Machine Learning Lab Assignment

#### (You may answer the questions with code snippets or screenshots if applicable.)

# Part 1: Train and Deploy an AutoML Model in Azure Machine Learning Studio as an API.

In this part of the lab, you’re going to train a basic machine learning model on a dataset of your choice using the Azure Automated ML service and then deploy the best model as an API endpoint.

Note: You can do any/all of the steps in the Azure ML Studio GUI or the Python v2 SDK.

You won’t need to deploy the Azure Machine Learning workspace nor the AI Foundry Project. These have been deployed in the class Resource Group.

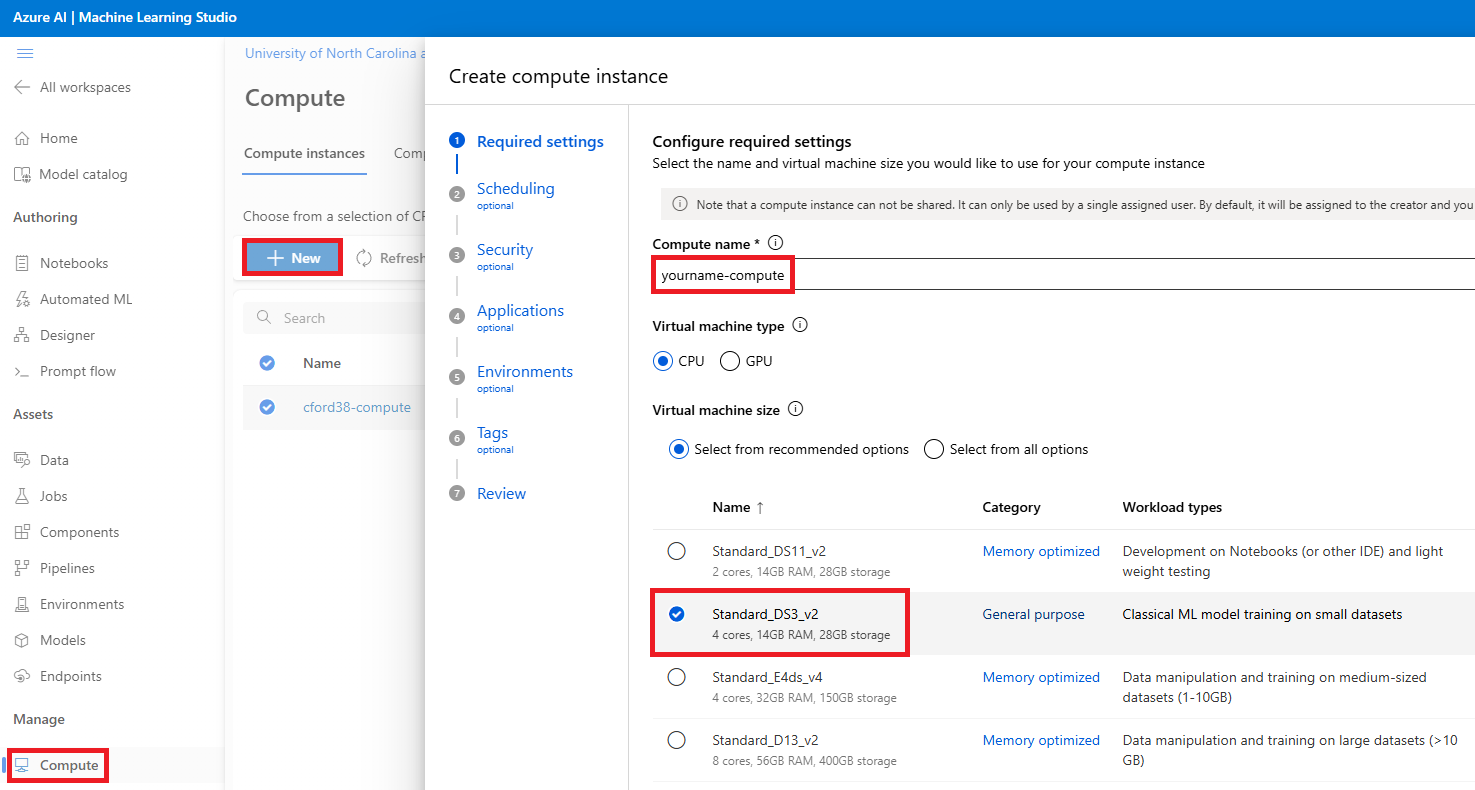
If you’re going to use the AutoML GUI, you can create a new ML job under Automated ML > +New Automated Job.

A screenshot of a computer

Description automatically generated

If you’re going to use the Python SDK (recommended), the first step is to create a Compute Instance in the Azure Machine Learning Studio. This is found under Compute > Compute instances > +New.

* Give your compute instance a name like: <YOUR NAME>-compute
* Select the Standard DS3\_v2 virtual machine size. (If you need a larger machine, ask.)
* You can leave all other settings as the default values. Make sure the *Auto shut down* feature is set to 60 minutes.



Next, create a Data Asset of your training data file. If your ML dataset is in your group’s data lake, you’ll need to create a Datastore (connecting your data lake to the Azure ML Studio). (Only 1 person from each group needs to create the Datastore link.) Alternatively, you can create a Data Asset from the GUI under Data > Data assets.

A screenshot of a computer

Description automatically generated

Once your compute instance is up and running, you can then use your desired IDE (Jupyter, VS Code, etc.) to create a Python notebook and complete the AutoML model training and deployment.

### Questions:

1. How did your AutoML process do? Did it make a decent model? Evaluate the best performing model(s) and describe their performance here. (Screenshots are great, too.)
2. Call your API endpoint using Python and score a couple rows of sample data. Provide a copy of the response from the API.

Note: Once you’ve deployed and tested your API endpoint, please delete it.

# Part 2: Play with an Azure OpenAI model and compare it to an open-source model.

This part is easy. To start, simply play with a deployed Azure OpenAI model in the Azure AI Foundry. You’ll ask the model to do something, giving it some instructions and context beforehand.

Next, find an open-source model and ask it the same sorts of queries. Try to instruct this model with the same context as in the OpenAI model for a more “apples-to-apples” comparison.

## Questions

1. Describe what you’re trying to have the LLMs do. (Screenshots of your inputs are great.)
2. Describe the models.
   1. Which Azure OpenAI model did you use?
   2. Which open-source model(s) did you use? Prove a URL link to it.
3. What are the model size differences in these models? What else is different in the model architectures/design/purpose?
4. Compare and contrast their performance on the task that you’re giving them. (Screenshots are great.)