**# Optimizing an ML Pipeline in Azure**

**## Overview**

This project is part of the Udacity Azure ML Nanodegree.

In this project, we build and optimize an Azure ML pipeline using the Python SDK and a provided Scikit-learn model.

This model is then compared to an Azure AutoML run.

**## Useful Resources**

- [ScriptRunConfig Class](https://docs.microsoft.com/en-us/python/api/azureml-core/azureml.core.scriptrunconfig?view=azure-ml-py)

- [Configure and submit training runs](https://docs.microsoft.com/en-us/azure/machine-learning/how-to-set-up-training-targets)

- [HyperDriveConfig Class](https://docs.microsoft.com/en-us/python/api/azureml-train-core/azureml.train.hyperdrive.hyperdriveconfig?view=azure-ml-py)

- [How to tune hyperparamters](https://docs.microsoft.com/en-us/azure/machine-learning/how-to-tune-hyperparameters)

**## Summary**

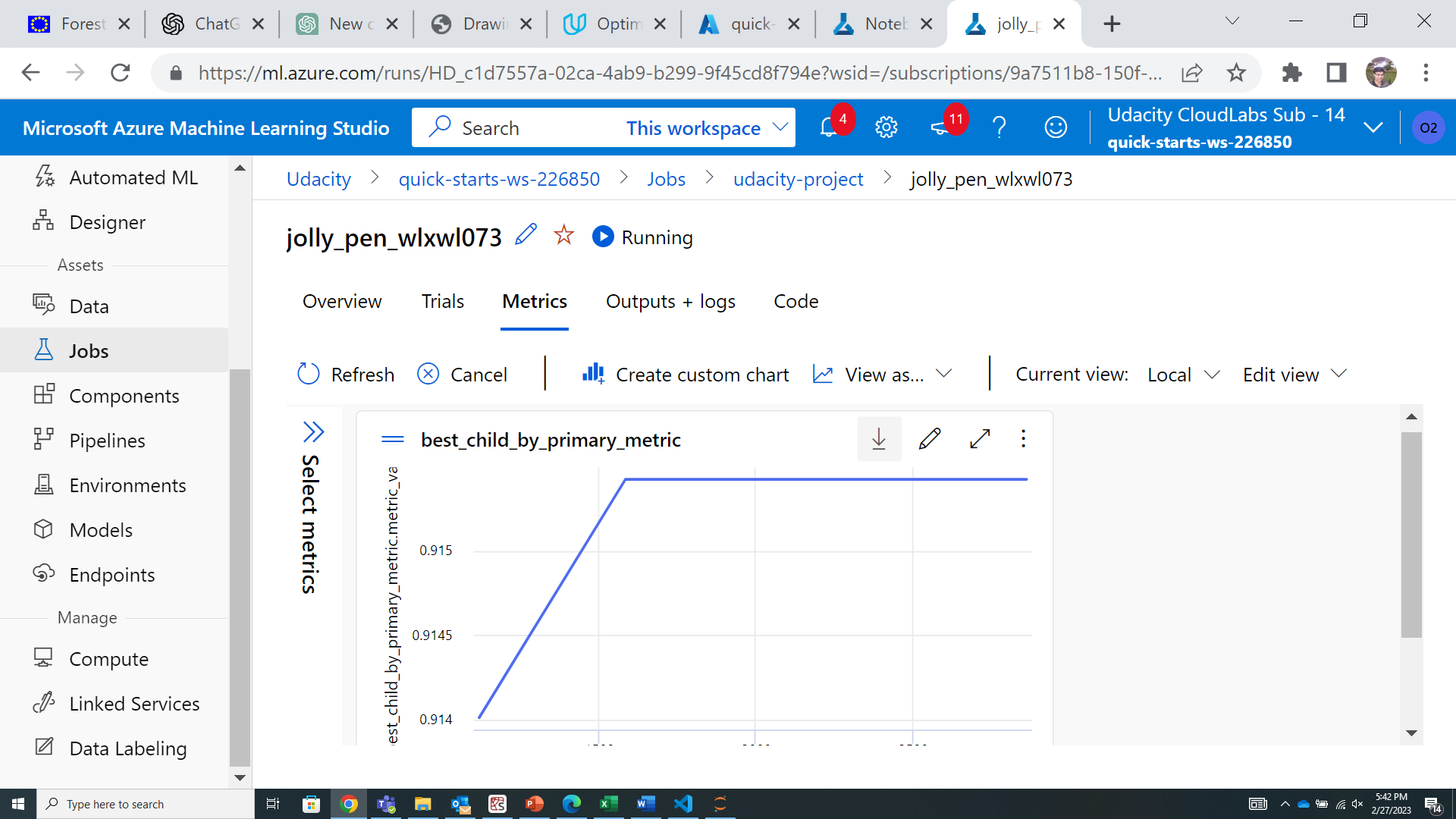
This dataset contains data about different customers of a bank/ The task is to predict whether a customer will subscribe to a deposit. The target variable 'y' is binary

**## Scikit-Learn training pipeline**

1. Setup the train and test data
   * Create dataset using TabularDataSet factory
   * Clean the data i.e. one-hot encoding of categorical features and target, date processing and remove null values
   * Divide data into train and test sets
2. Create the scikit learn estimator. It is a logistic regression model. The estimator takes 2 parameters as input which were ‘C’: regularization strength and ‘iter’ : number of iterations
3. Create a sklearn\_env from Environment.from\_conda\_specification using a yml file. This sklearn\_env will be passed to hyperdrive configuration in the next step
4. Configuring Hyperdrive using HyperDriveConfig: The parameter sampler was random sampling and policy was bandit policy. The primary metric was accuracy and goal was to meaximize this metric
5. Saving the final optimized model

The Random parameter sampler samples the parameter space considerably quicker as compared to grid sampling in case of big dimensional spaces. This can be very useful to get an initial good estimate of the parameters which can be further optimized later.

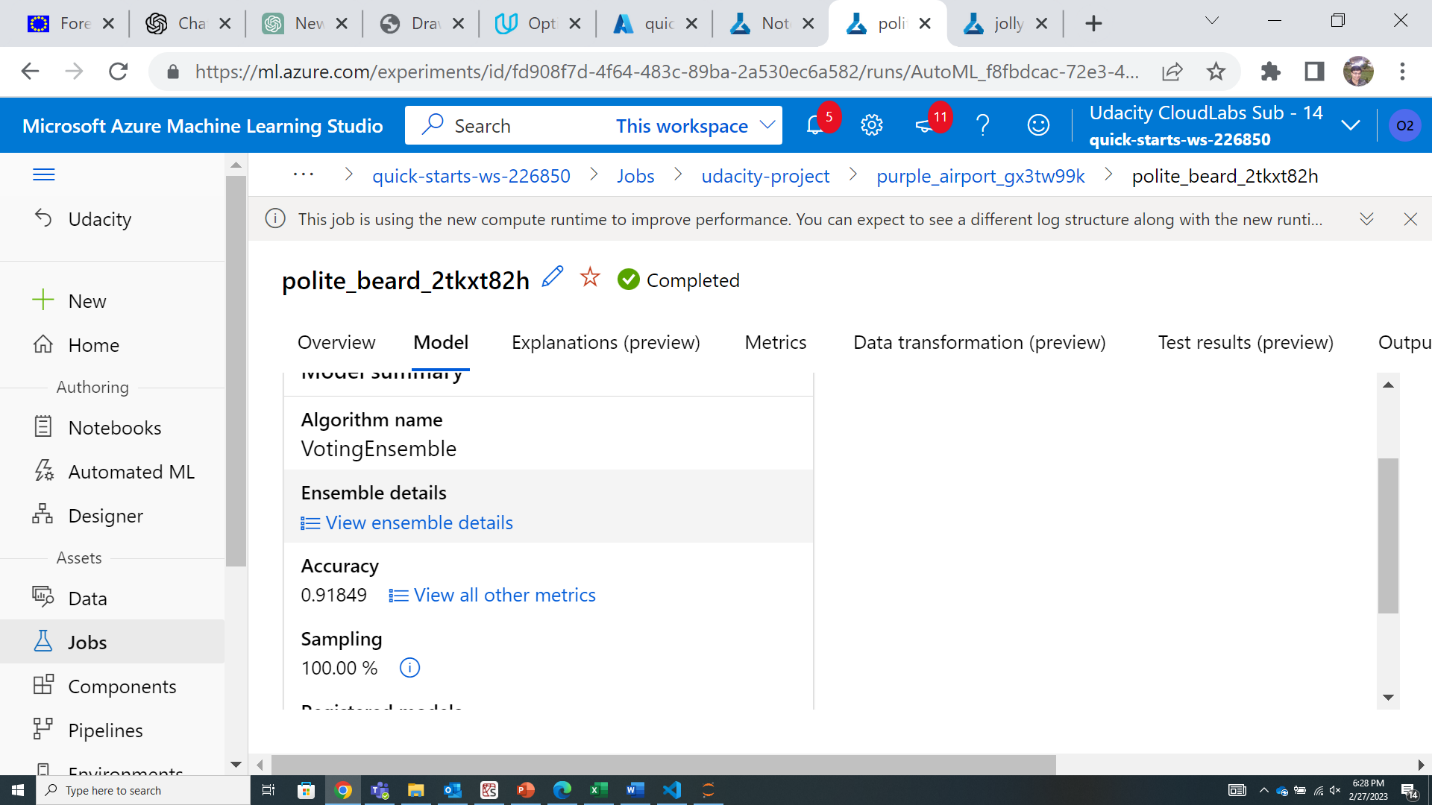
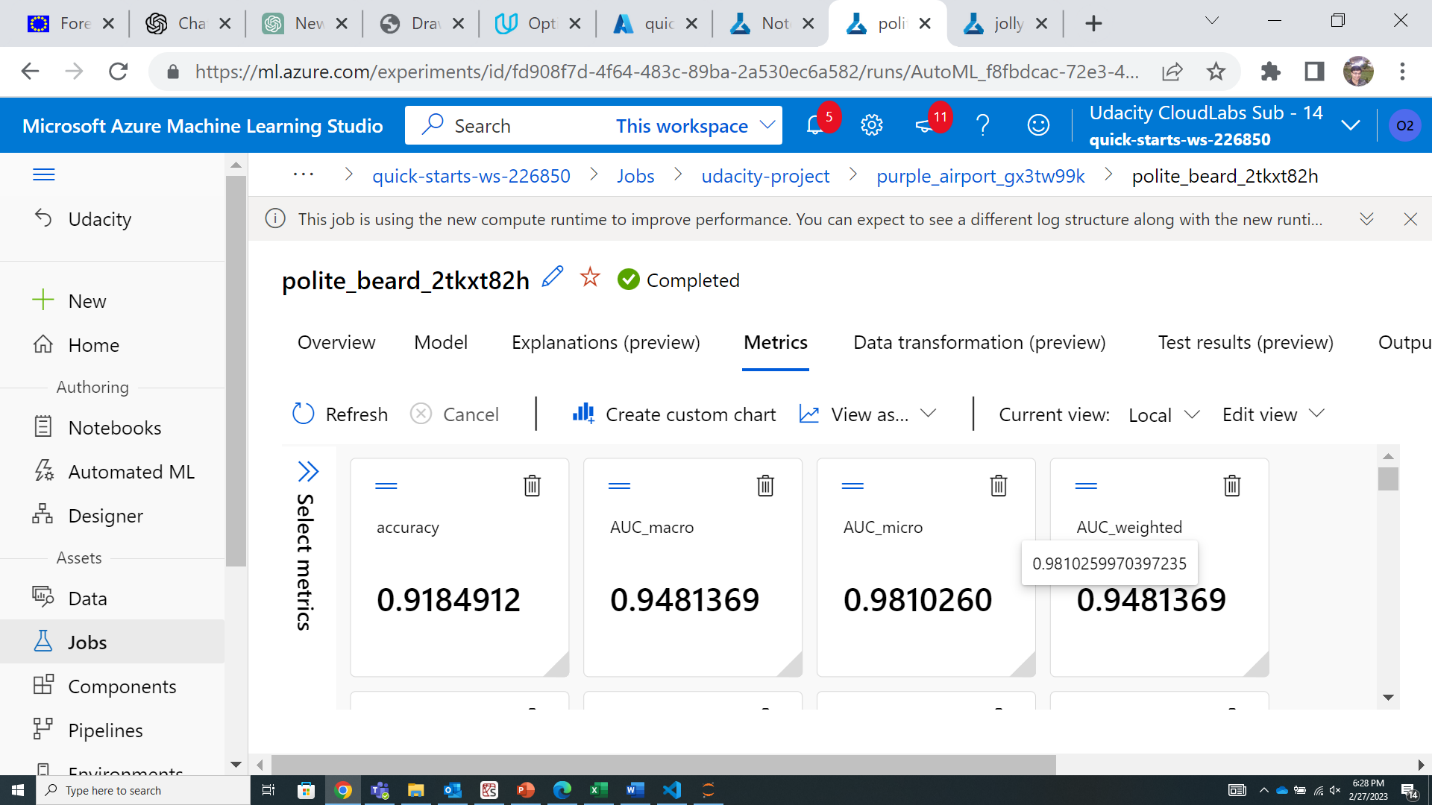
The early stop policy is the Bandit policy in the code. This policy terminates any run that is not within the best metric +- slack\_factor\*best\_metric after the iterations in the run exceed the evaluation\_interval



Best metric 0.9154 from hyperdrive run

**## AutoML**

Voting ensemble model generated by AutoML. Acuracy=0.9184192



**## Pipeline comparison**

AutoML performed better in terms of accuracy. AutoML had an accuracy of 0.9184 vs 0.9154 accuracy from hyperdrive optimized logistic regression

The voting ensemble model proposed by AutoML is an ensemble of different algorithms like XGBoost, LightGBM, Logistic Regression models with different weights. Each of these constituent models have different hyperparameters.

The hyperdrive optimized only a Logistic Regression model. Therefore, it is evident that a combination of models with varying weights improves accuracy in case of Voting Ensemble vs only Logistic regression model.

**## Future work**

Bayesian parameter sampling can be used instead of random parameter sampling. Bayesian parameter sampling uses the previous samples and accuracy at those sampling points to decide on future sampling points thus improving efficiency of the sample search.

XGBoost classifier can be used instead of Logistic regression in the hyperdrive optimization