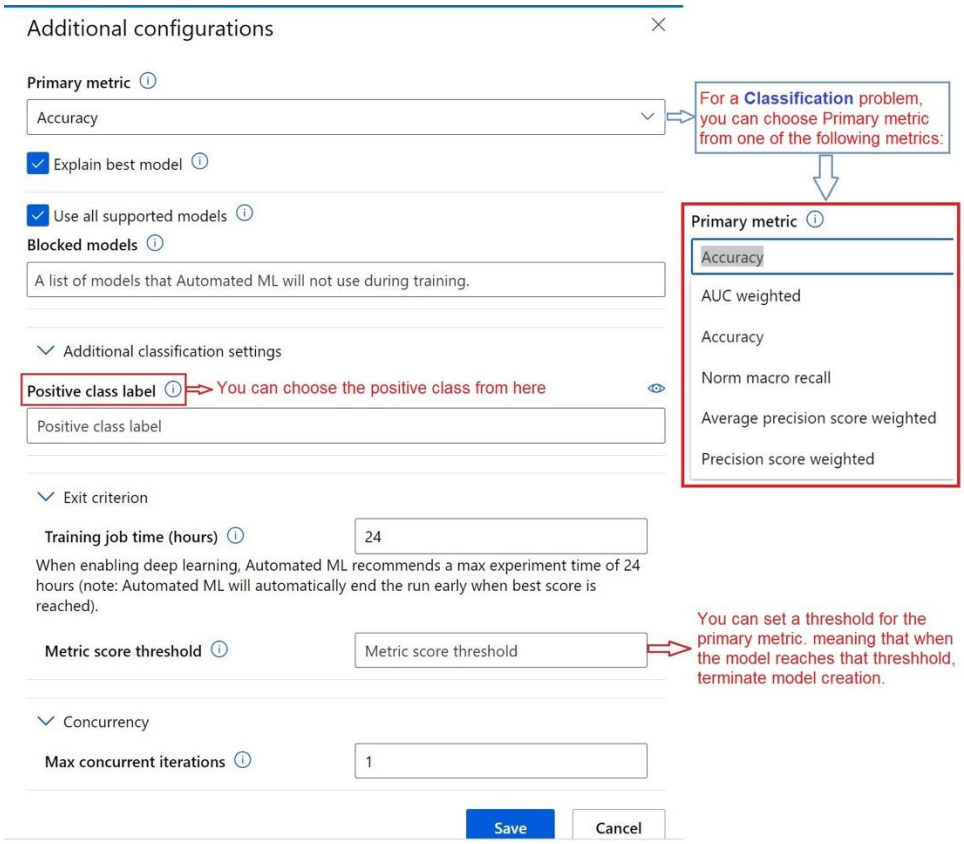


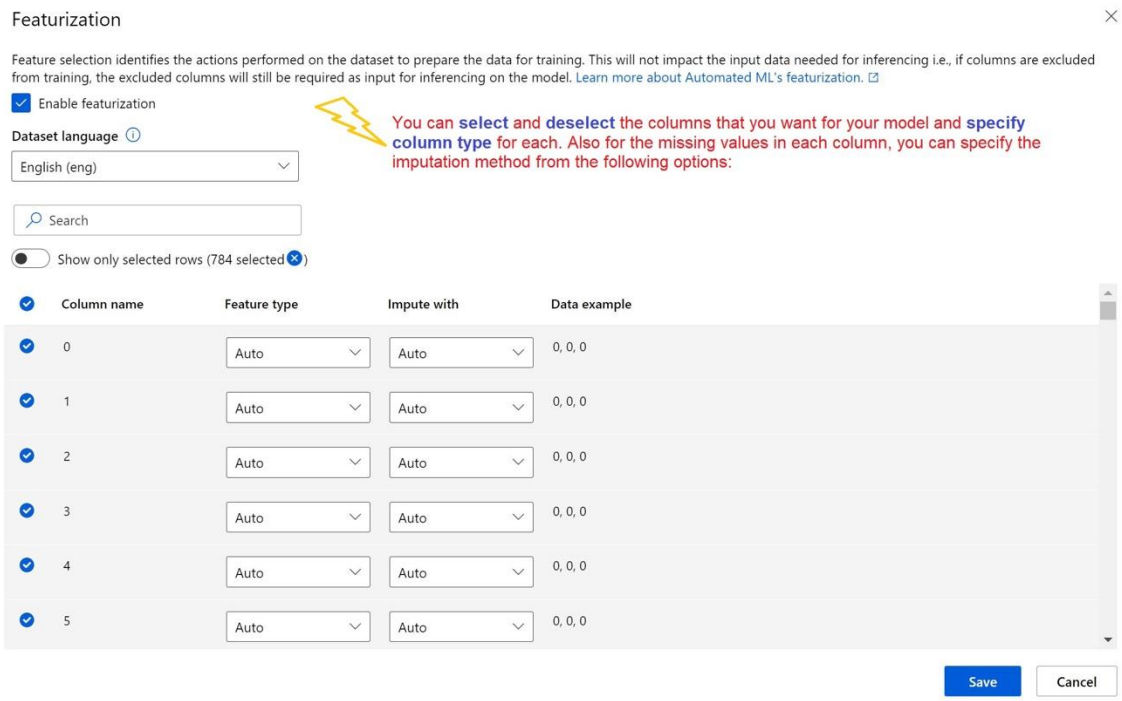


After creating the compute, it will show you the appropriate machine learning task based on your data. Of course, you can change it to the type you would like but mostly the recommendation is the correct one. Similarly, to the previous training model you can change the configuration and featurization settings to tune your model:



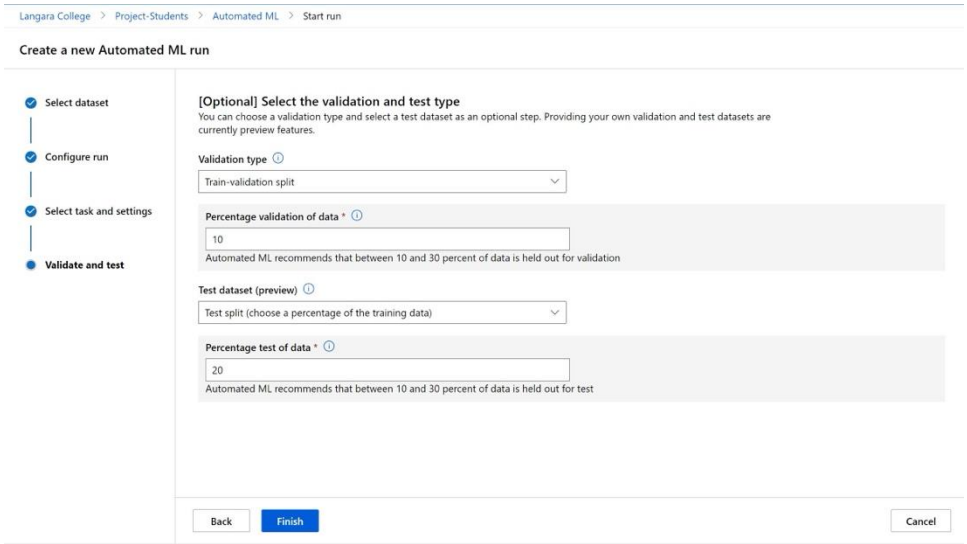
For example, if you click on “View additional configuration settings” or “View featurization settings” you can tune your model shown in the images below respectively:



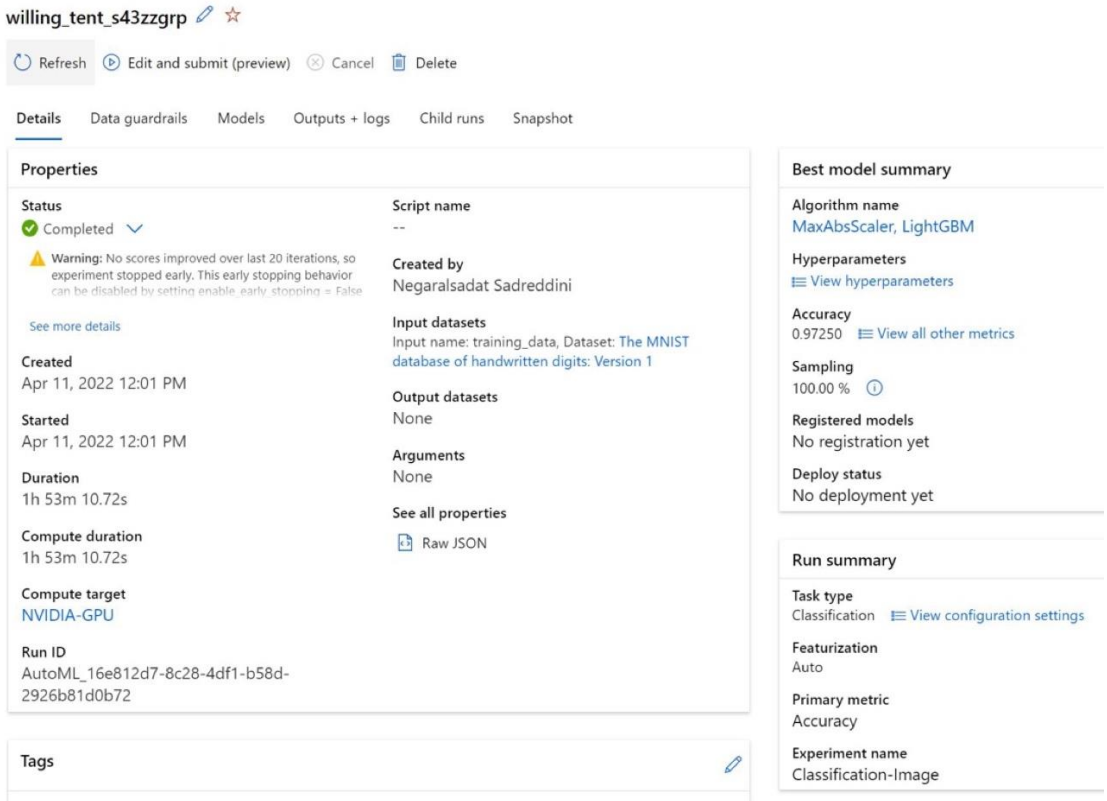


And the last step, which is an optional one, is to tune the validation and test sets. You can also choose another dataset as your validation or test tests. But if you want to split your own data

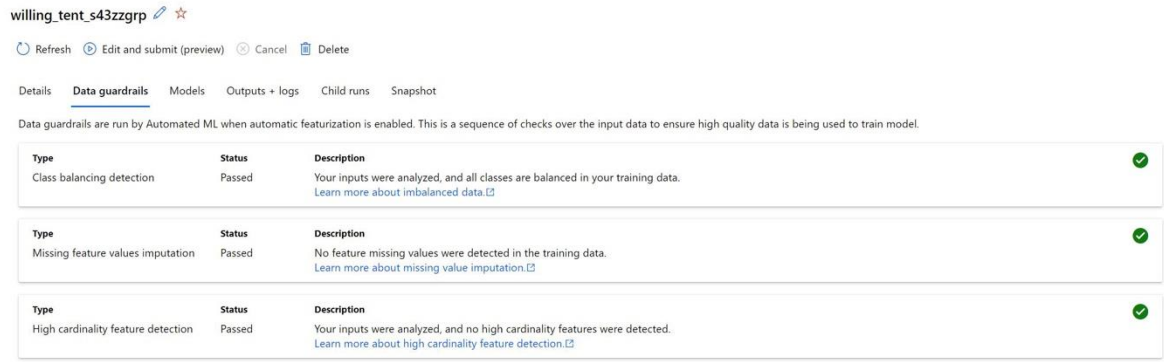
into training, validation, and test you need to specify the percentages. Here we chose 10% of the training set as a validation set and 20% of the whole dataset as test dataset:



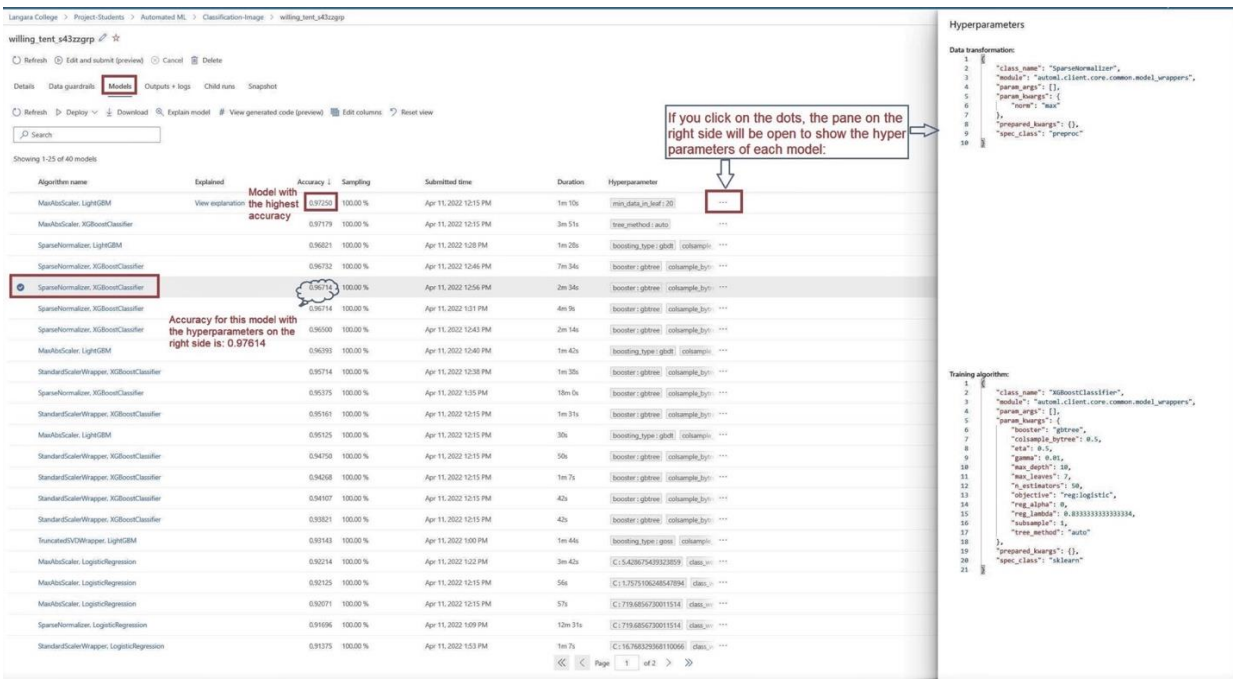
After that, you click on the Finish button and your model training will be start. After the training is finished, you will have access to all the trained algorithms. The first thing that you will see is the model details that shows you general information about the model such as creation date, compute duration, best model summary and so on.



You can ensure the data quality by choose data guardrails tab and detect all the potential problems and the solutions the automated ML used to address them. For example, here there was no issue with the data:

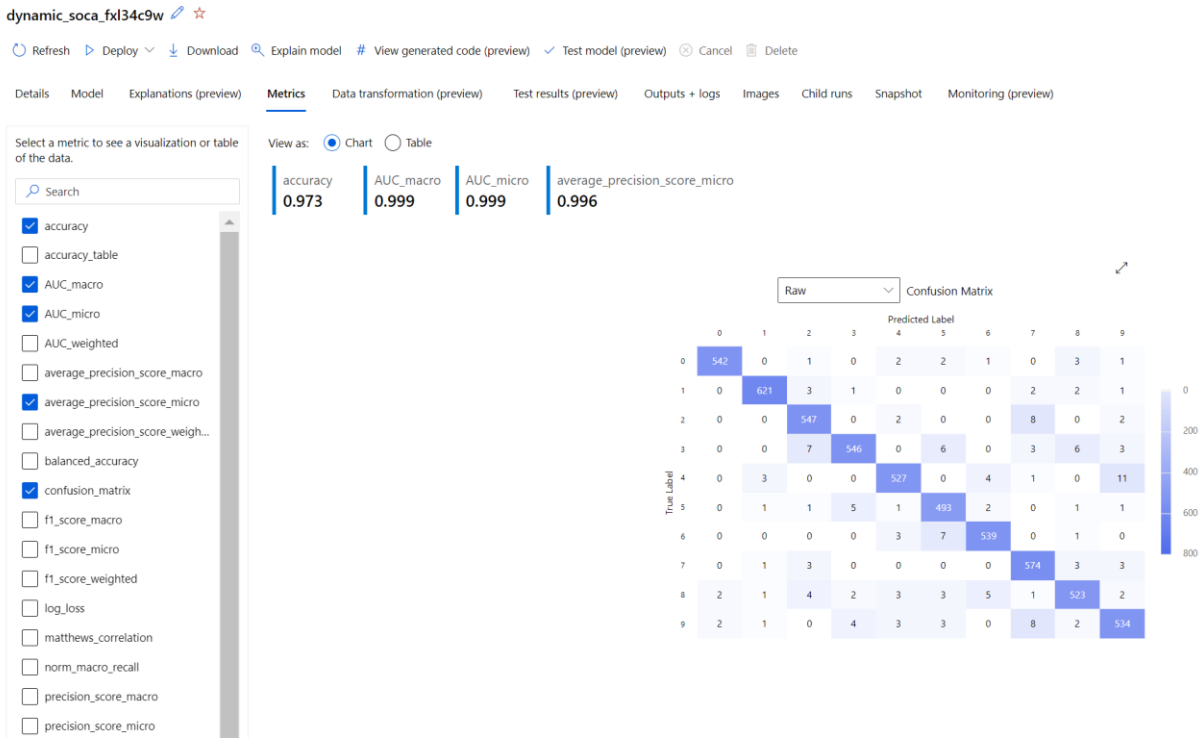


And if you choose Models, you will get all the details for each trained algorithm:

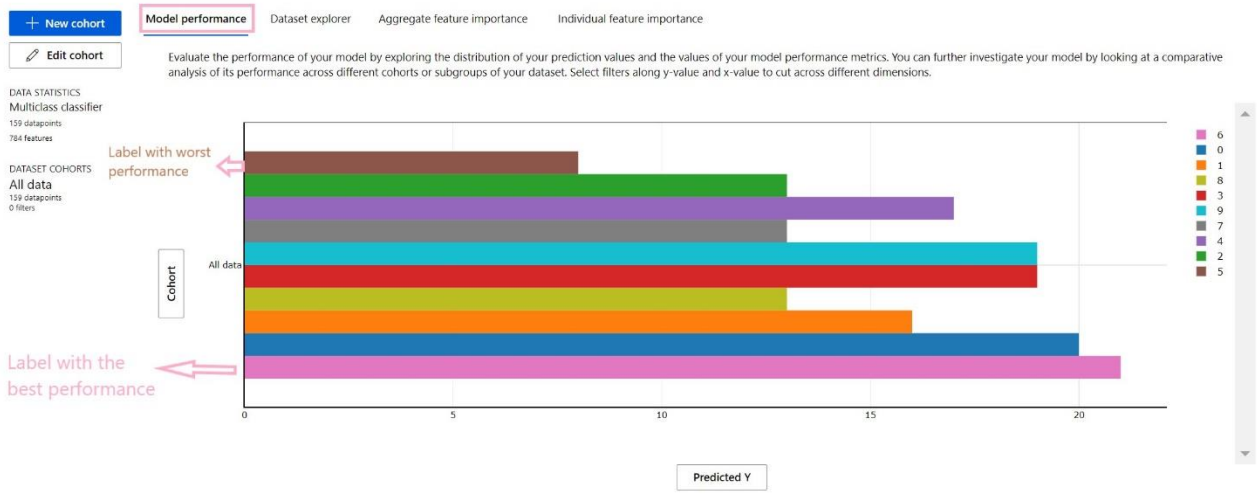


Because this is a classification problem, the model explanation is more attractive than a regression model. So, you can work on the labels and intuitively check the importance of each label in the trained model and algorithm. After clicking on the View explanation, you can check all the metrics that you desire. Here confusion matrix shows an interesting fact. For example, the labels 7 and 2 has high similarity with each other with 8 number of misclassifications. We can observe that these two numbers are also more like each other in handwritings. Or the

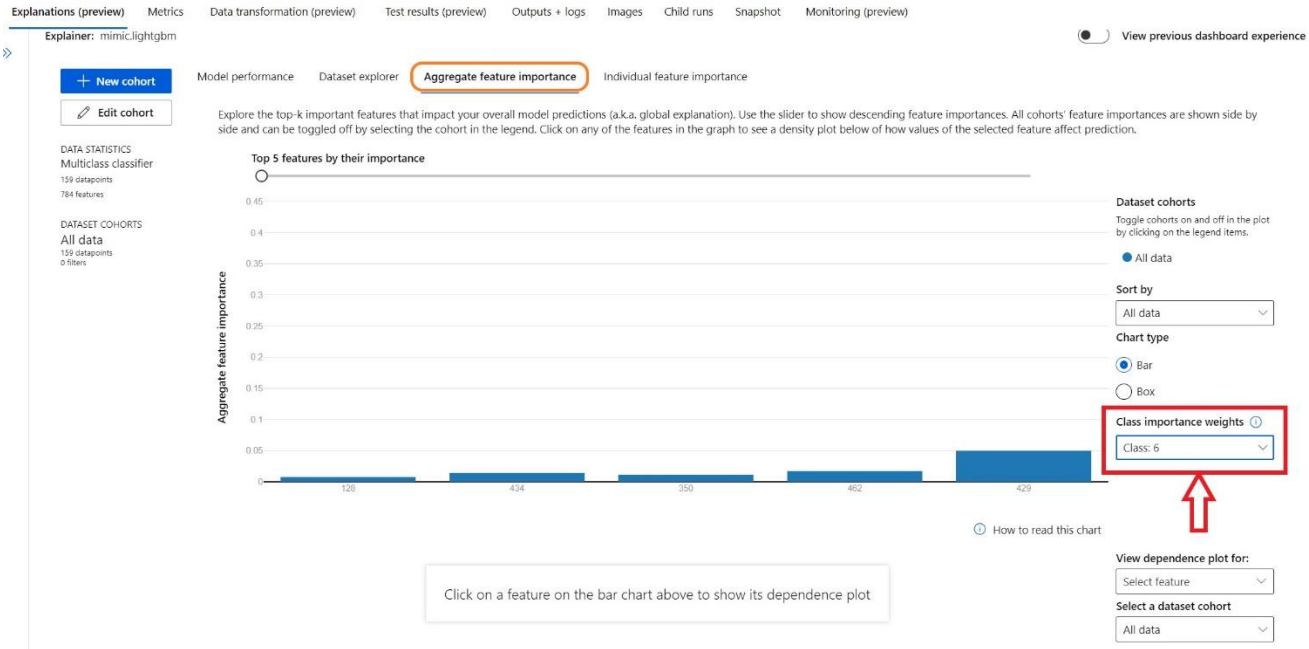
number 9 has been predicted as number four, 11 times and we know that these two numbers are also remarkably similar to each other.



Then, if we click on explanations, we can check the model performance for each label. Here we can see that the label 6 has the best performance in terms of prediction. If we go back to the confusion matrix in the image above, we can justify this model performance.



We can also check the top features in the trained model and interactively choose the number of key features by dragging the slider. Also, we can check the importance of those features for each label in the dataset. The first image below shows the top 5 features by their average importance weigh and the next image shows the importance of the same features in terms of prediction of the label 6:



Comparing the two images above, you can see that the importance weight of the most prominent features varies label by label and the first image shows the mean aggregated feature importance.

Model deployment is the same as the one explained in Designer parts. To avoid the repetition, we did not include the steps here, but you can look at model deployment in Designer section to see how you can test your model with new data after deployment of the model.

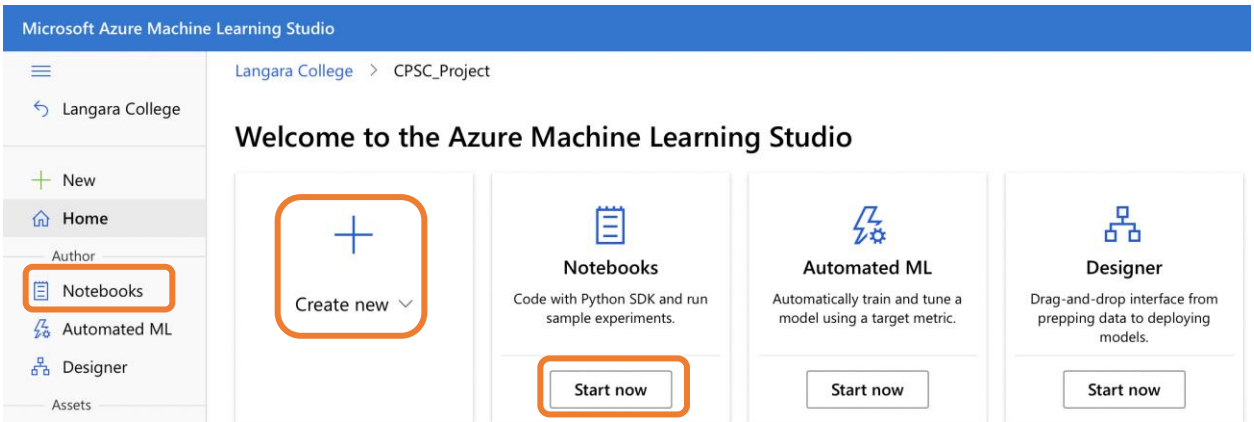
In general, it is a very wide area that not all the tabs and features can be shown in this document. But it is very user friendly that everyone can explore it easily by just going deeper into the details. It is designed in a way that everyone can understand and extract the information they need. So, you can have your own interpretation and a set of metrics and features that you want to include in each of your projects or works.

Notebooks

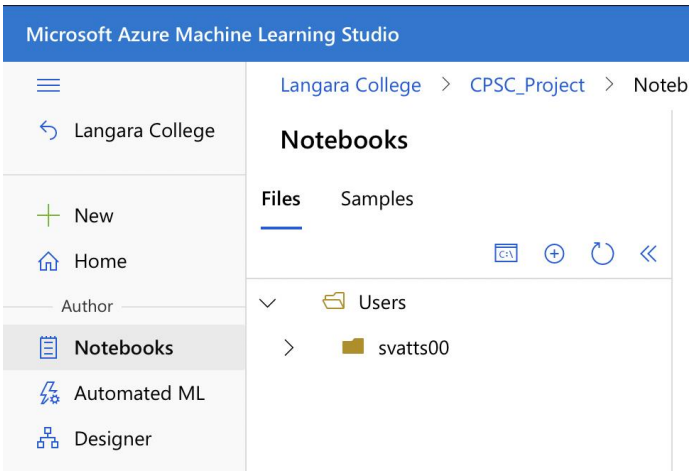
Notebooks provide a developer-friendly interface for developing Machine Learning Models on Azure. In this tutorial, we are focussing on creating a Language Detection system. This is a classification problem where the input is a text in any language and the model predicts the Language of the text.

Development Phase

Step 1: In your Machine Learning Studio, click on Notebooks under Author from side menu. You can also click on Create New drop down or 'Start Now' under 'Notebooks'

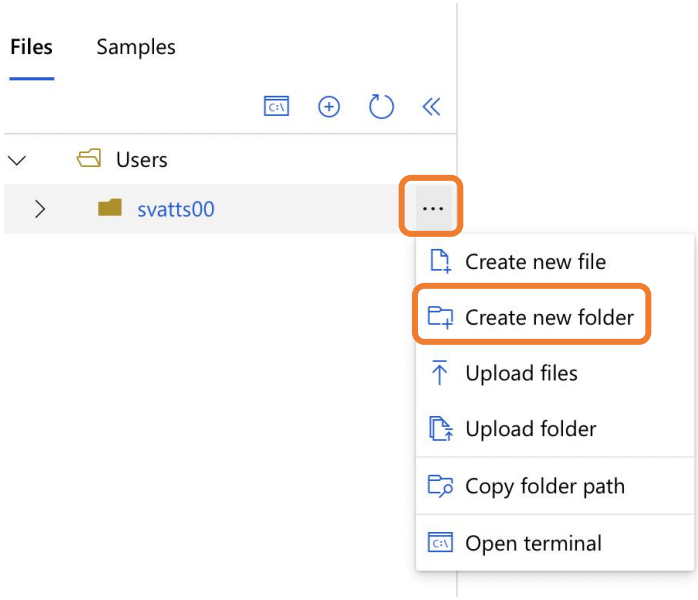


You will get a folder structure like a working environment you have on your local systems.



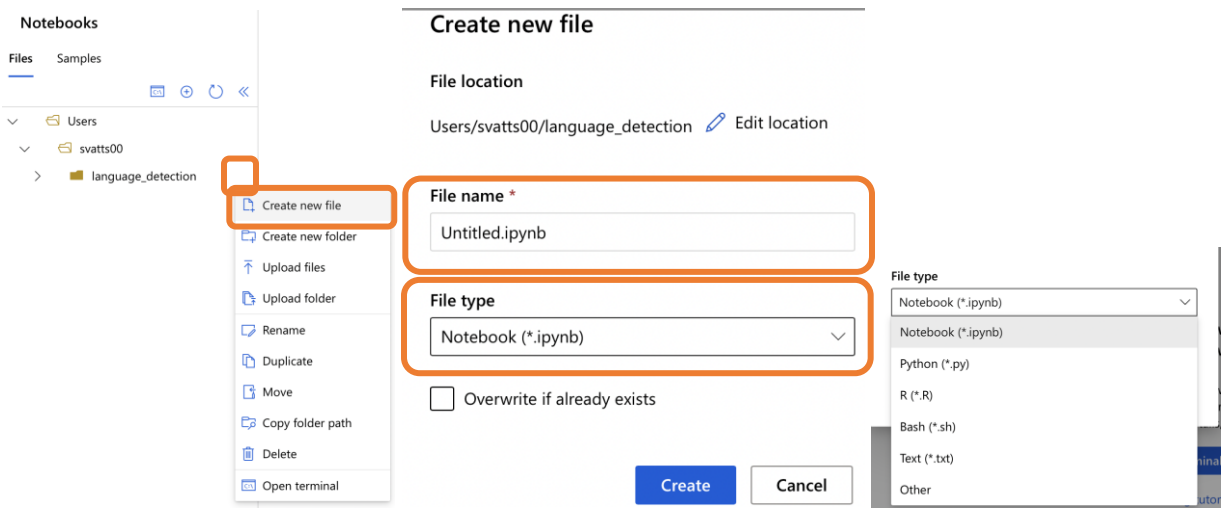
Here, you can create a project structure for your ML tasks or upload existing folders/files from your local system. We will create everything on the cloud.

Step 2: Create a directory for Language Detection project by clicking on "…" of your username folder, and then on create new folder.



In a similar way, you can create an entire directory structure if needed.

Step 3: Create a new file for writing your machine learning code. Click on "…" on your newly created folder and then on Create new File.



Step 4: Enter the Name of the file and select the file type. By default, Jupyter Notebook files are selected. You can select a different file type to create R, Python, Shell scripts etc.

Step 5: Click on Create. Your notebook is ready to start coding.

As known, Jupyter notebooks (or any other files) use a Ipython Kernel to run the code. This kernel resides in a compute target. If you have already created a compute instance, it gets automatically selected. Make sure the instance is in running state before running any cell of the notebook.
If you do not have a compute target yet, refer to this section to create one.

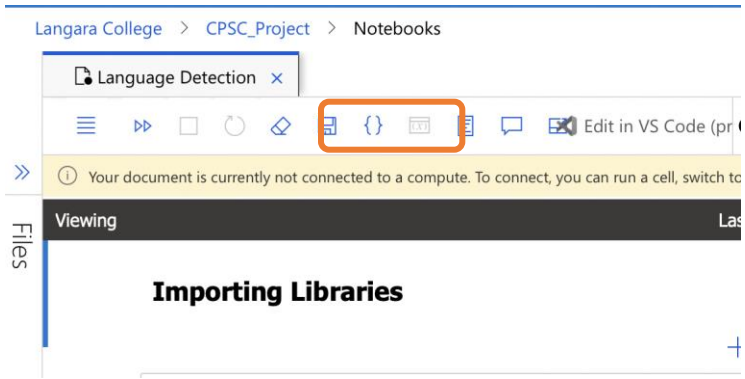
Step 6: Write your code in the notebook. You can download the Language Detection notebook from here

Step 7: Upload the Dataset in the same directory as that of notebook. To upload a dataset, select "…" on the project folder and click on upload files. Now choose the file from your local system and upload it to Azure. You can find the formatted dataset here.

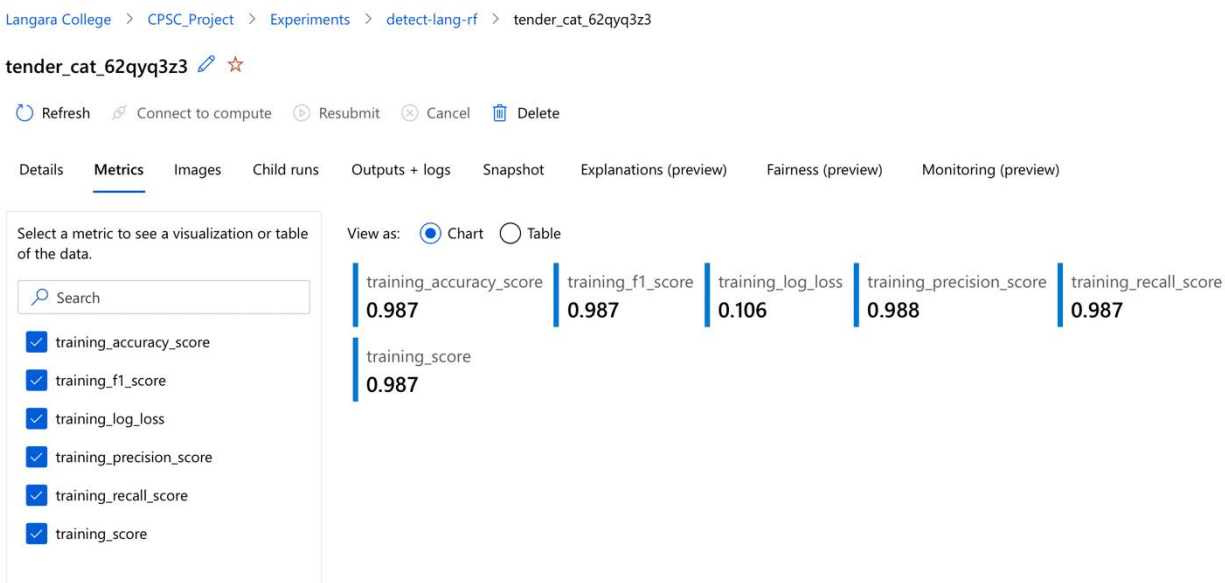
Step 8: Run the Jupyter notebook. This notebook contains 4 different models which are being analysed for best accuracy.

Once the development is completed and you have selected a final model for your classification problem (here language detection) along with the optimal hyperparameters, you can proceed to model deployment.

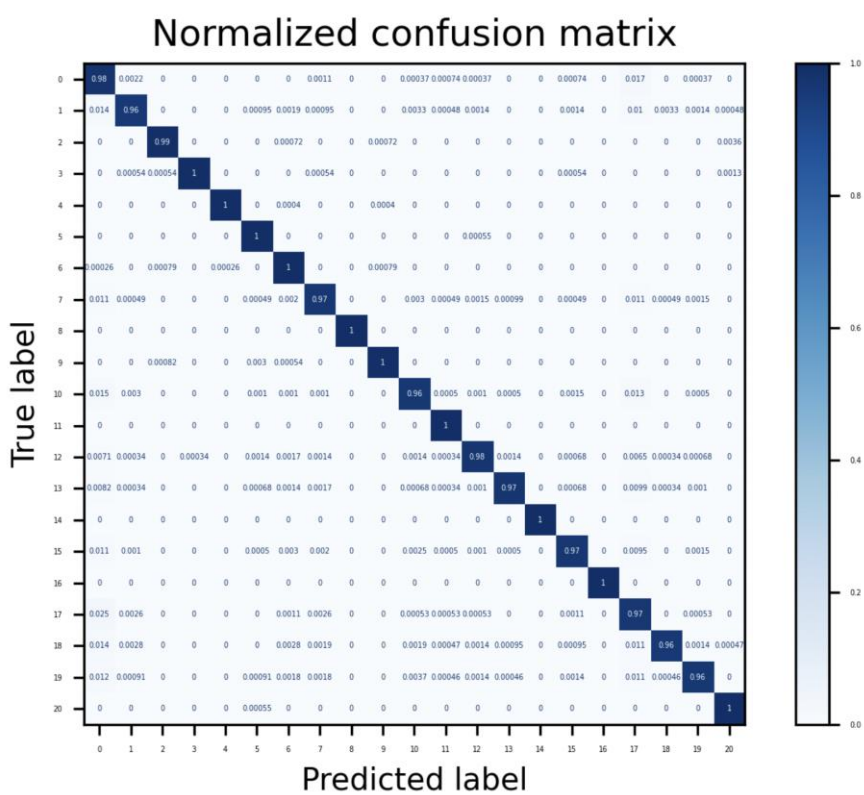
Note: The Azure provides integration with VS Code. You can click on "Edit in VS Code" to setup the environment on your local system to edit notebooks, manage your azure resources and run everything on your Azure cloud from local system.



You can view the metrics of the model by going into *Experiments -> Selecting our Experiment -> Selecting a pipeline -> Clicking on Metric*



You can also check out the confusion matrix by going into Images.

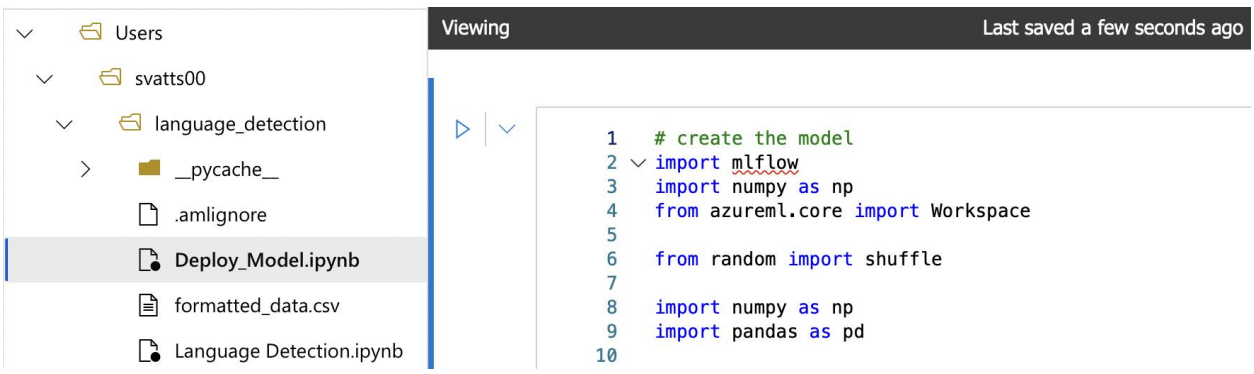


Deployment Phase

Deploying your machine learning models using Azure Notebooks is slightly complicated as compared to Automated ML and Designer. It requires you to configure all the components such as Workspace, Experiments, Model Registry, Endpoints etc.

Don't worry. Azure provides Jupyter notebooks with complete code for configuring your deployment. In this tutorial, we are going to explain the different configurations required to deploy our Language Detection Model. Refer to this guide alongside for better reference.

Step 1: Create a new notebook for deployment configuration and settings.



Step 2: Add the Model Development Code for the Optimal model selected along with required Libraries. Also, add the code required for deployment from the above link. You can download the compiled notebook from here.

Let's understand the deployment code:

- The deployment requires the following libraries for connecting to Azure Services:

```
# create environment for the deploy
import mlflow
from azureml.core import Workspace
from azureml.core.environment import Environment
from azureml.core.conda_dependencies import CondaDependencies
from azureml.core.webservice import AciWebservice
import uuid
from azureml.core.model import InferenceConfig
from azureml.core.environment import Environment
from azureml.core.model import Model
```

- To connect the python runtime to Azure workspace, we need to setup the workspace configuration. When running the Azure notebook from Azure Cloud (or connected VS Code environment), we do not have to set any Environment variables to define workspace

configuration. If you are performing it locally, you need to define required settings for workspace. Refer to guide link above for more info.

```
# connect to your workspace
ws = Workspace.from_config()

# create experiment and start logging to a new run in the experiment
experiment_name = "detect-lang-rf"

# set up MLflow to track the metrics
mlflow.set_tracking_uri(ws.get_mlflow_tracking_uri())
mlflow.set_experiment(experiment_name)
mlflow.autolog()
```

- Once the workspace is configured, we need to connect to an Experiment. We can define a new experiment or use an existing experiment.
- We can set the autologin feature to enable logging of each step of deployment. It helps in debugging if deployment fails.
- To train the model inside an Experiment pipeline, we need to encapsulate the model Fitting inside MLFlow run. This enables logging of model metrics.

```
# train the model
with mlflow.start_run() as run:
    optimal_modelRF.fit(X_train_features,y_train_features)

```

- Once trained, we need to register the model. Registerig model is similar to saving the model as a pickle file in cloud environment to be used in future. To register the model, we need to define a URI for the model using the given format. Since models are versioned, we need to select the latest version using the run_id and register the model using a model name.

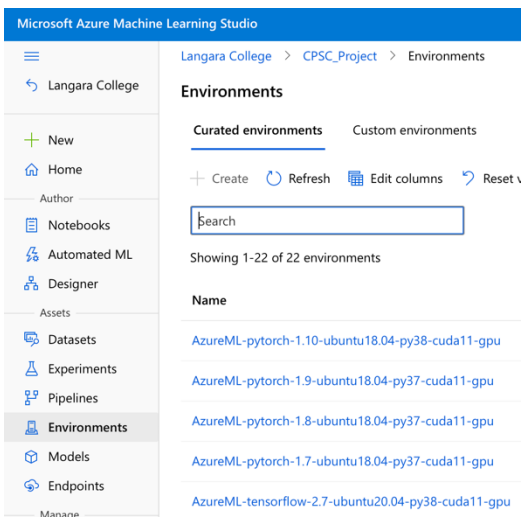
```
# register the model
model_uri = "runs:/{}/model".format(run.info.run_id)
model = mlflow.register_model(model_uri, "lang-det-rf-model")
```

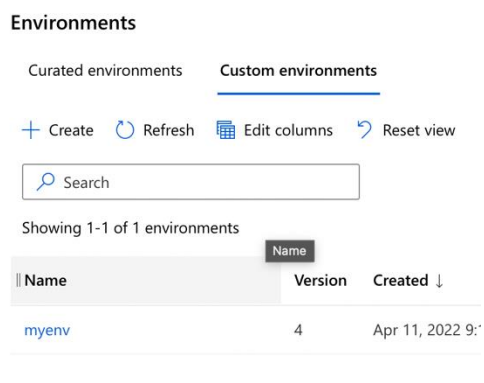
- Once the model is registered, we need to provide an Environment on the compute instance in which the model will run. Model runs in a container on the compute service so that it runs with pre-configured environment with defined libraries and appropriate versions.

```python
# get a curated environment
env = Environment.get(
    workspace=ws,
    name="myenv",
    version=3
)
env.inferencing_stack_version='latest'

# create deployment config i.e. compute resources
aciconfig = AciWebservice.deploy_configuration(
    cpu_cores=2,
    memory_gb=2,
    tags={"data": "languages", "method": "sklearn"},
    description="Predict Language with sklearn",
)
```

- There are many preconfigured Curated Environments available under Environment section. You can select one as per your requirement or create a new one using Conda Yaml or DockerFile.





- We also need to specify deployment configurations. It includes the number of CPU cores, RAM and other metadata details. You can check out the complete list of options by exploring the AciWebService class. Here we have selected 2 cores and 2GB of RAM.

- Finally, we need to use our registered model. By default, it selects the latest version of the model, but you can specify version to select a previous one as well.

```
%%time
# get the registered model
model = Model(ws, "lang-det-rf-model")

# create an inference config i.e. the scoring script and environment
inference_config = InferenceConfig(entry_script="score.py", environment=env)

# deploy the service
service_name = "sklearn-langdet-svc-" + str(uuid.uuid4())[:4]
service = Model.deploy(
    workspace=ws,
    name=service_name,
    models=[model],
    inference_config=inference_config,
    deployment_config=aciconfig,
)

service.wait_for_deployment(show_output=True)
```

- The model deployment also called as Inference Pipeline needs to be configured using an entry script. Entry Script defines functions to initialize the model, accept the data from user and evaluate precisions to be consumed. We also need to specify the environment configured above in which the inferences will execute.
- Once all these configurations are setup, we initiate our deployment by specifying all these configurations in Model.deploy function. We also need to specify the name of service.
- As we run the above code, the model deployment initiates and takes around 15-20 minutes to complete.

Once the model has been deployed, you can use the Endpoints section to test and consume as describe in above tutorials.