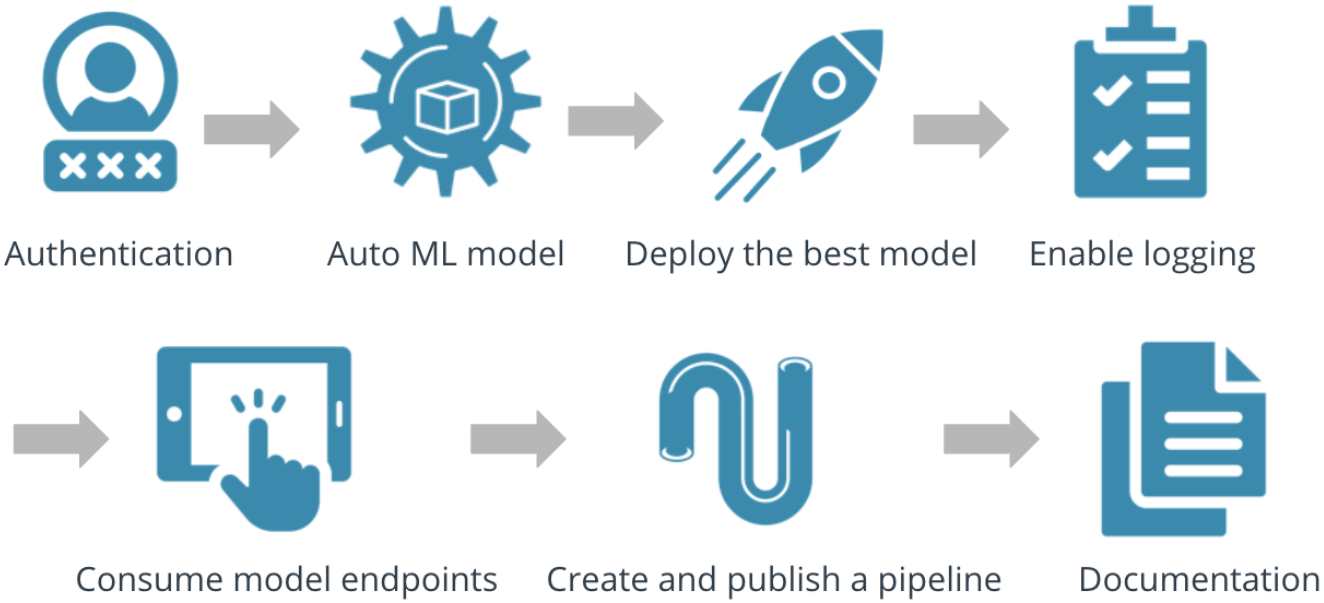
# Operationalizing Machine Learning

## Overview

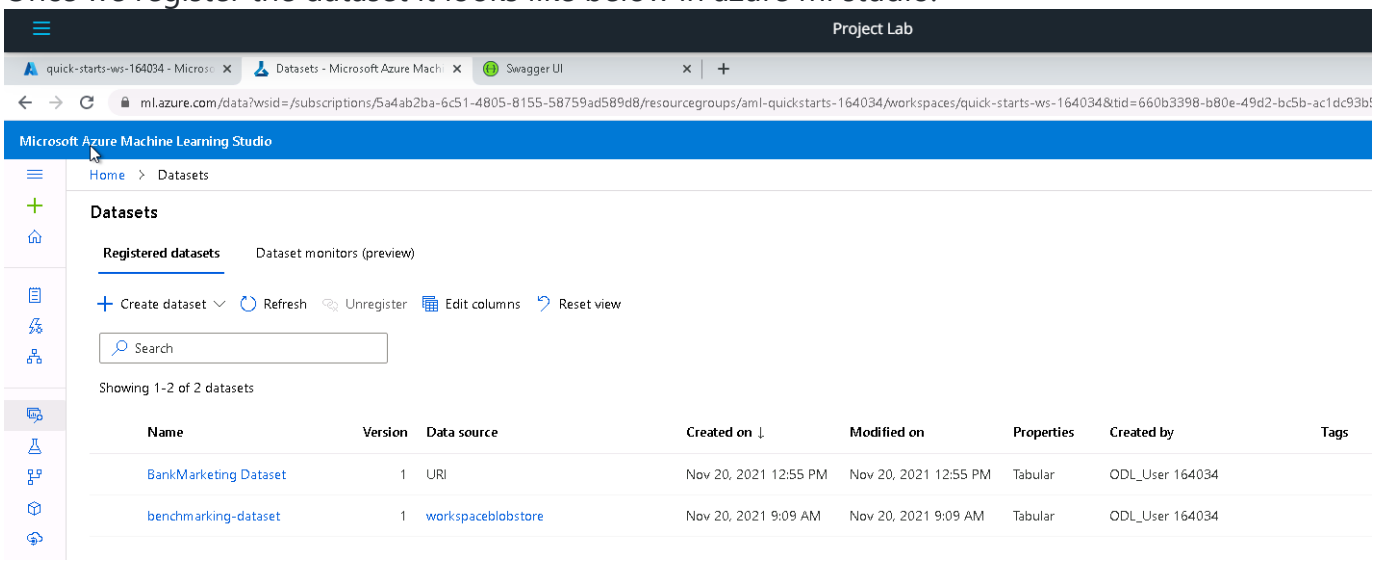
This project is part of the Azure ML Nano degree offered by Udacity. Here In this project, we use train a machine learning model with different algorithms using AutoML and also get the best performing model, and operationalize it using below workflow. In this project we demonstrated training of model using first by Azure ML experiment and then by SDK option.
Below figure shows an overview of steps involved:



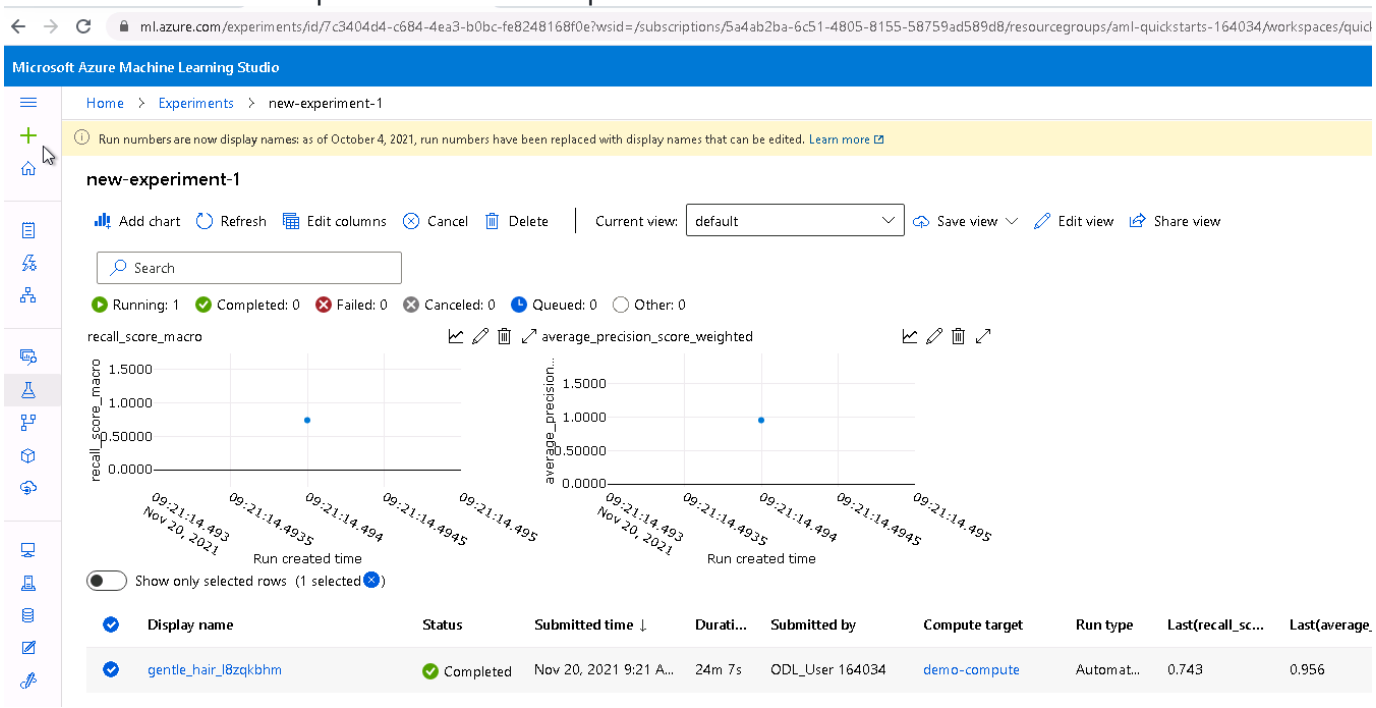

A brief description of each step involved is as follows:
As a first step we need to create a compute instance of running notebook and compute cluster for training the model. Next second step we need to upload dataset and register so that we can start AutoML experiment. Now we are ready to create a new Automated ML experiment as a third step. This experiment gives us the best model best classification model (fourth step). Next we have to deploy the best model (fifth step), once deployment is completed it gives a restful link to interact with the deployed model (sixth step). Now to see the model performance we can enable application insights (seventh step) (also we model testing has to be preform) and model documentation can be viewed using swagger (eight step) and Model is then consumed (ninth step).

# Below are the main steps:

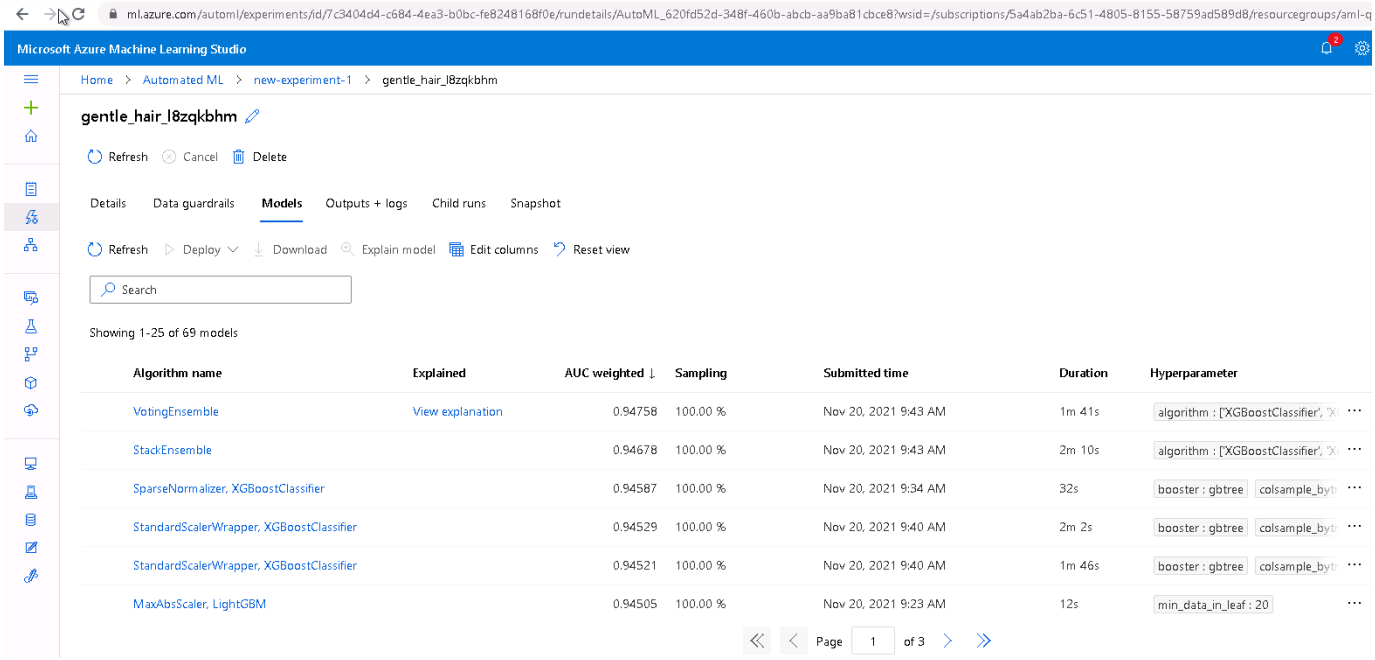
Once we register the dataset it looks like below in azure ml studio:



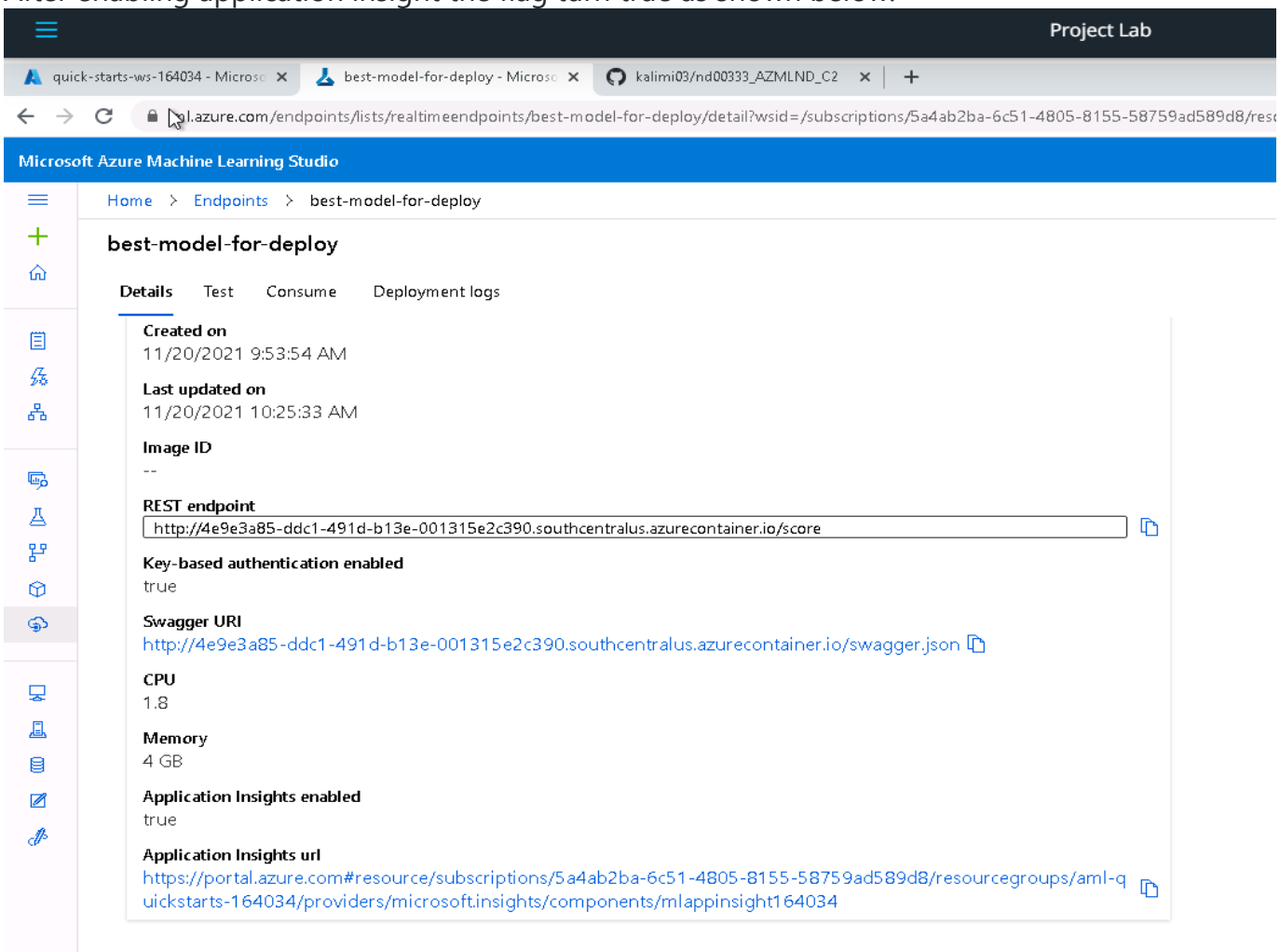

The Automated ML experiment after completion will look like this:



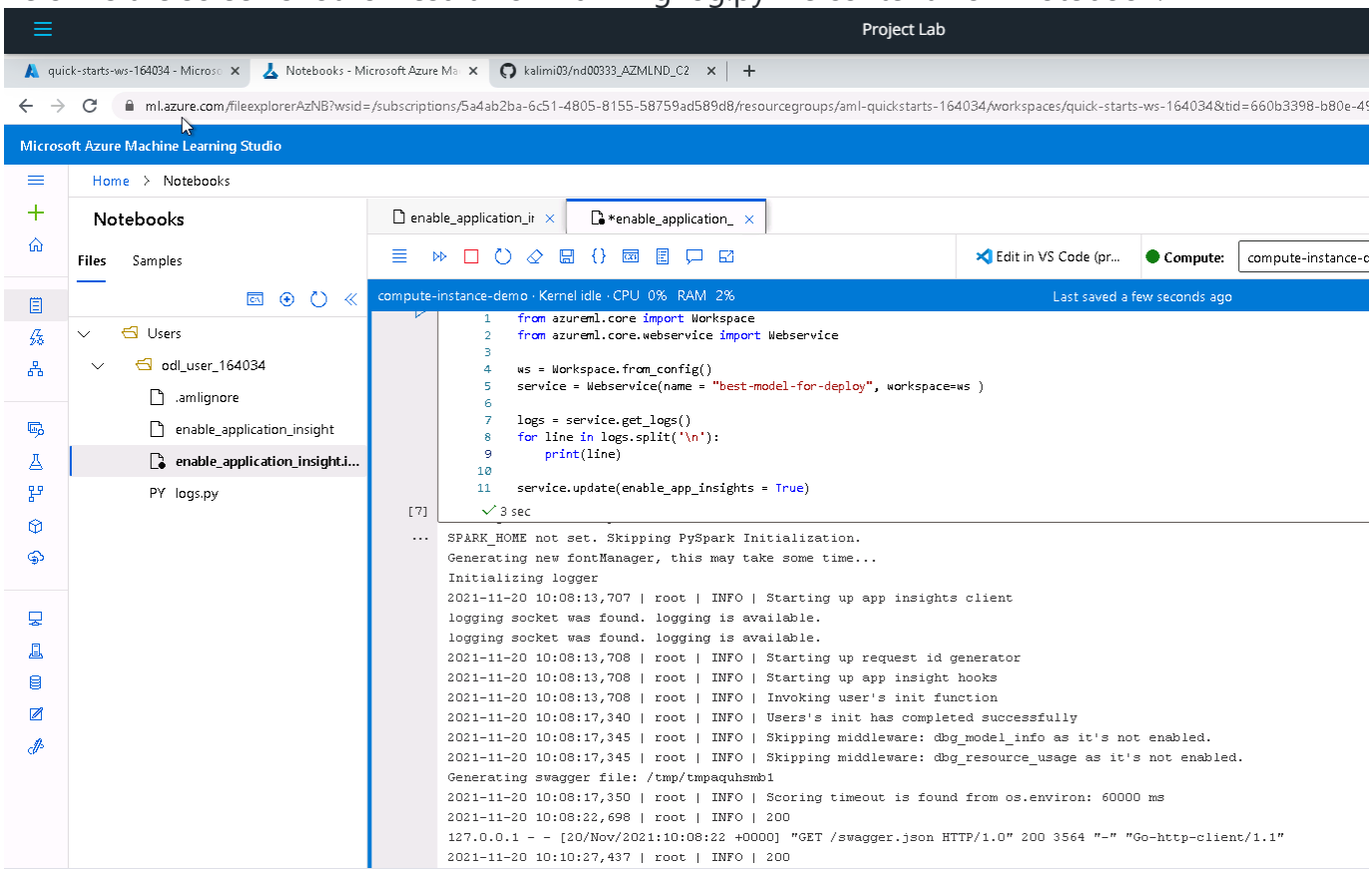
The best performing model is the Voting Ensemble:



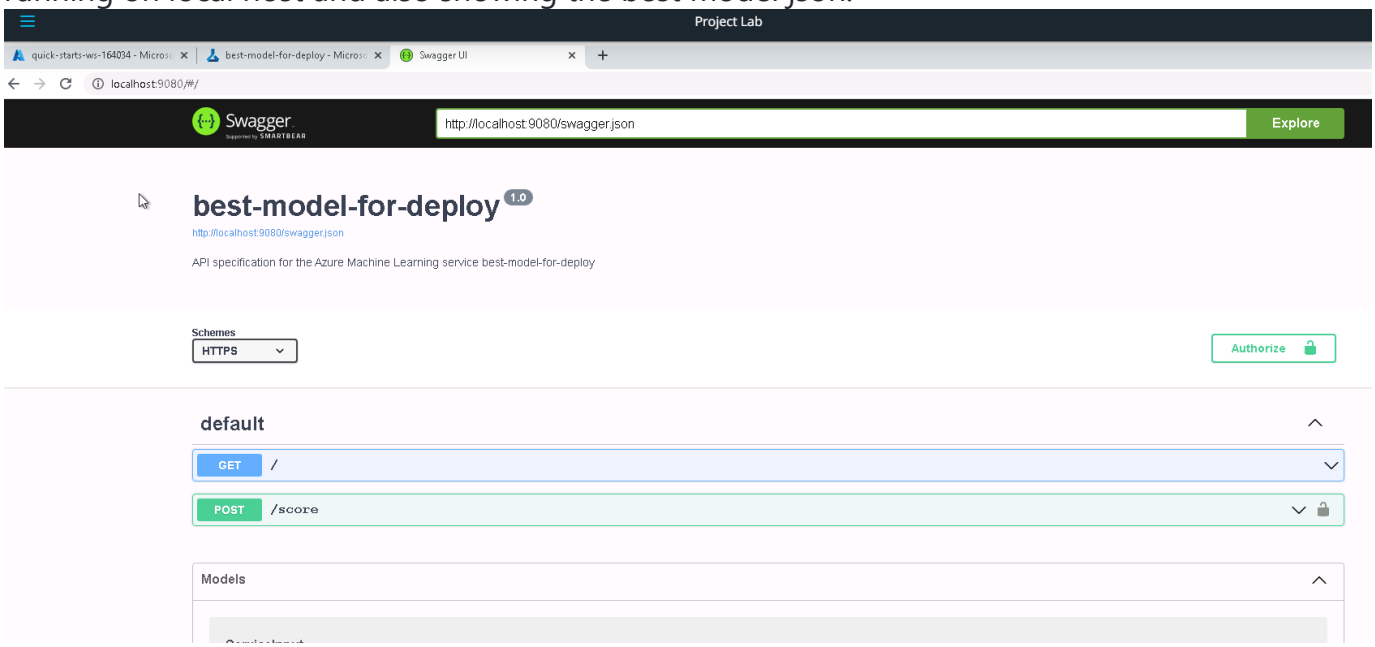

After enabling application insight the flag turn true as shown below:

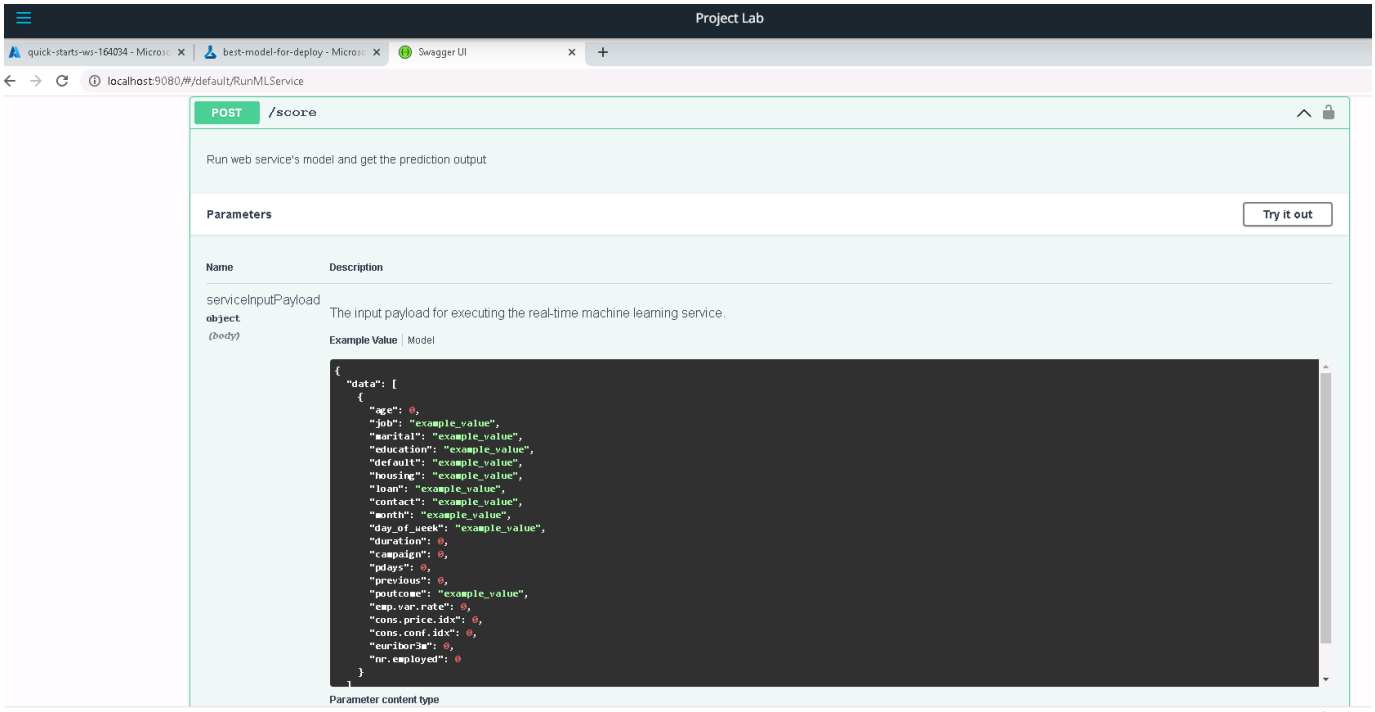


Below is the screen shot for result from running log.py file content from notebook:

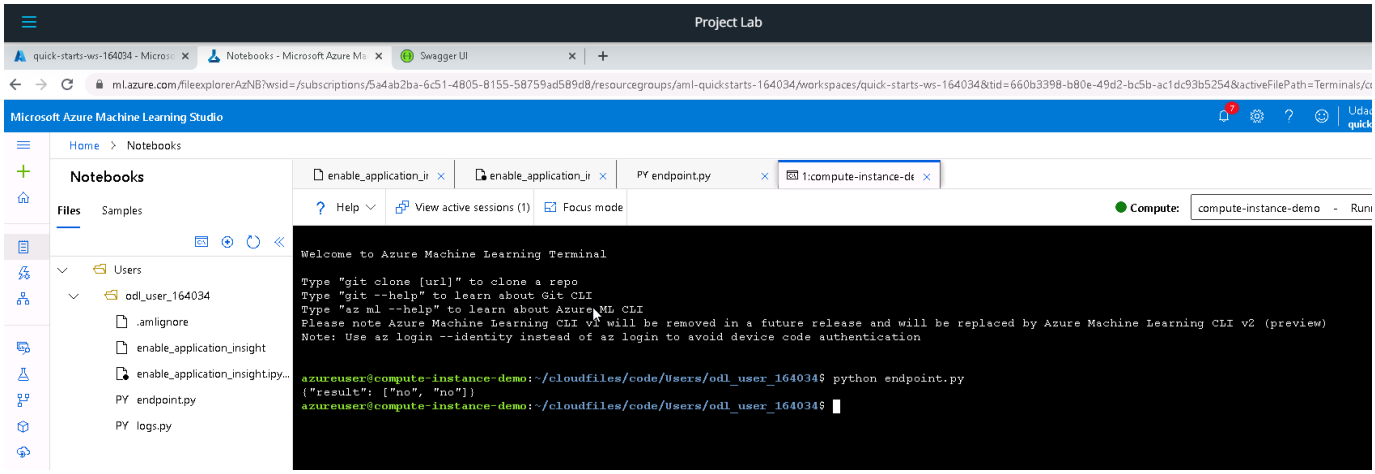



Swagger is basically used for communicating rest apli endpoint documentation a json file. The bash file swagger.sh is used to run a swagger docker image and serve.py make swagger json file available on an local host server. Below figures shows docker image (or swagger-ui.html) running on local host and also showing the best model json:

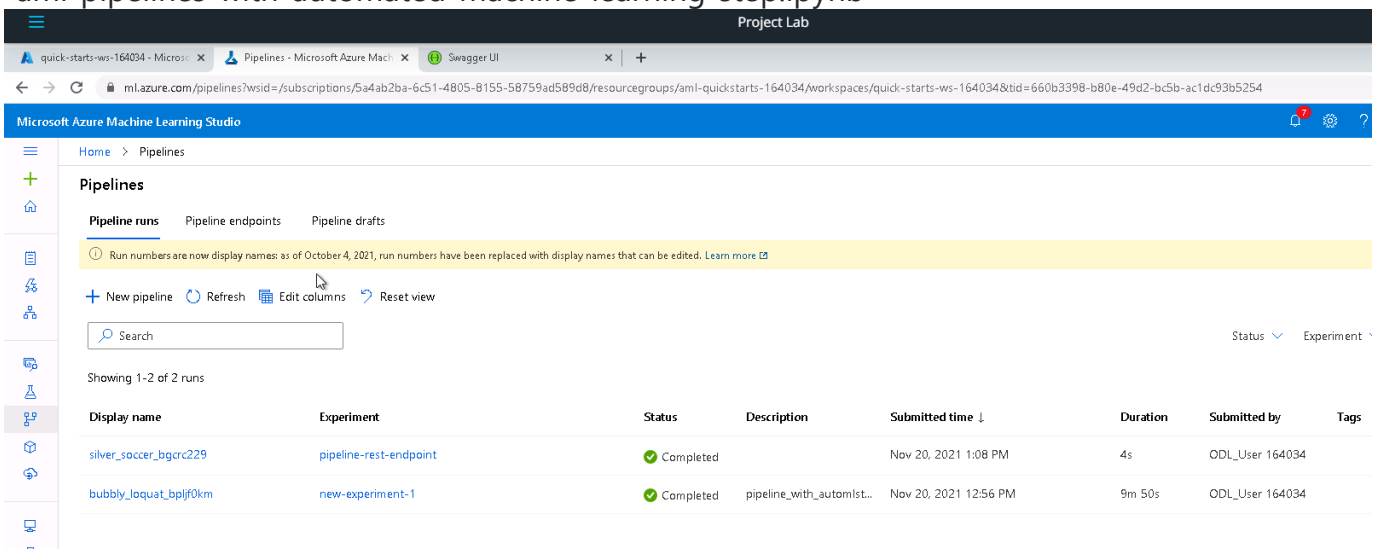




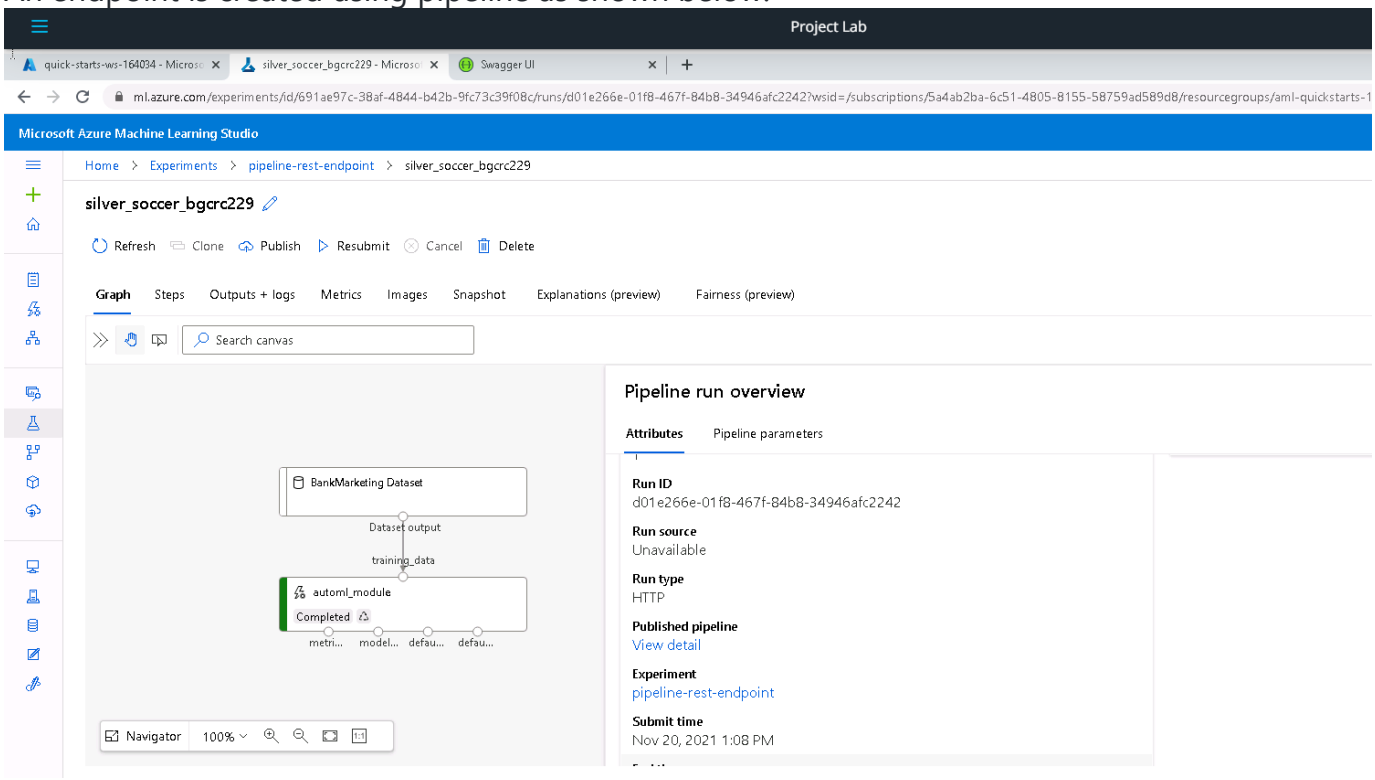

We use endpoint.py file for deployed model consumption. When a json format document is passed on to the rest api endpoint using POST method, then the model processes it and gives the outcome as show below:



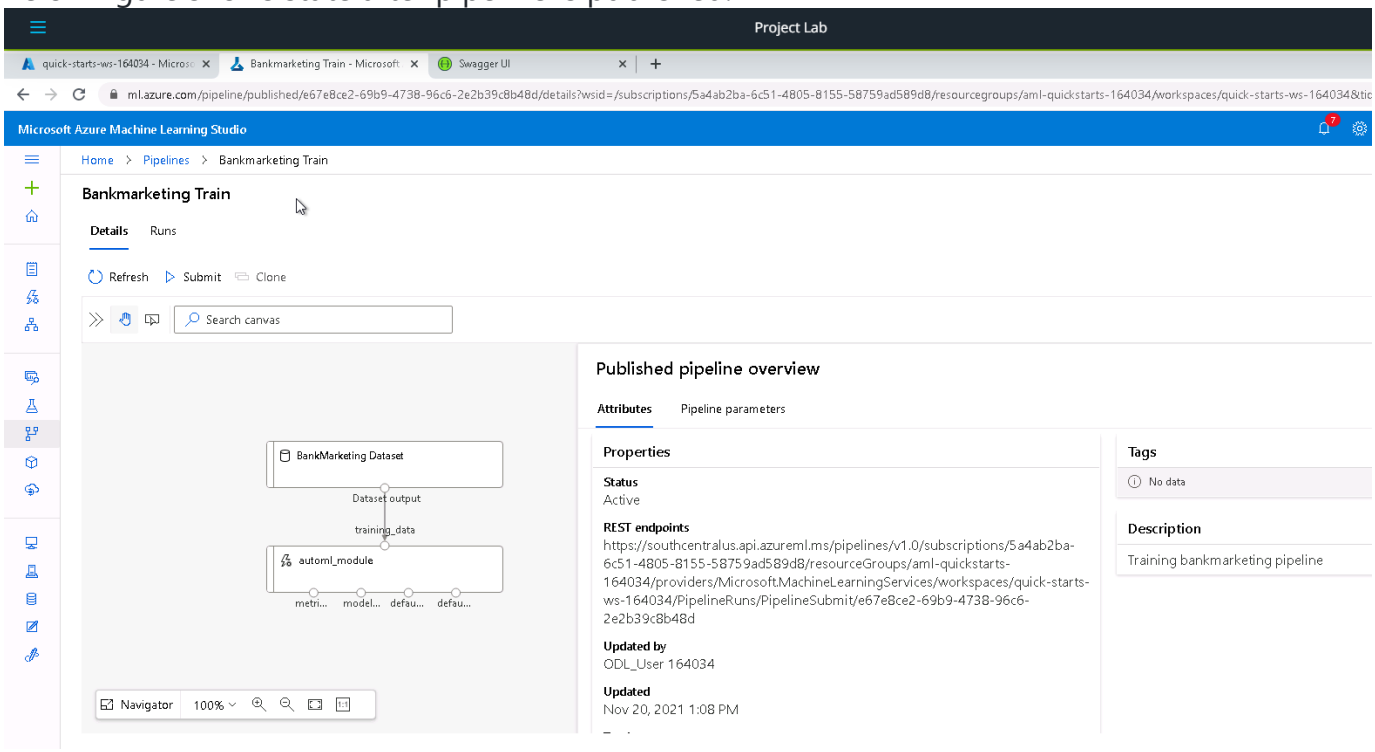
Now creating a pipeline for the entire process using Jupyter Notebook: “aml-pipelines-with-automated-machine-learning-step.ipynb”



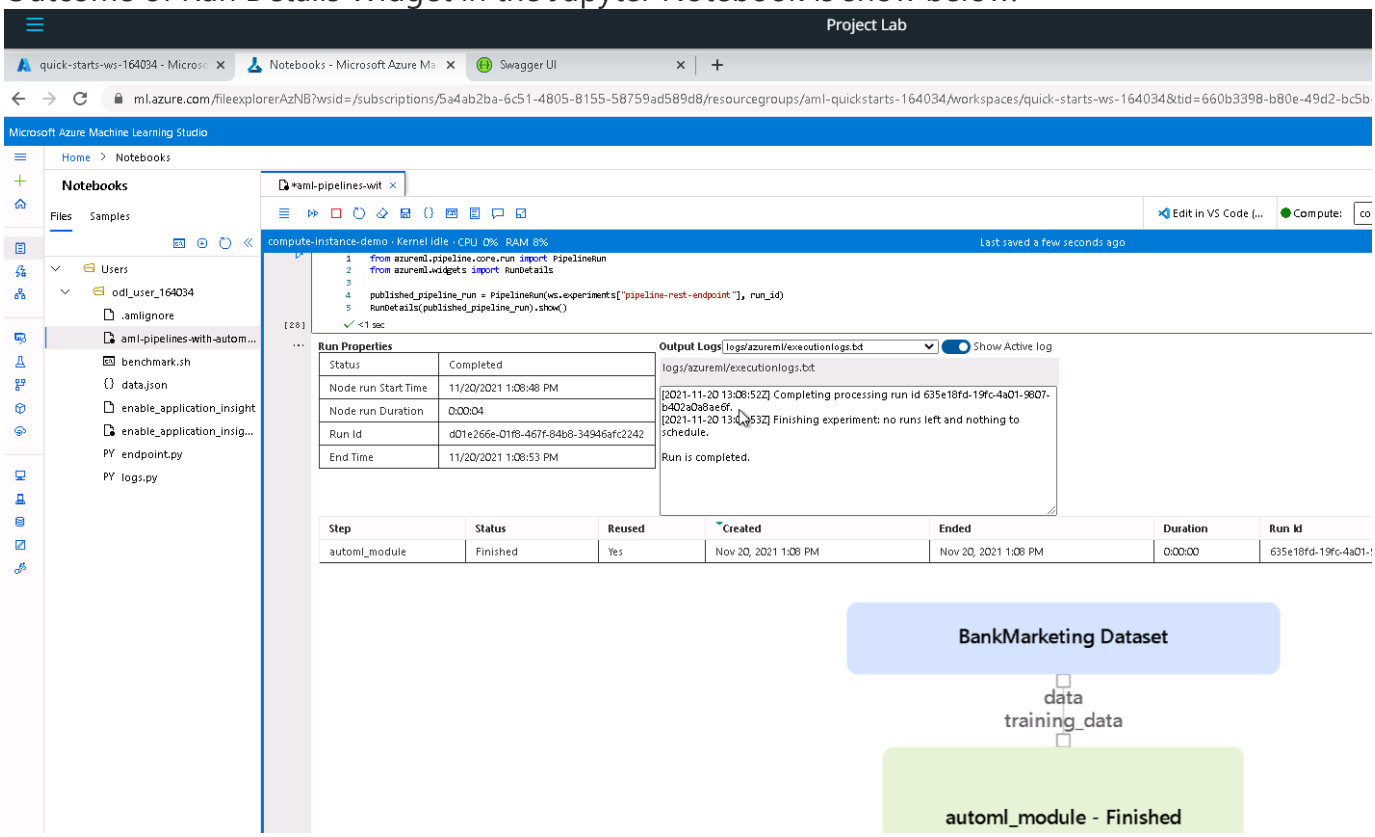

An endpoint is created using pipeline as shown below:



Below figure shows state after pipeline is published:




Outcome of Run Details Widget in the Jupyter Notebook is show below:



## Further work:

1. Use of Deep Learning: This may provide a better performance, since image pixel correlation information is better utilized in case deep learning.
2. Increasing the dataset size will also help.
3. Adding a CI/CD pipeline to interact with the Published Pipeline will be useful
4. Maybe data preprocessing (in case of deep learning) will boost the performance.

## Screen Recording

You can view the screencast of all the steps mentioned above at this:

https://www.youtube.com/watch?v=6afnf27xvfI