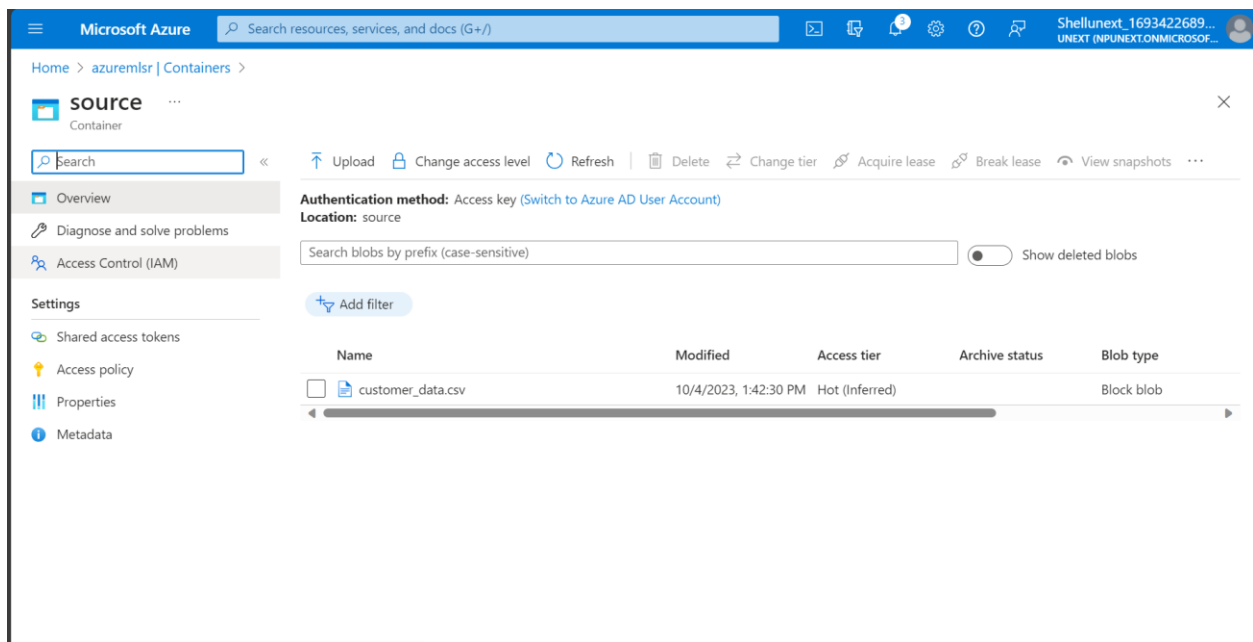


By Pagidala Saketh Reddy

Employee ID: 655074

## Data Preparation

A new storage account is created and customer\_data is stored in a container inside that storage account.



Created a data asset in azure ML that is connected to above storage container and is linked to the dataset in it

Create data asset

✓ Data type

✓ Data source

✓ Source storage type

4 Storage path

5 Review

Choose a storage path

Navigate to or enter the storage path you want to use for this data asset.

☒ Browse to storage path ☐ Enter storage path manually

Selected path: customer\_data.csv

↻

↑

Filter...

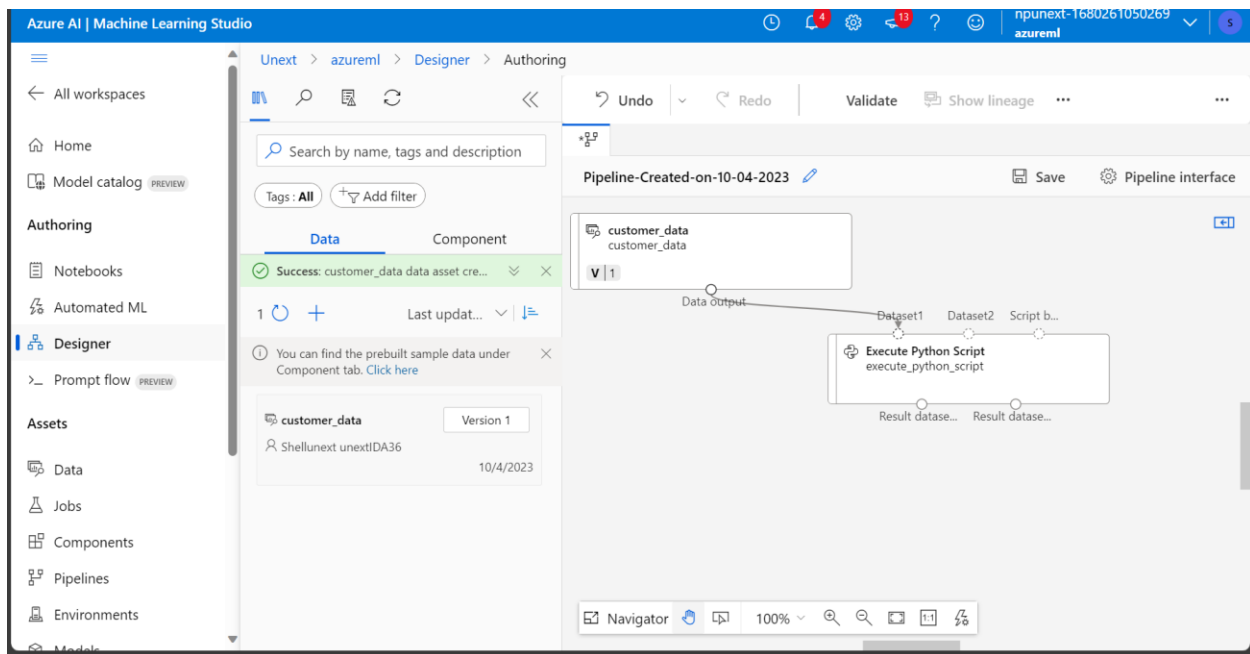
	Name	Created on	Modified on
✓	customer_data.csv	Oct 4, 2023 1:42 PM	Oct 4, 2023 1:42 PM

Back

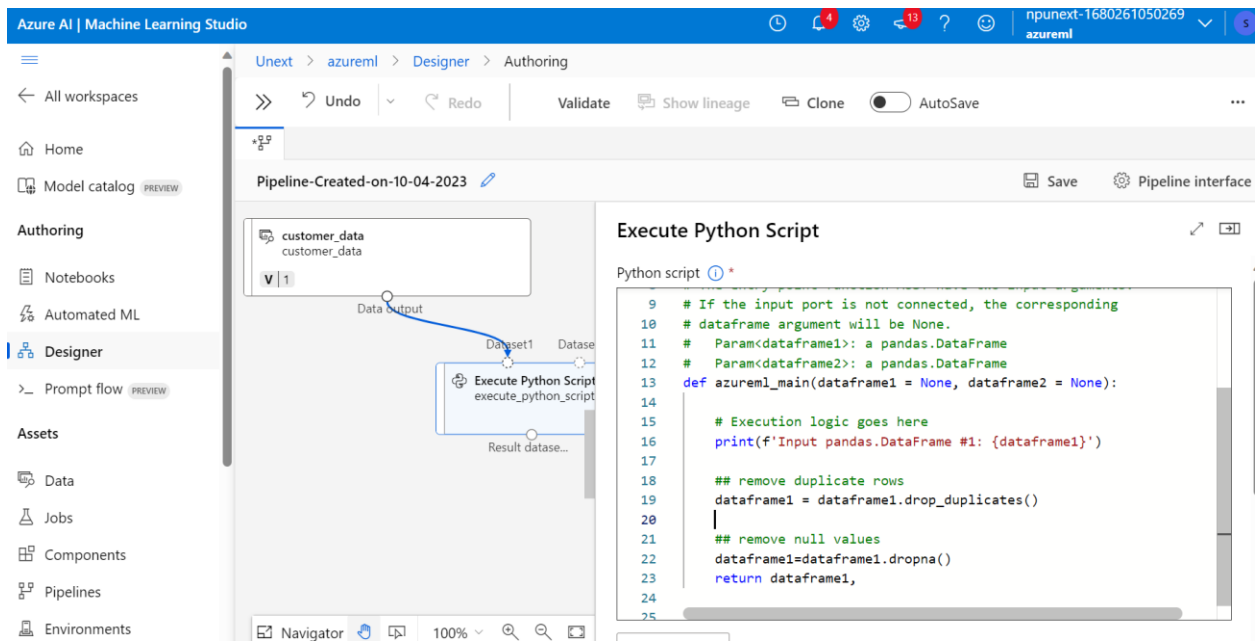
Next

Cancel

## Connected the data asset to the python script

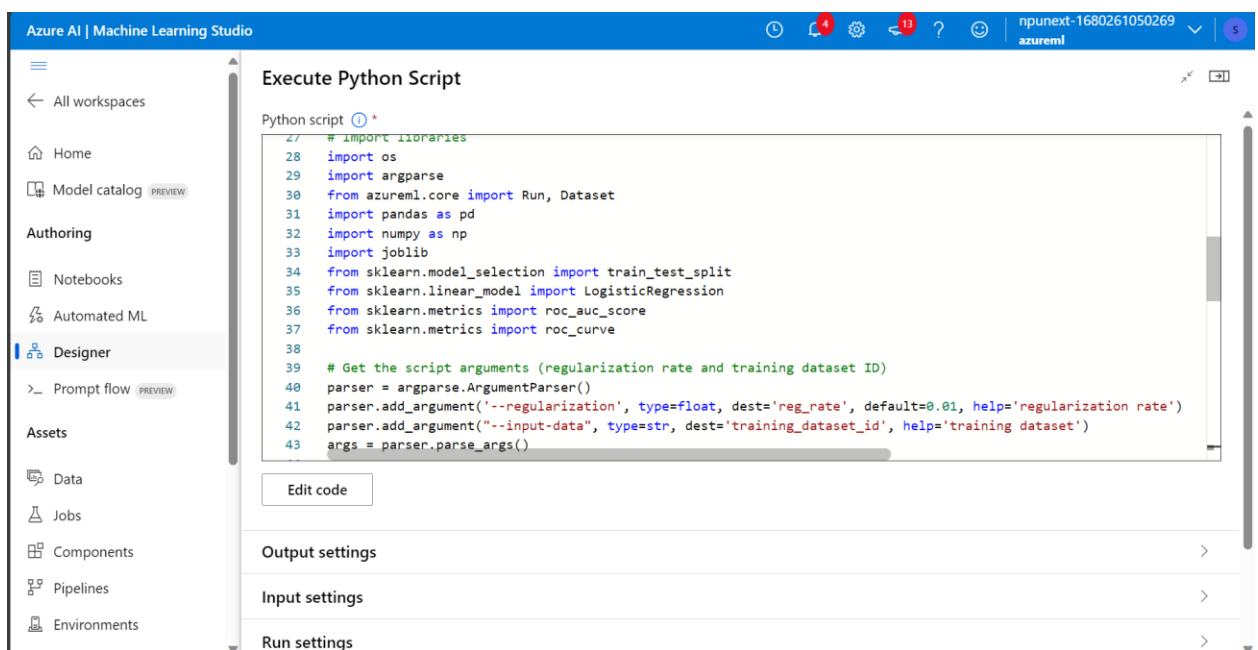


## Cleaning the data by removing duplicate rows and removing the null values

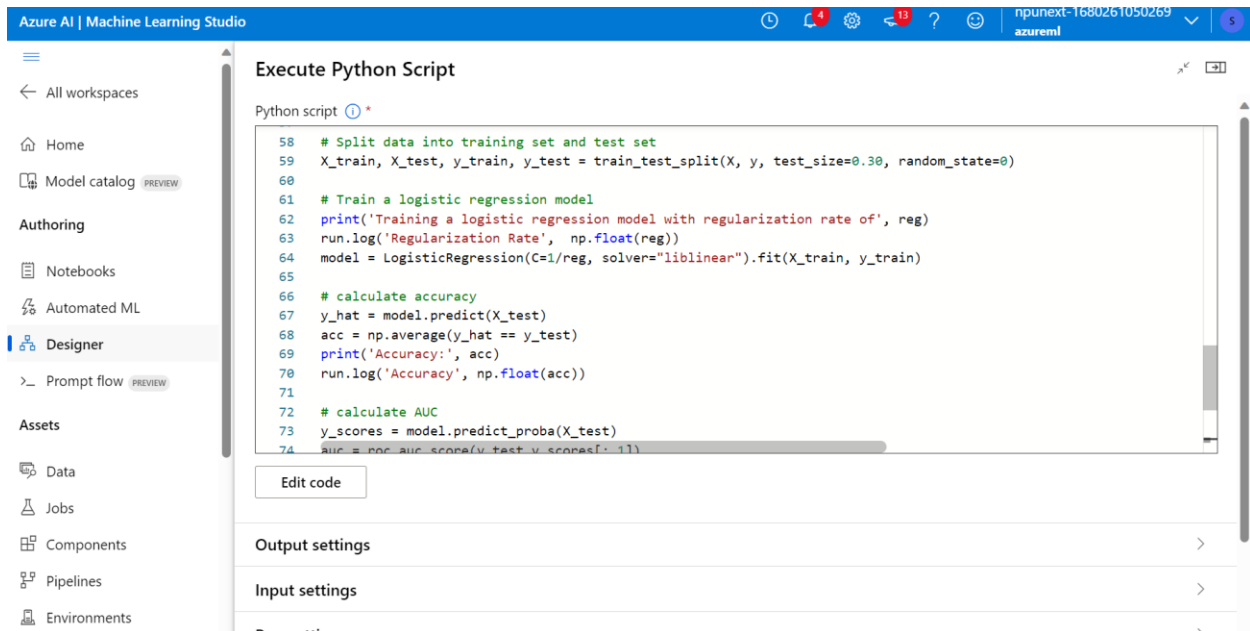


## Model Development

Importing the necessary libraries for the model



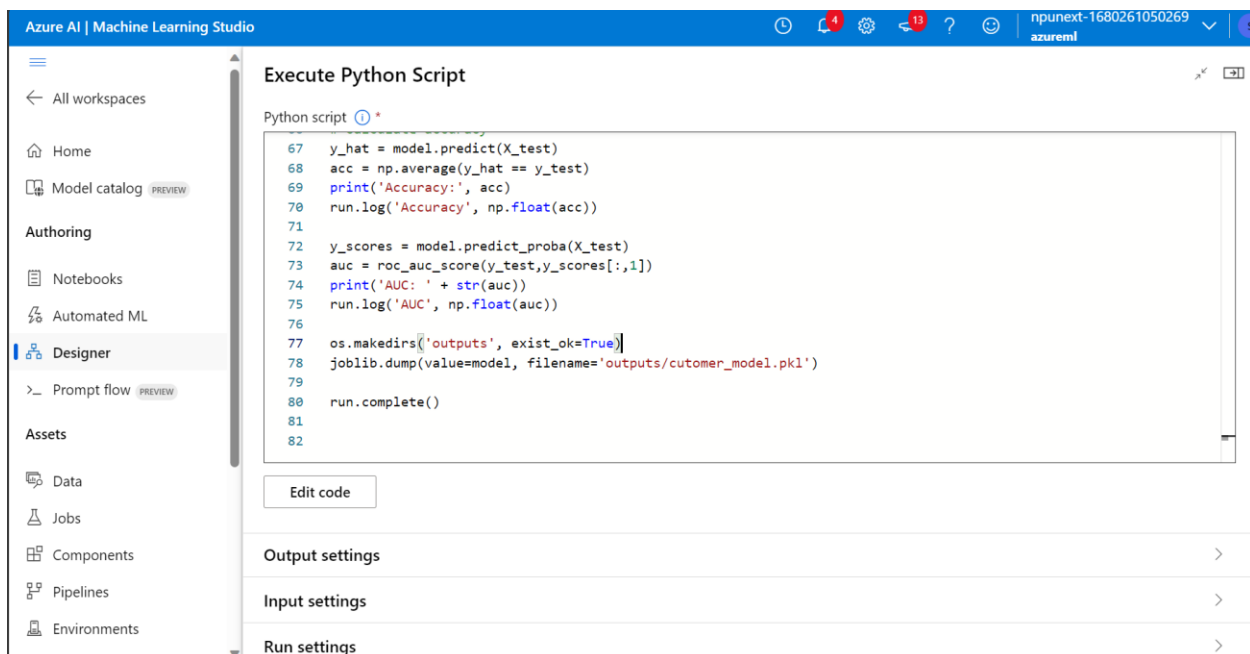
Splitting the dataset into 70:30 ratio where training is 70% and testing is 30%. Creating a logistic regression model with X\_train and Y\_train as parameters. Calculating the accuracy



The screenshot shows the Azure ML Studio interface with the 'Execute Python Script' notebook. The left sidebar contains navigation options like 'All workspaces', 'Home', 'Model catalog', 'Authoring', 'Notebooks', 'Automated ML', 'Designer', 'Assets', 'Data', 'Jobs', 'Components', 'Pipelines', and 'Environments'. The main area displays a Python script for training a logistic regression model and calculating its accuracy.

```
58 # Split data into training set and test set
59 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
60
61 # Train a logistic regression model
62 print('Training a logistic regression model with regularization rate of', reg)
63 run.log('Regularization Rate', np.float(reg))
64 model = LogisticRegression(C=1/reg, solver="liblinear").fit(X_train, y_train)
65
66 # calculate accuracy
67 y_hat = model.predict(X_test)
68 acc = np.average(y_hat == y_test)
69 print('Accuracy:', acc)
70 run.log('Accuracy', np.float(acc))
71
72 # calculate AUC
73 y_scores = model.predict_proba(X_test)
74 auc = roc_auc_score(y_test, y_scores[:, 1])
```

Run the model and save it as Customer\_model pickle



The screenshot shows the same Azure ML Studio interface, but the Python script is updated to save the trained model as a pickle file. The code includes the same training and accuracy calculation steps as the previous notebook, followed by saving the model to a file named 'customer\_model.pkl' in the 'outputs' directory.

```
67 y_hat = model.predict(X_test)
68 acc = np.average(y_hat == y_test)
69 print('Accuracy:', acc)
70 run.log('Accuracy', np.float(acc))
71
72 y_scores = model.predict_proba(X_test)
73 auc = roc_auc_score(y_test, y_scores[:, 1])
74 print('AUC: ' + str(auc))
75 run.log('AUC', np.float(auc))
76
77 os.makedirs('outputs', exist_ok=True)
78 joblib.dump(value=model, filename='outputs/customer_model.pkl')
79
80 run.complete()
81
82
```

**Hyper Parameter Tuning**

## Assessment Questions

**1. What are the key steps involved in preparing the dataset for training a machine learning model using Azure Machine Learning? Briefly explain each step.**

**Answer:**

1. Collect data from various sources.
2. Explore and visualize the dataset.
3. Clean the data by handling missing values, duplicates, and errors.
4. Transform data by encoding categorical features and scaling numerical ones.
5. Augment data if necessary (for images or text).
6. Balance the dataset for class imbalance.
7. Select relevant features and split the data into training, validation, and test sets.
8. Label and annotate data for supervised learning.

**2. Why is it important to split the dataset into training and testing sets when developing a machine learning model? How does this help in model evaluation?**

Model Evaluation:

It provides a way to assess how well your model generalizes to unseen data. The testing set acts as a proxy for new, real-world data that the model will encounter after deployment.

Preventing Overfitting:

Splitting the data helps detect overfitting, where the model learns to memorize the training data rather than learning general patterns. If you didn't have a separate test set, you might mistakenly think your model performs well when it only memorizes the training examples.

Hyperparameter Tuning:

You can use the testing set to tune hyperparameters (e.g., learning rate, regularization strength) without using the final evaluation set. This ensures that your model's performance is not optimized based on the evaluation set, preventing data leakage.

Bias and Variance Trade-off:

Splitting the dataset allows you to evaluate the trade-off between bias and variance. A model that performs well on the training set but poorly on the testing set has high variance (overfitting), whereas a model that performs poorly on both sets has high bias (underfitting).

**3. Describe a machine learning algorithm suitable for predicting customer purchasing behavior in the given scenario. Explain why you chose this algorithm.**

Certainly, here are a few more reasons why Linear Regression is a suitable choice for predicting customer purchasing behavior:

Assumption of Linearity:

Linear regression assumes that the relationship between the input features and the target variable is linear. In many cases, this assumption holds true for aspects of customer behavior. For example, there is often a linear relationship between a customer's income and their spending habits.

Speed and Efficiency:

Linear regression is computationally efficient and can handle large datasets without significant computational overhead. This makes it practical for analyzing customer data, which can often involve a large number of records.

**4. What is hyperparameter tuning, and why is it important in machine learning? Explain a technique used for hyperparameter tuning and its benefits.**

Hyperparameter tuning is the process of optimizing the hyperparameters of a machine learning model to improve its performance on a given task. Hyperparameters are parameters that are set before training begins and control aspects of the learning process, such as the learning rate, the number of hidden layers in a neural network, or the depth of a decision tree. Tuning these hyperparameters is essential because they significantly impact a model's performance.

One common technique for hyperparameter tuning is Grid Search:

Grid Search: In grid search, you define a grid of hyperparameter values that you want to explore. The technique systematically evaluates the model's performance with different combinations of hyperparameters by training and validating the model multiple times. It exhaustively searches all possible hyperparameter combinations within the specified grid.