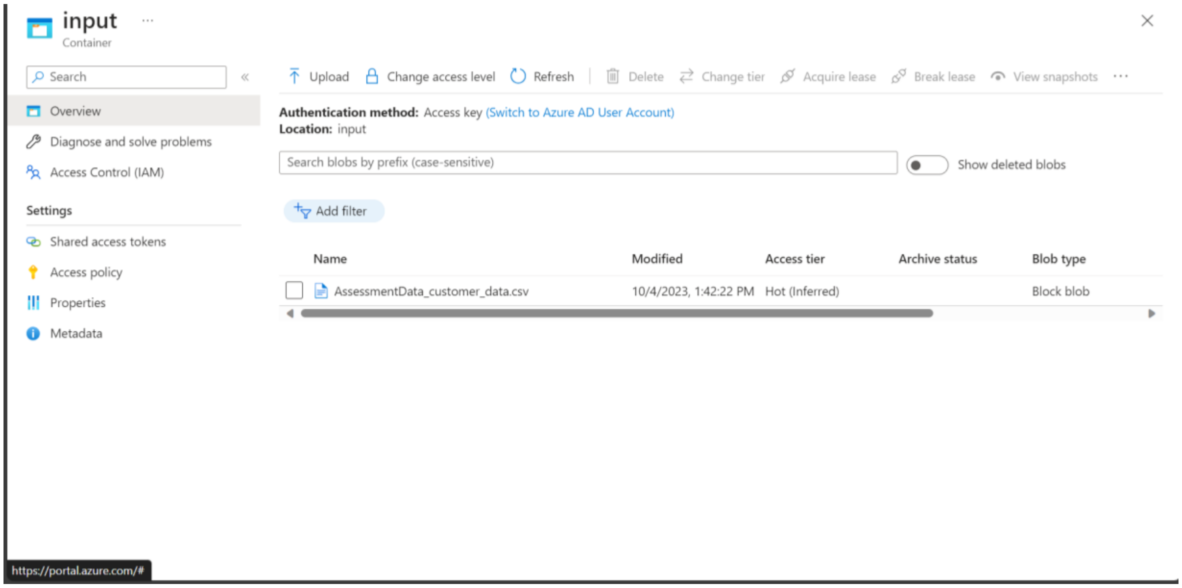
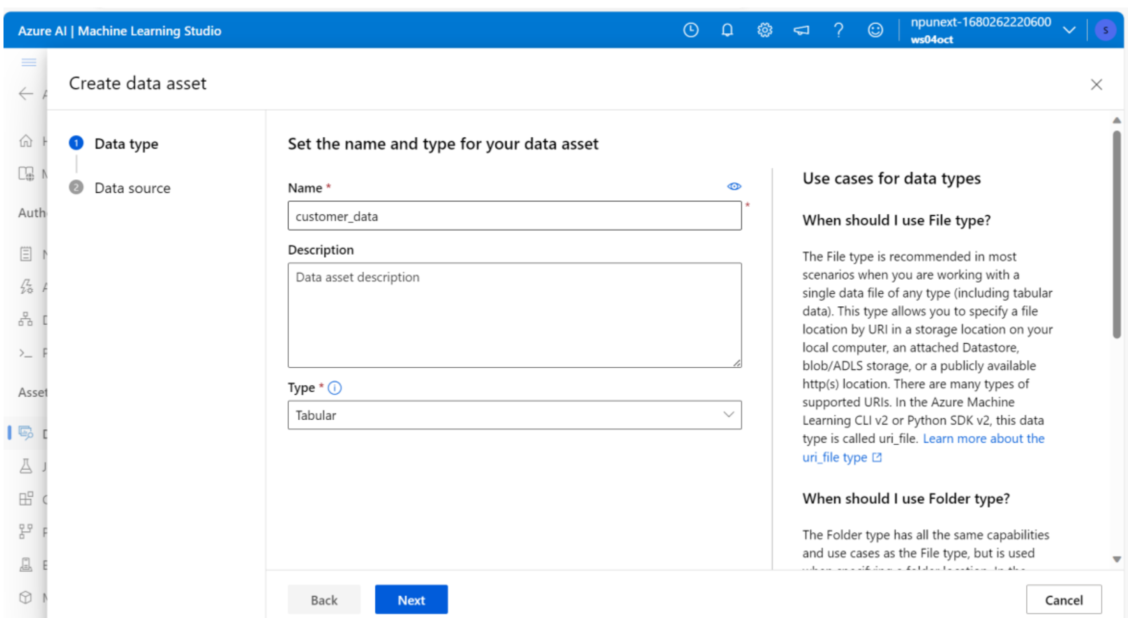
Purushottam Behera

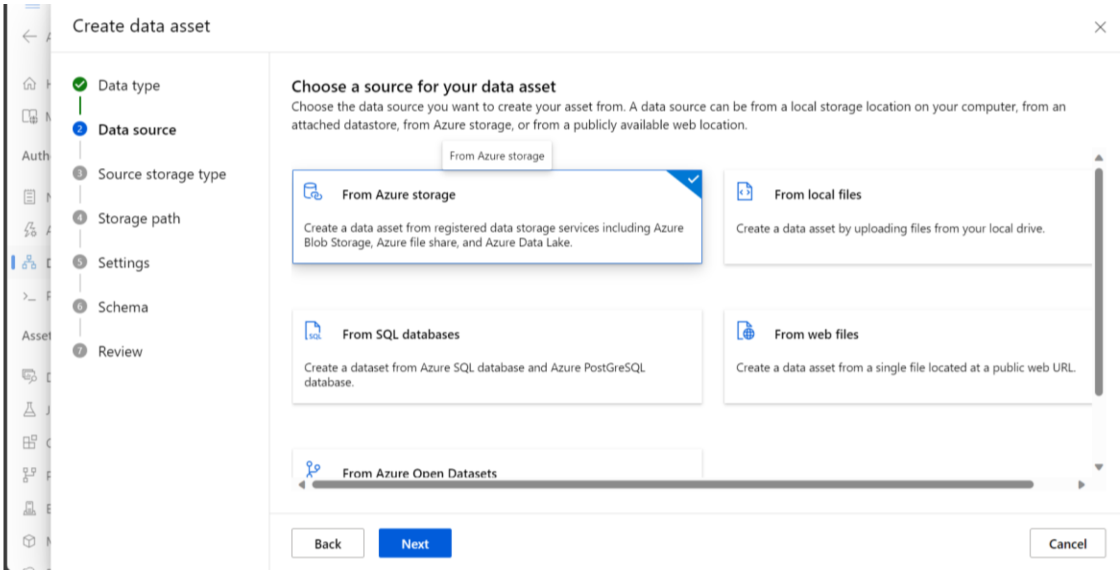
Azure ML Studio

UPLOADING DATA INTO STORAGE ACCOUNT

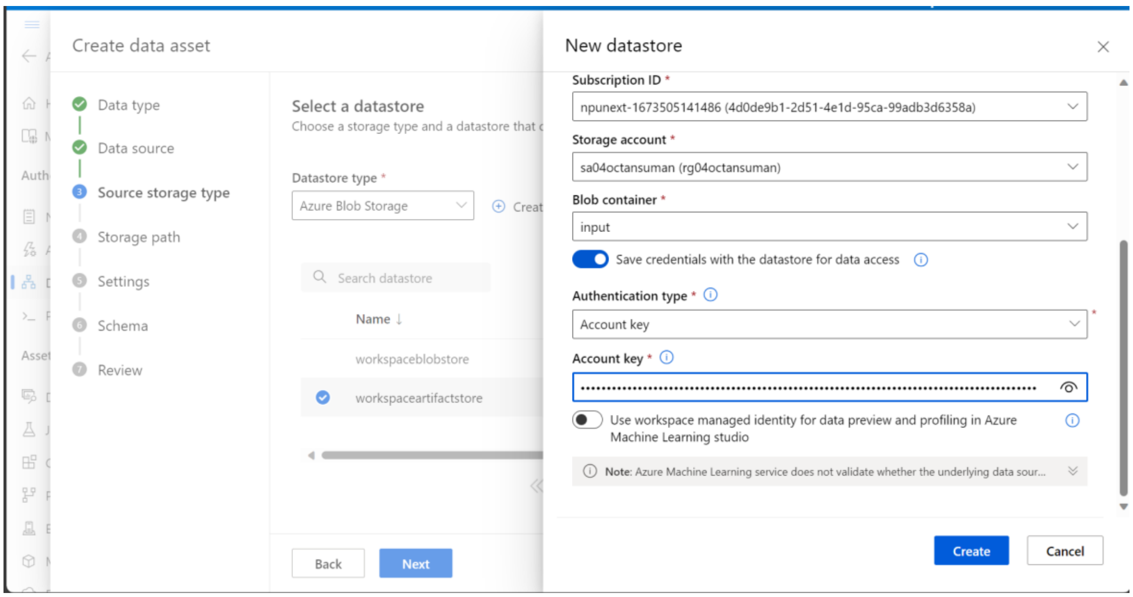


CONNECTING ML STUDIO WITH STORAGE ACCOUNT

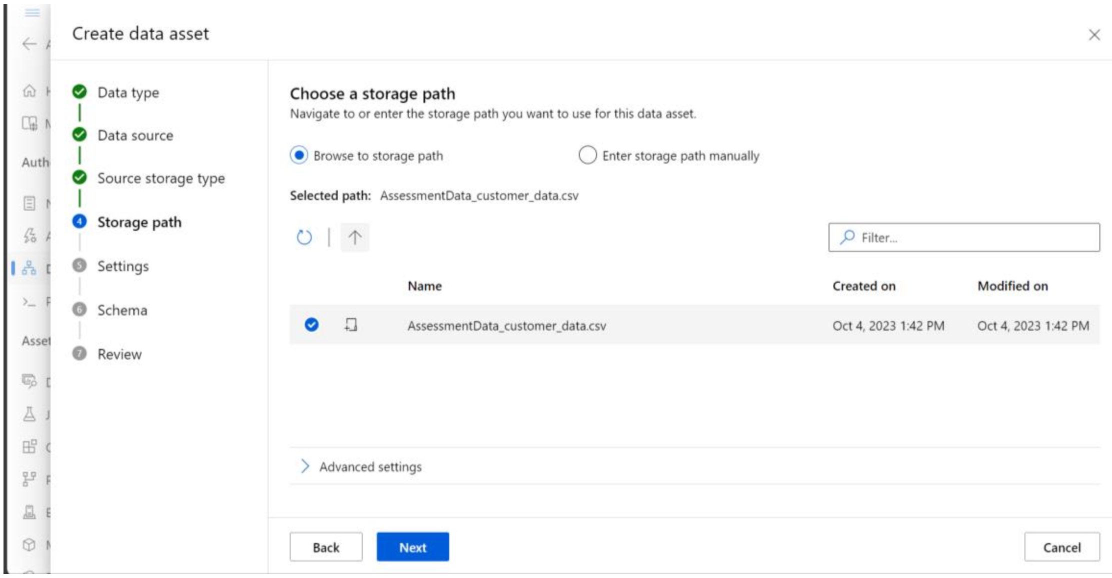




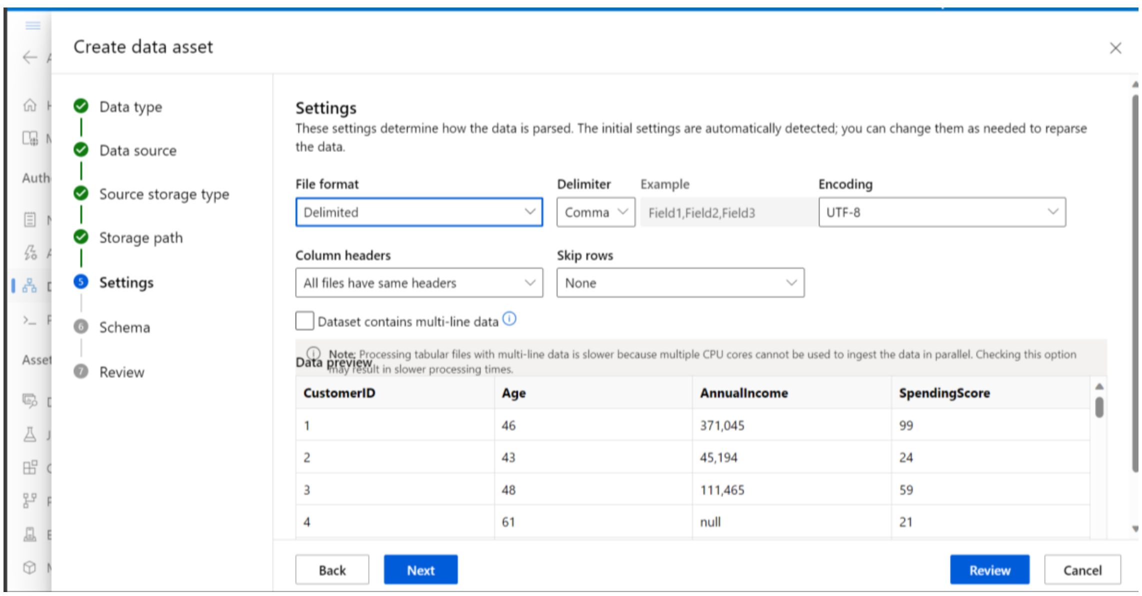
- Creating new datastore
- Providing the access key of the container



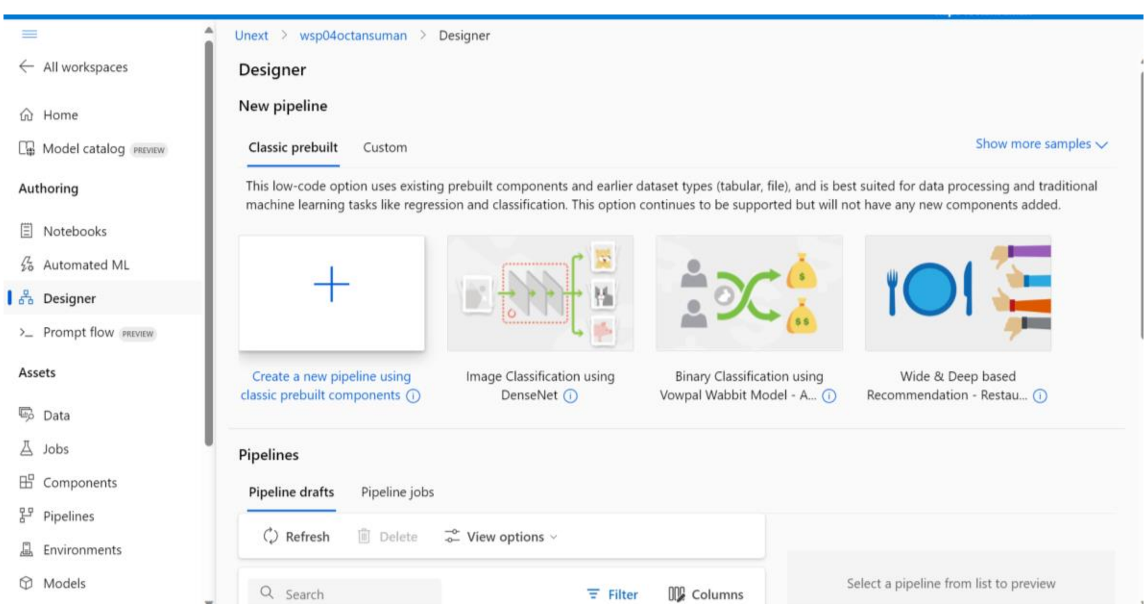
- Choosing a storage path



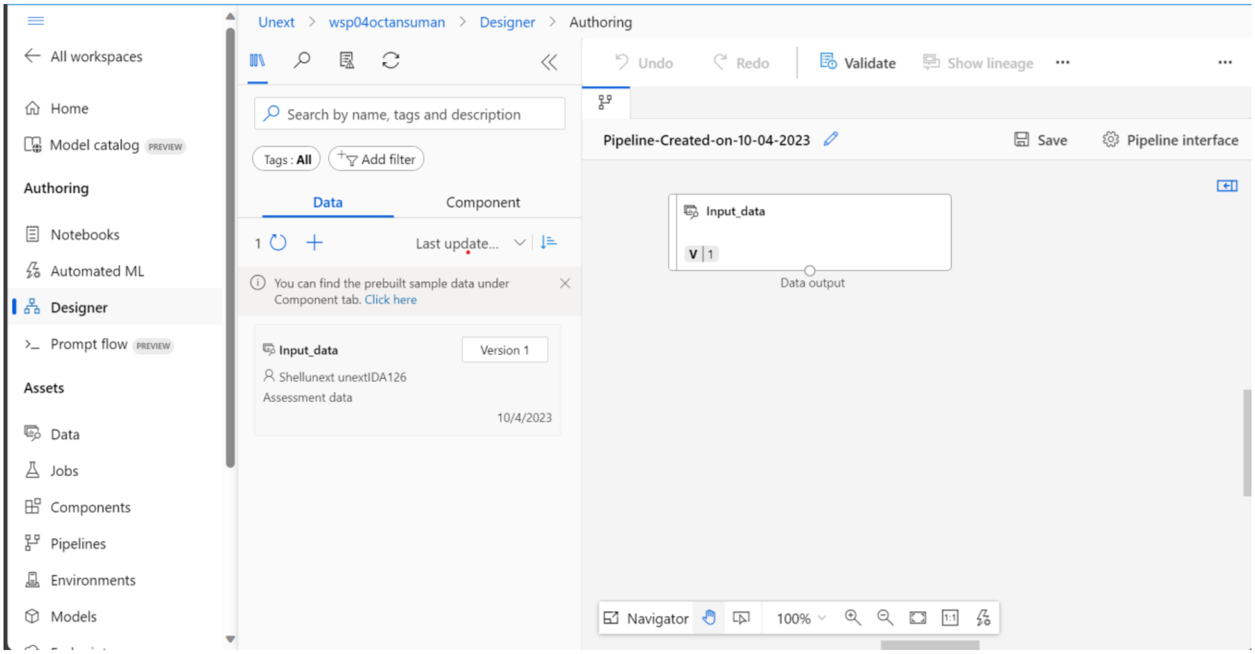
- Providing the correct settings



CREATING A NEW ML PIPELINE DESIGNER

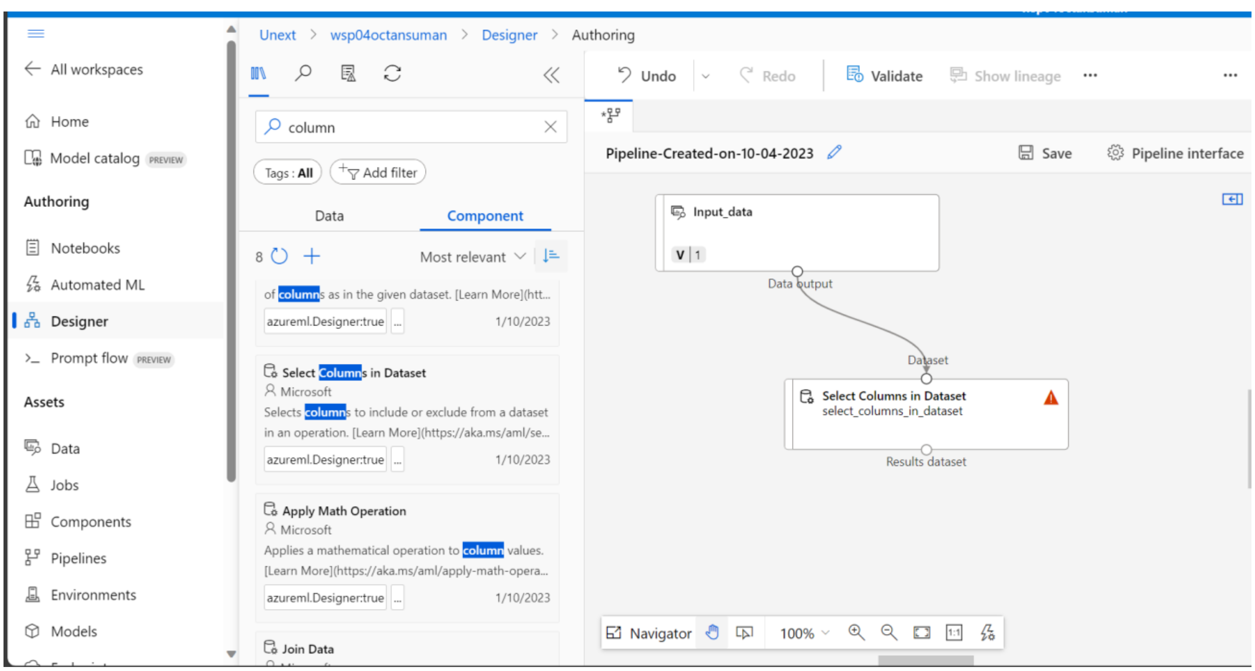


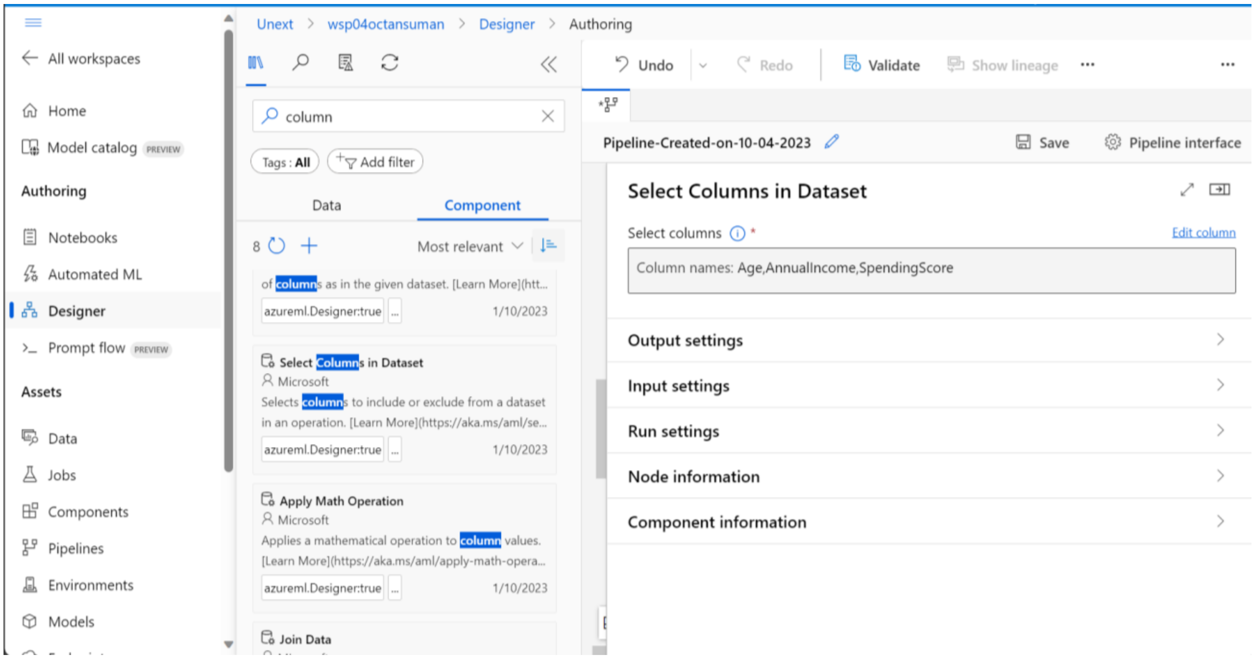
ADDING INPUT DATA INTO DESIGNER

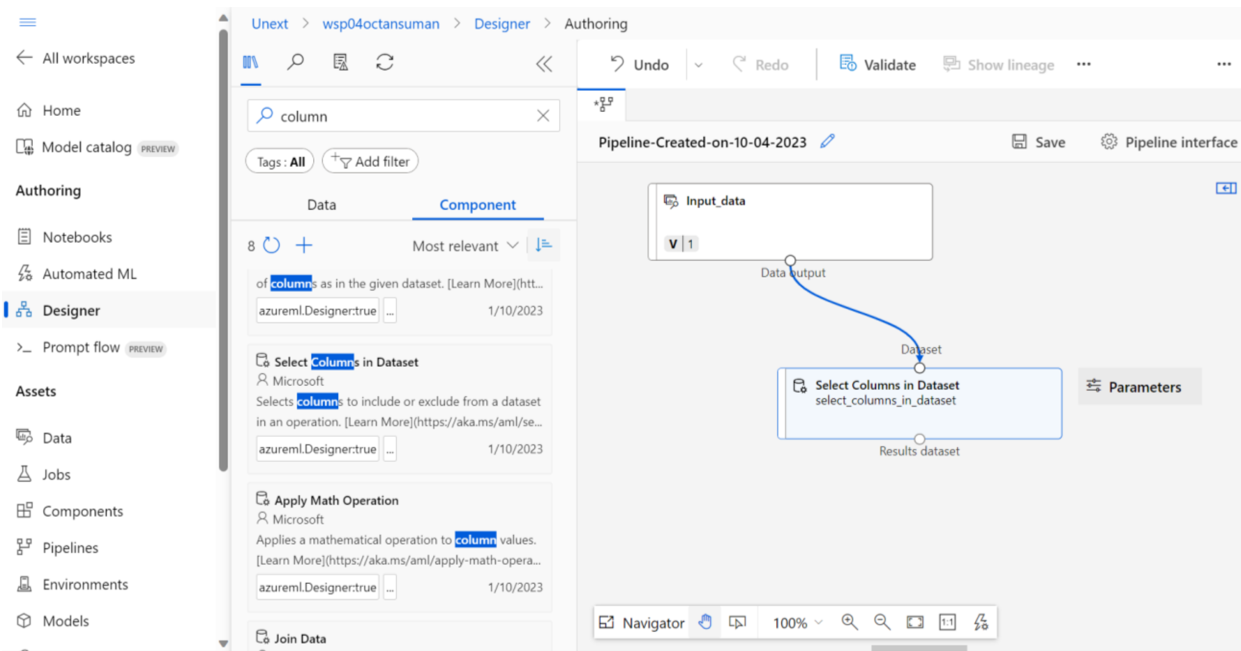


SELECTING REQUIRED COLUMNS

- Customer ID is not a required column in our dataset. It will not be used for prediction in ML model.
- Using the select column component toi select specific column

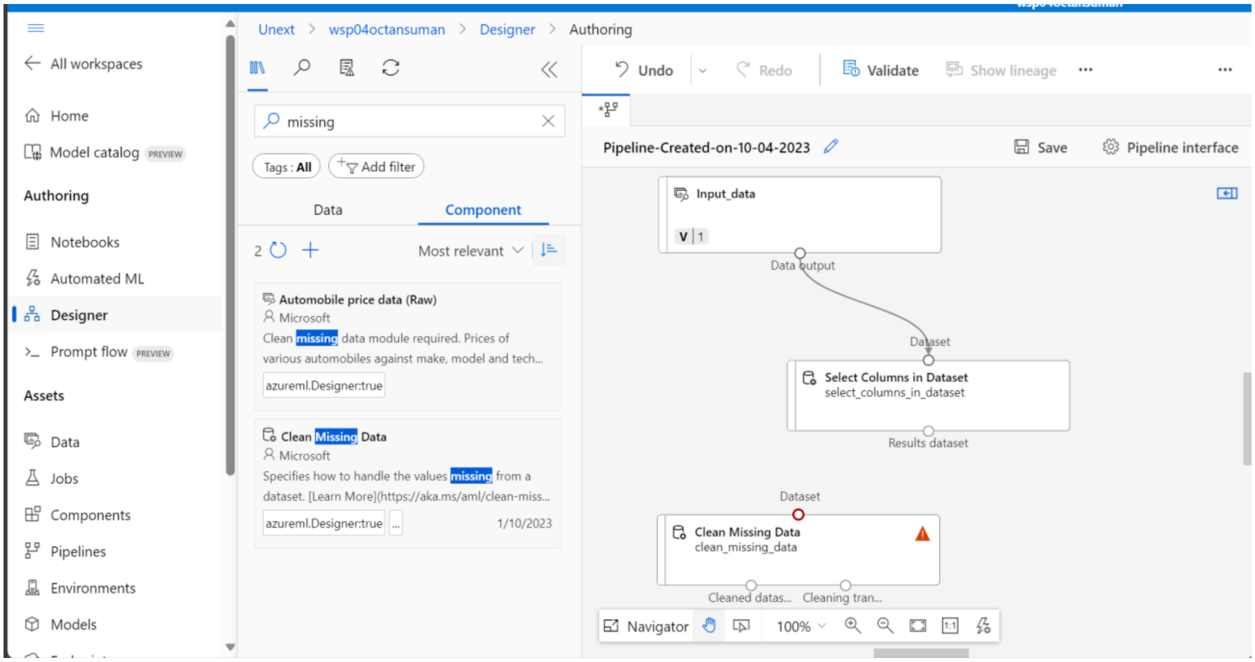




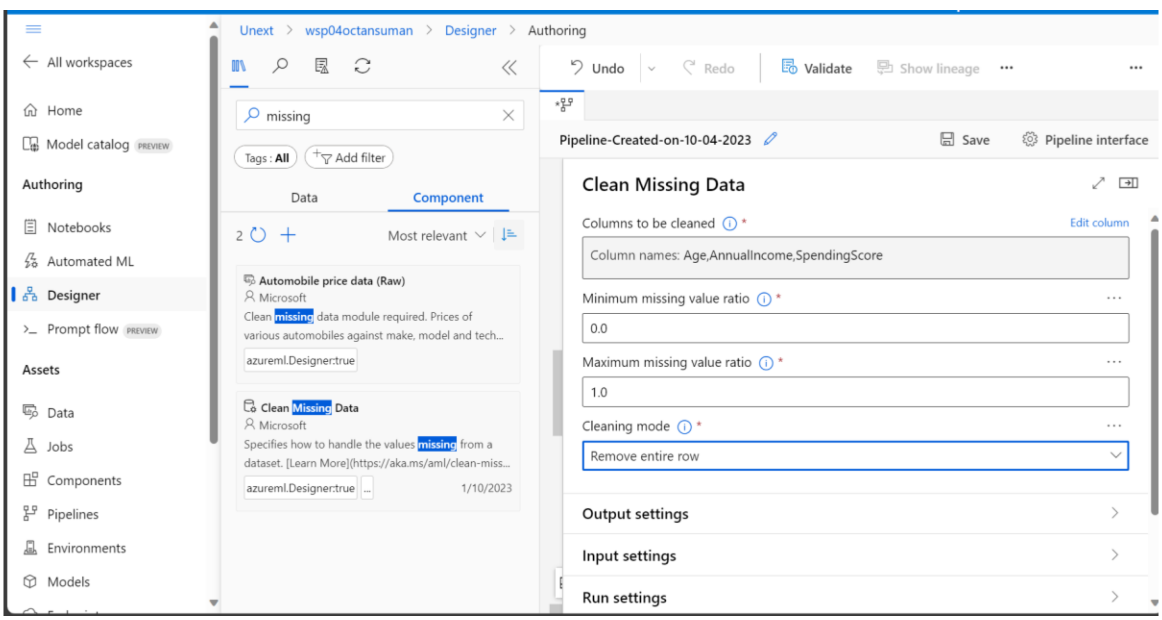


CLEANING MISSING DATA

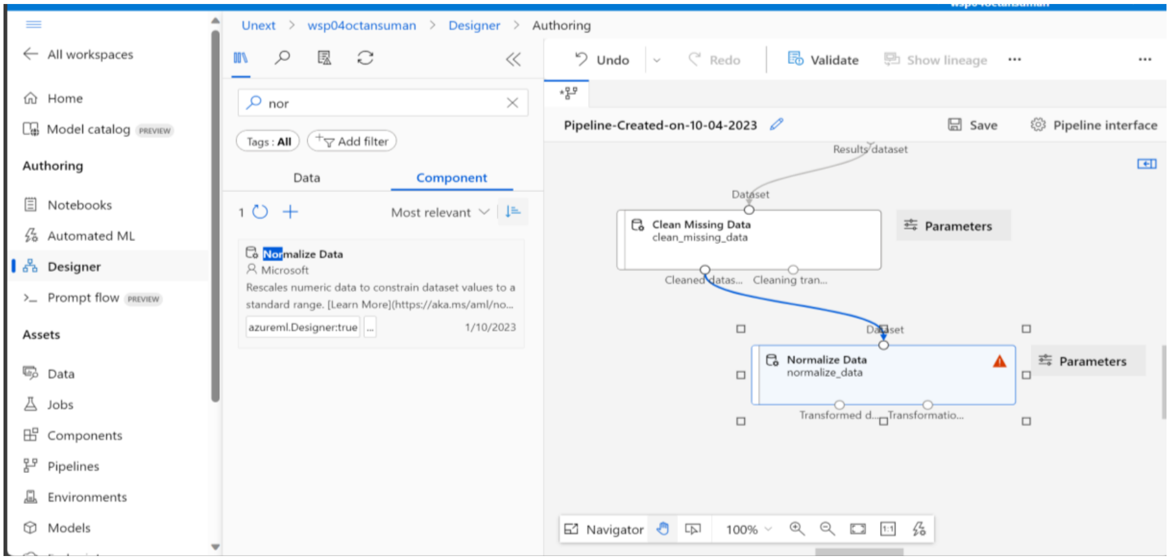
- Adding a clean missing data component



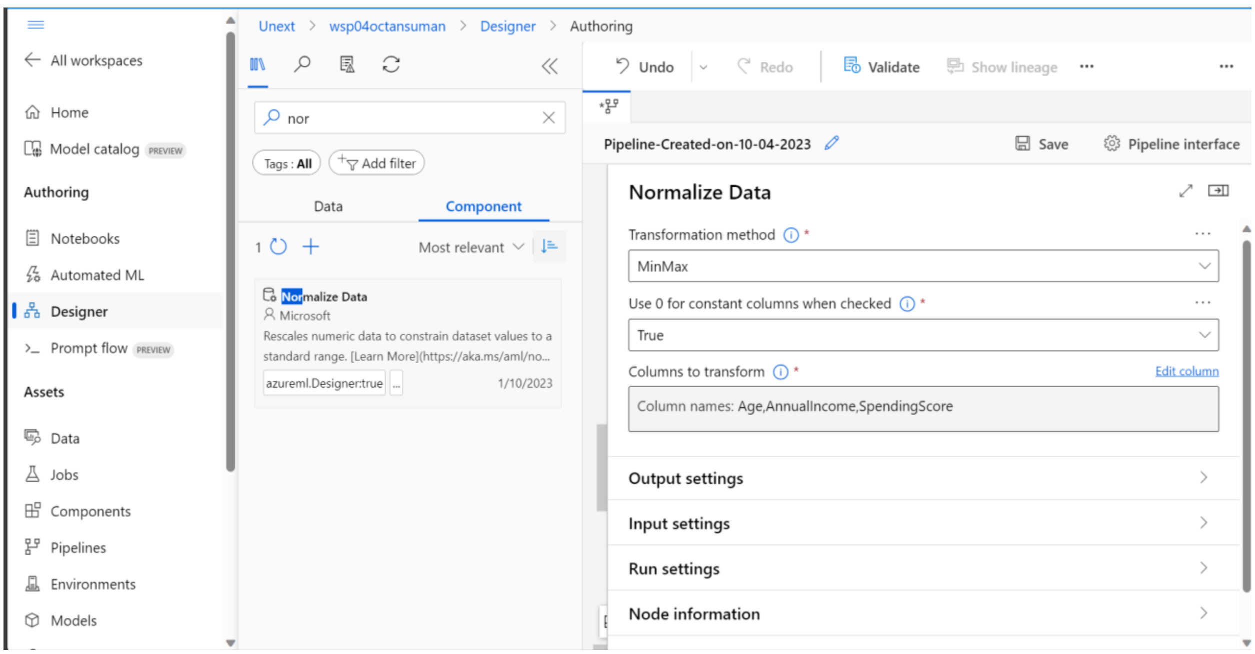
- Removing entire row if data is missing.

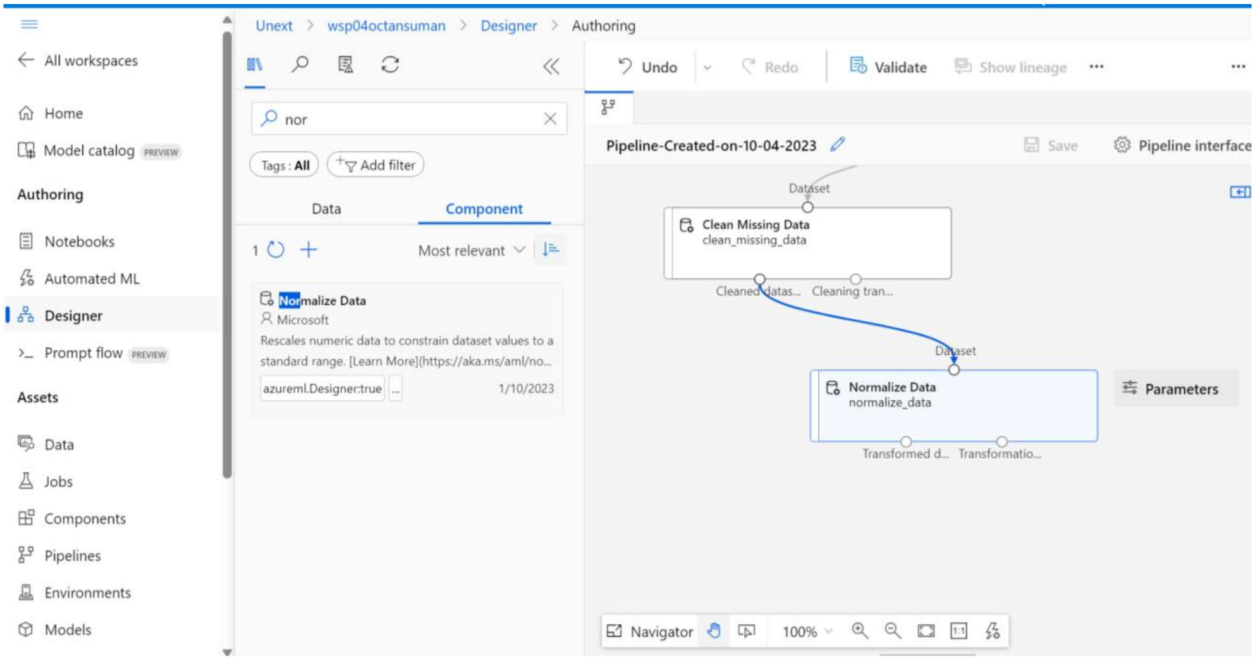


NORMALISING



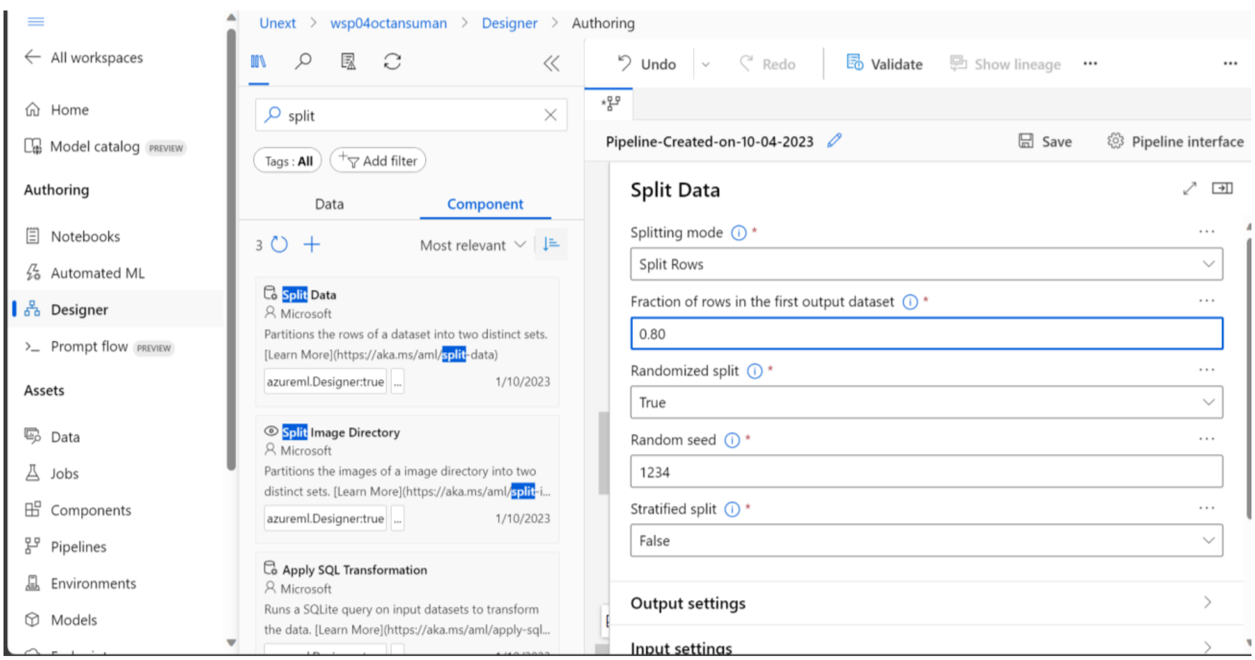
- MinMax Normalisation

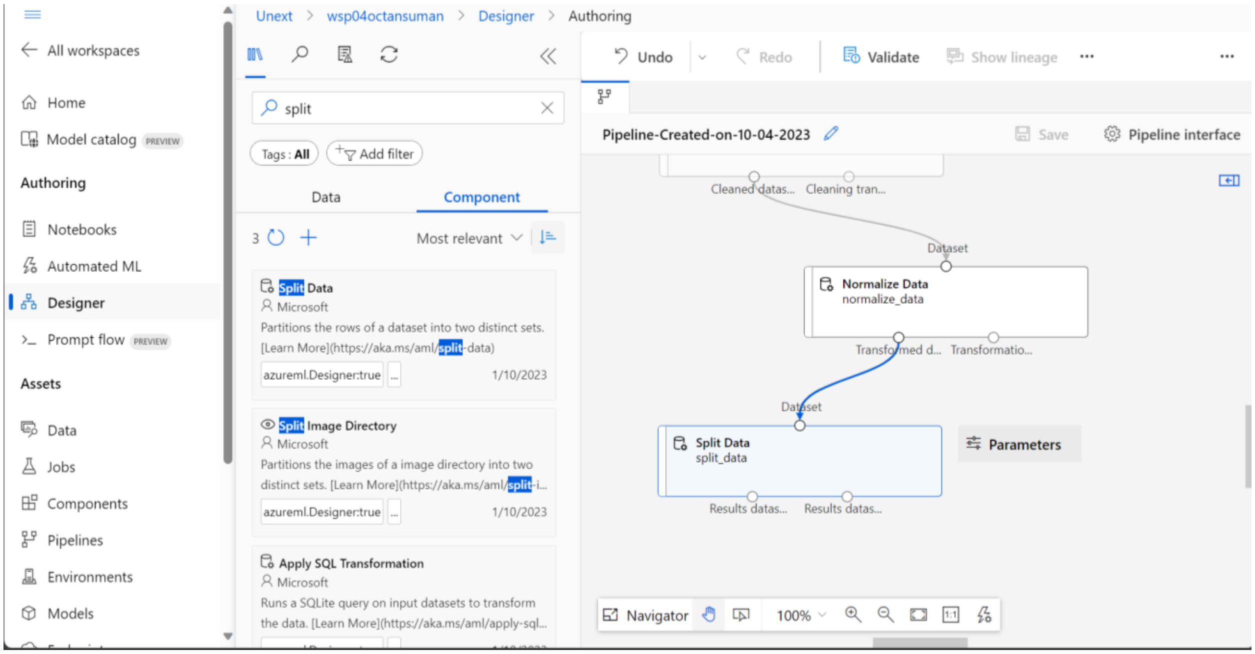




SPLITTING DATA

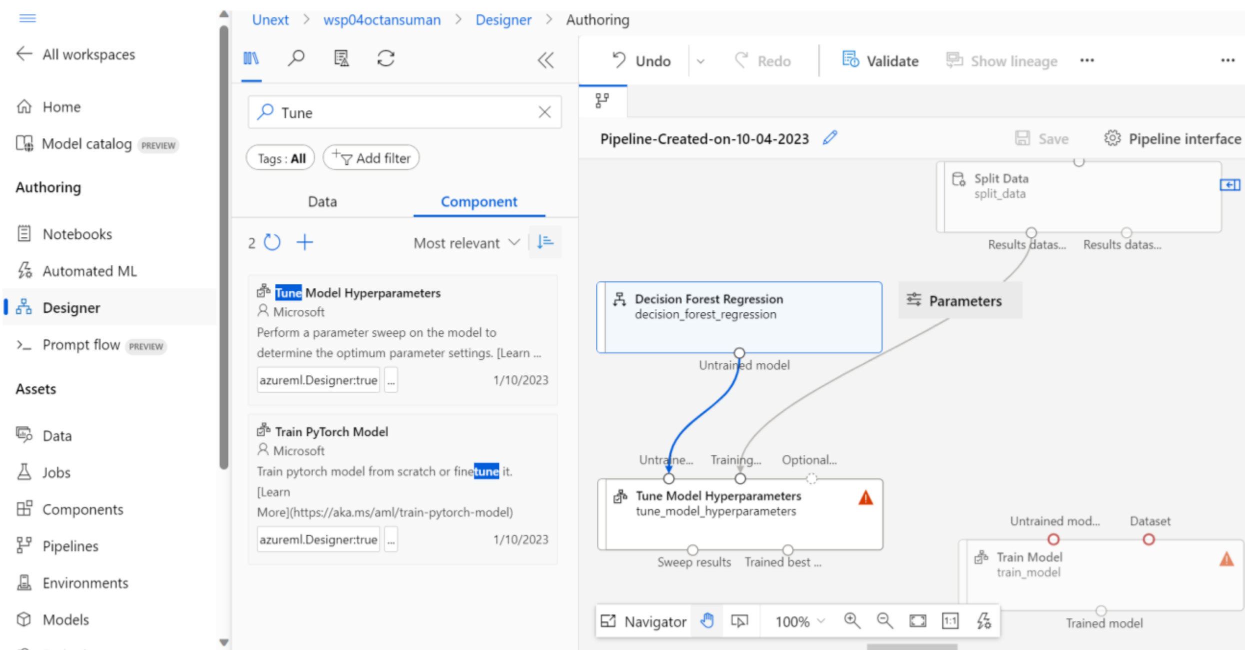
80:20 Split





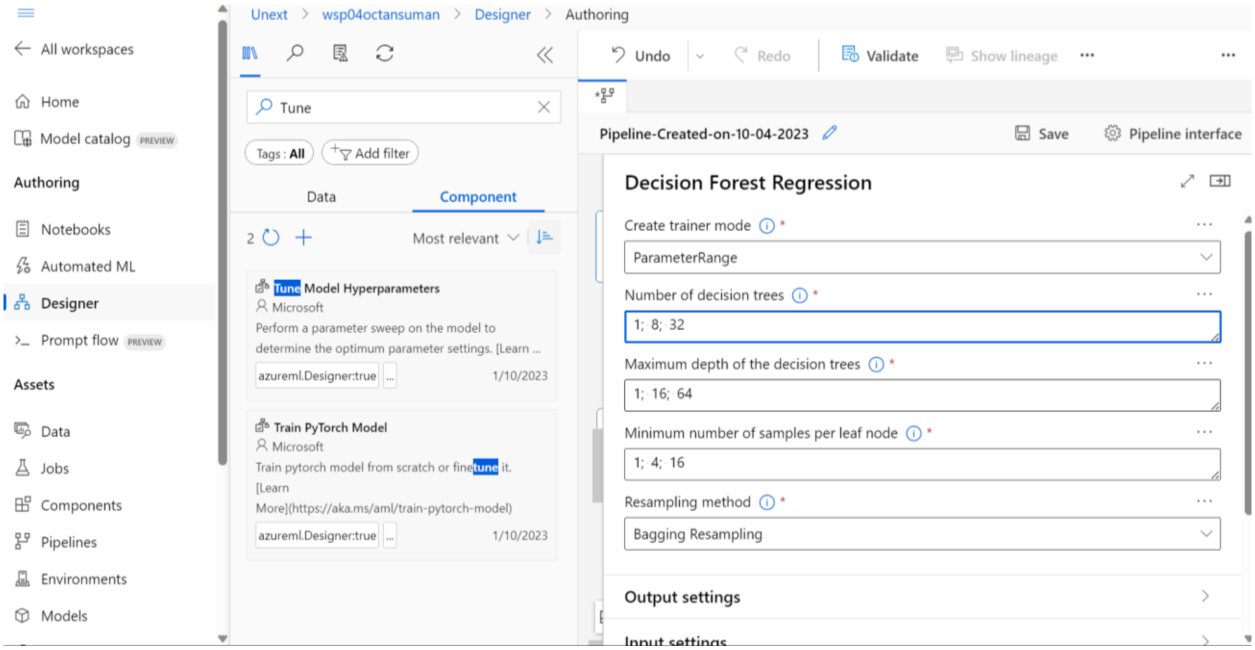
MODEL

Decision Tree Regressor

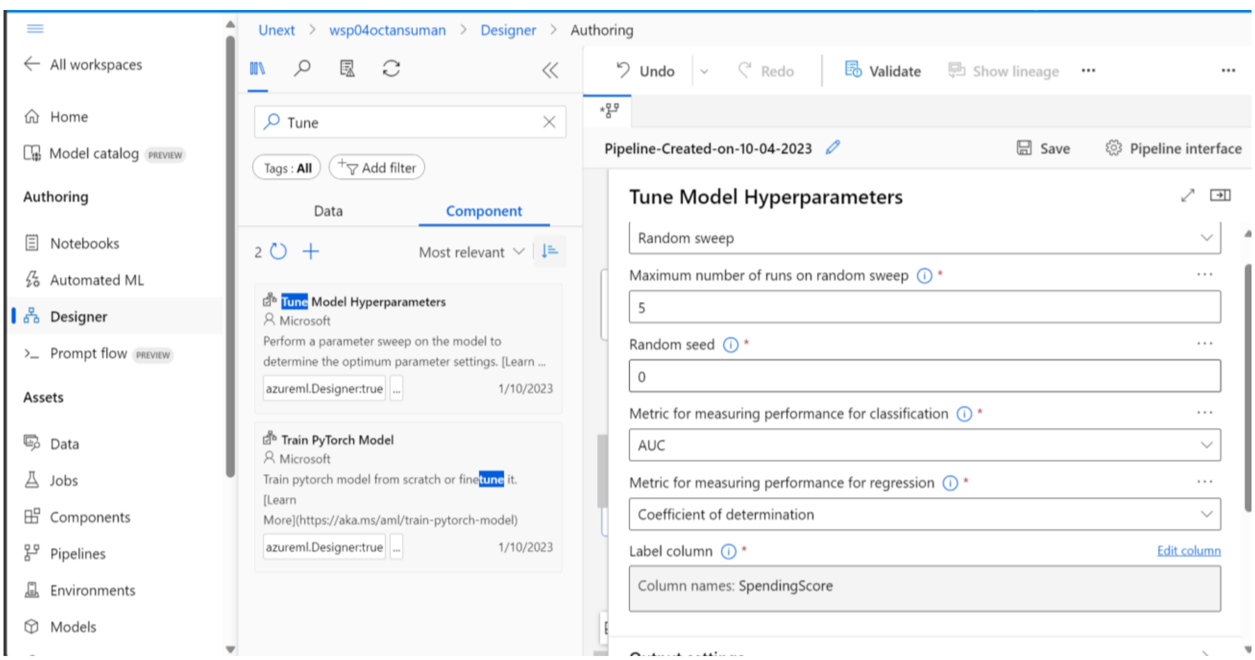


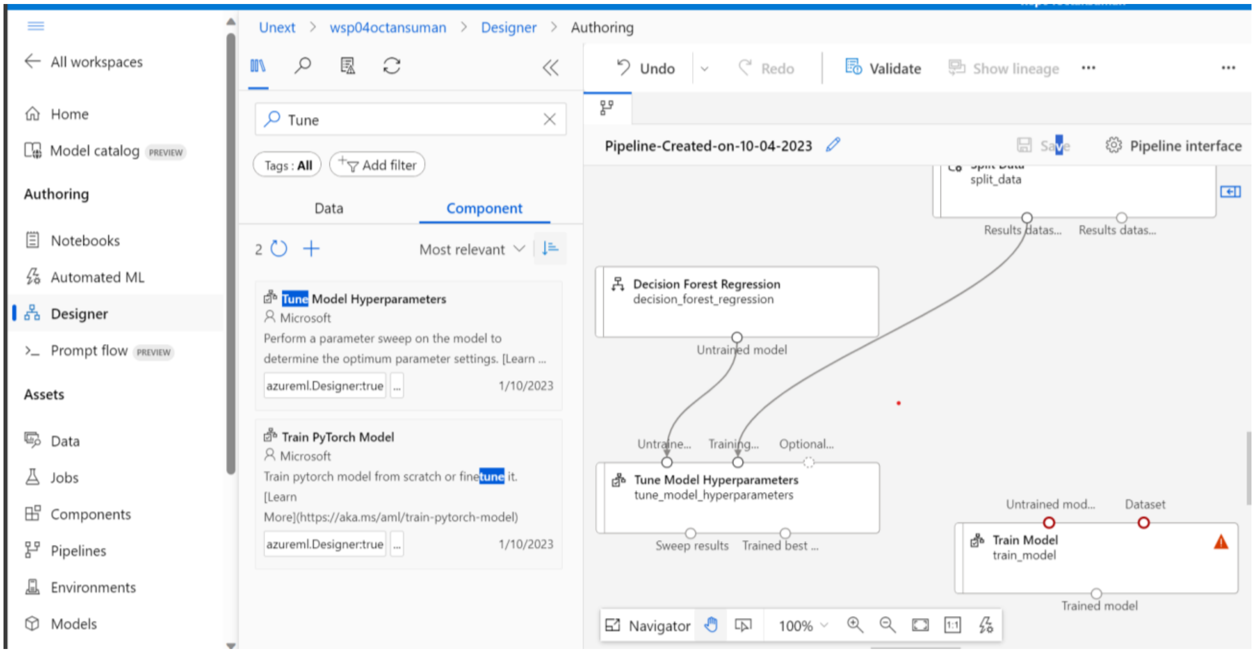
HYPER PARAMETER TUNNING

- Selecting range of hyperparameters

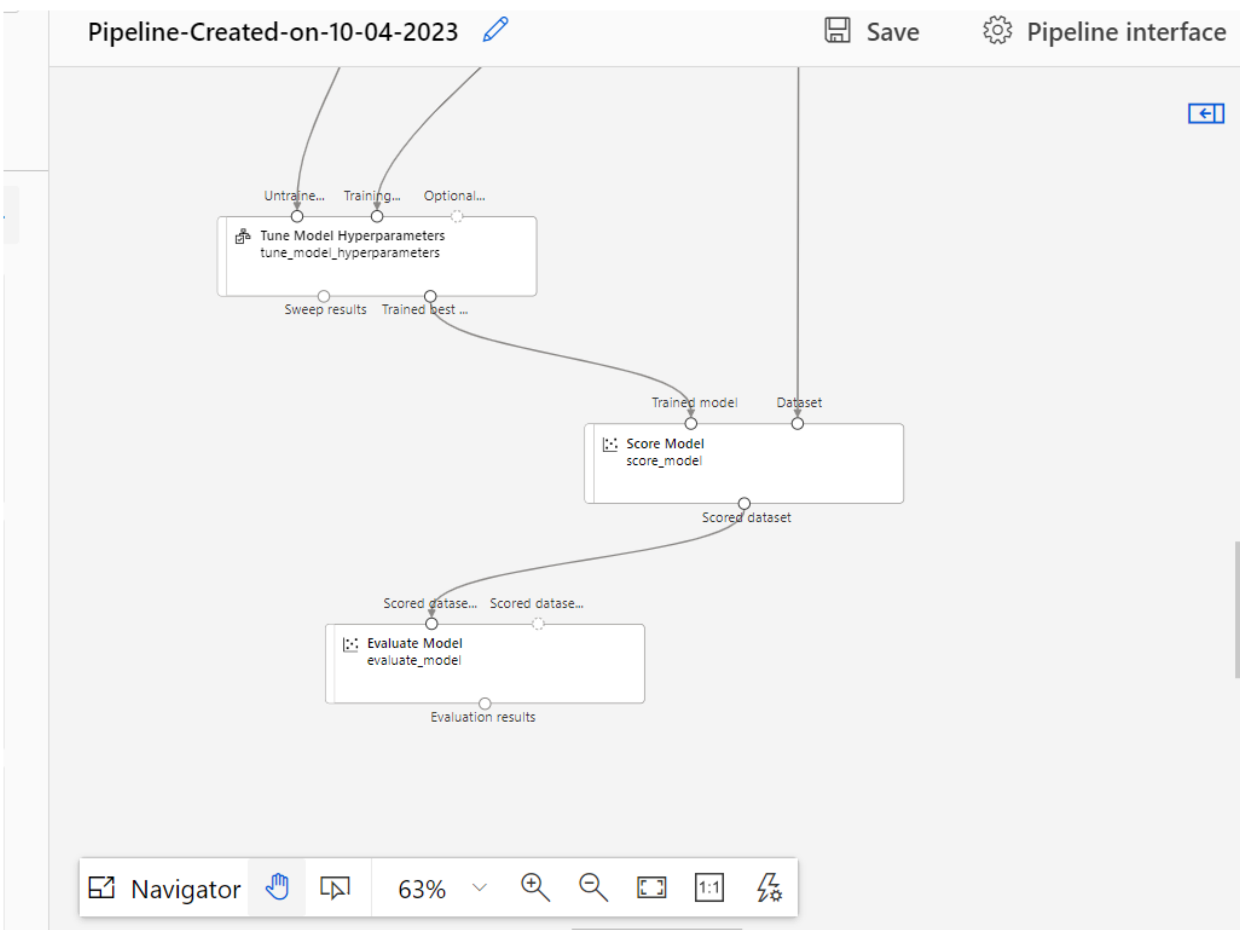


Providing hyper prameters

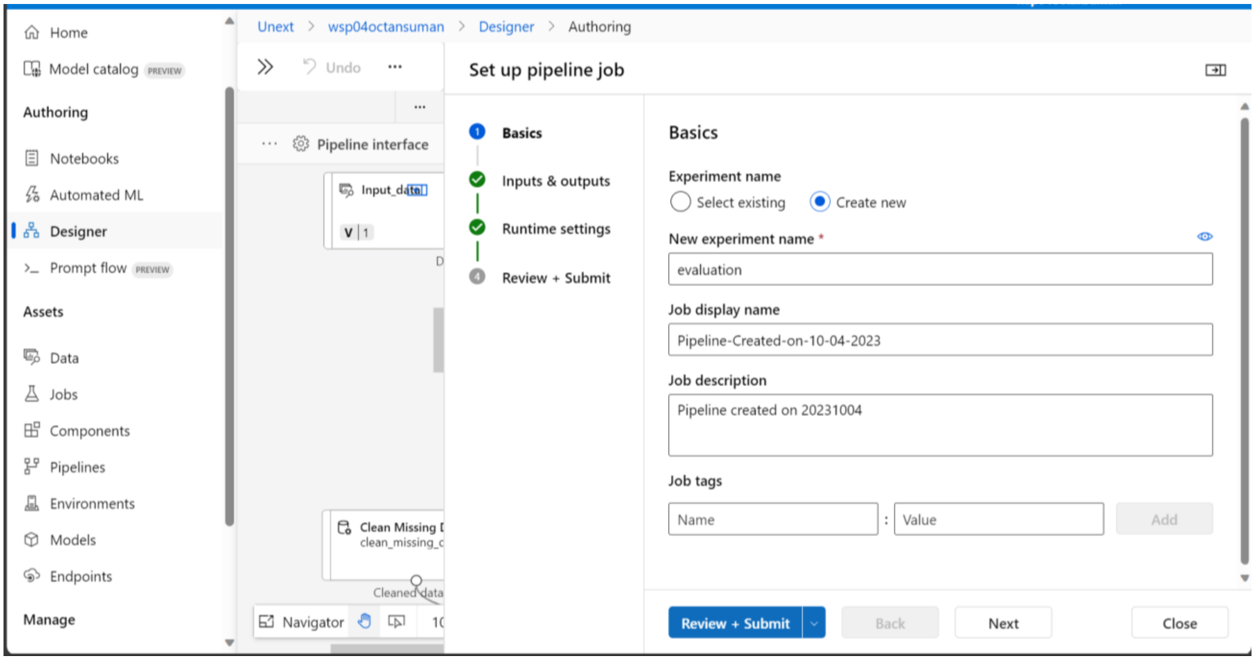


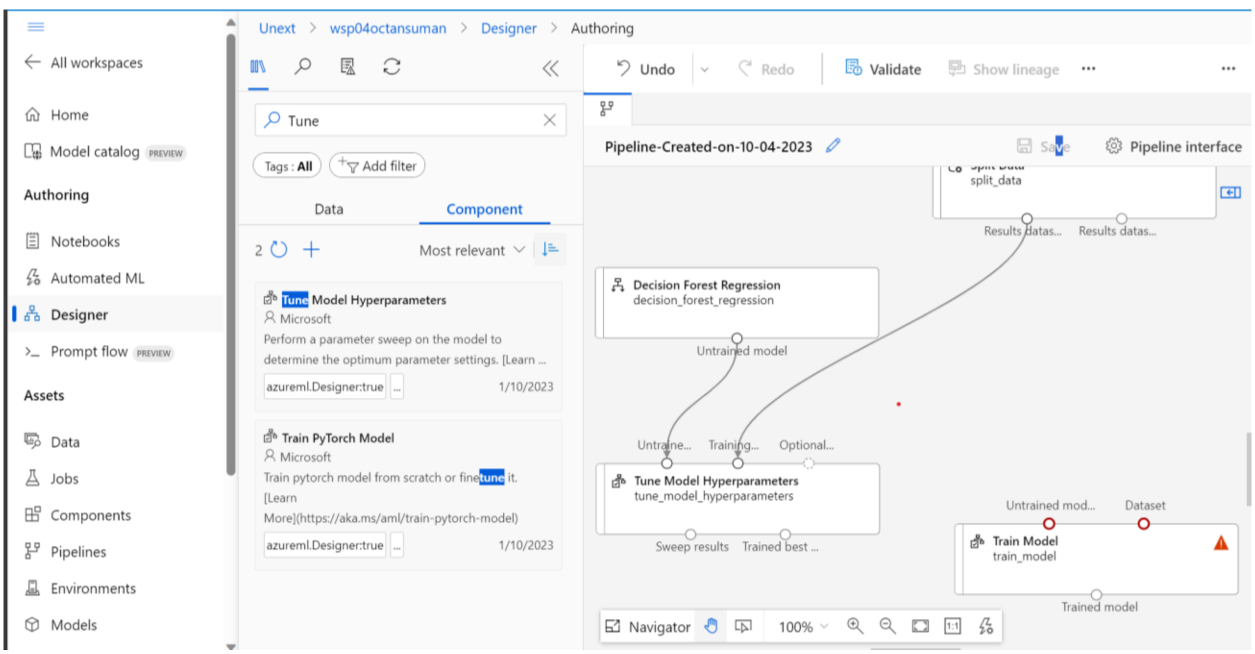


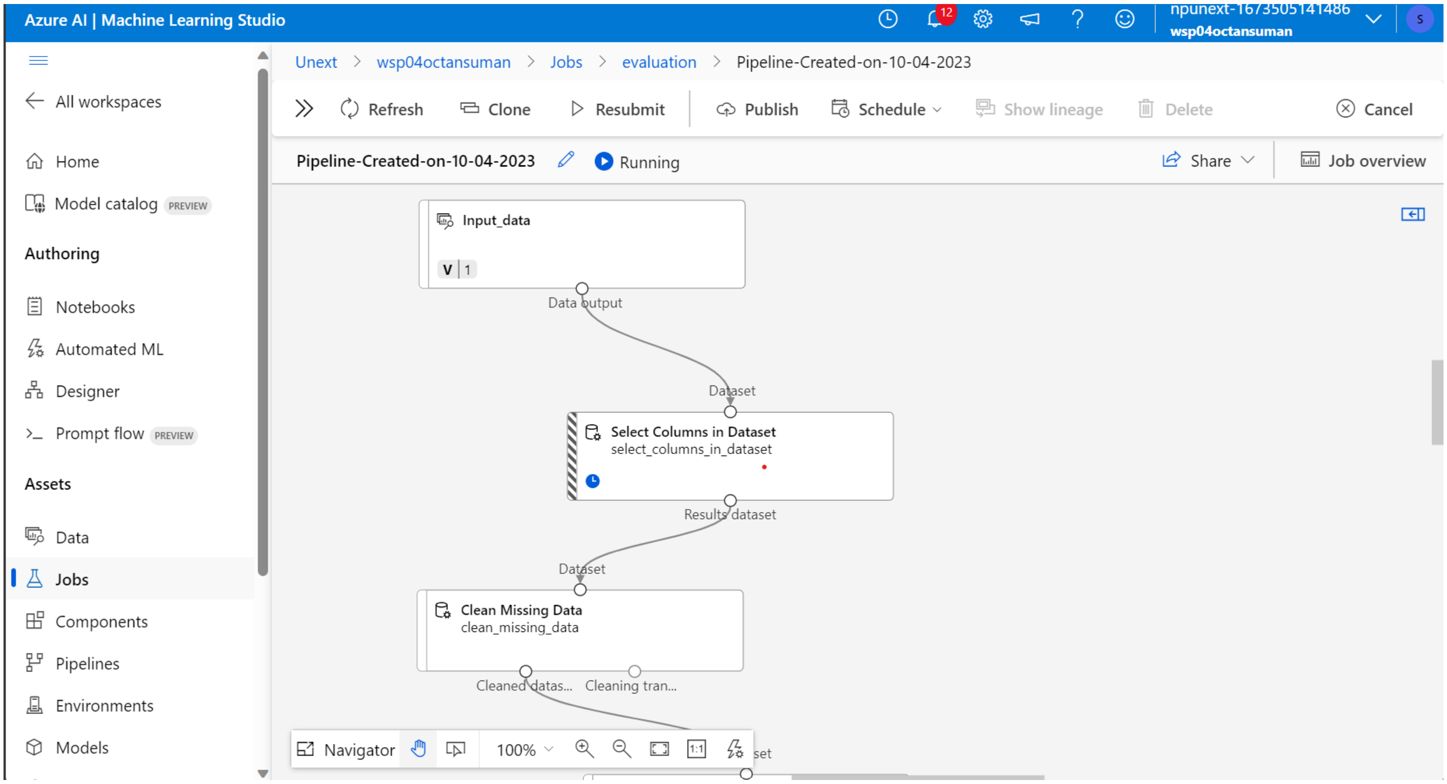
SCORING AND EVALUATING MODEL



RUNNING PIPELINE


Unext > wsp04octansuman > Designer > Authoring
Home
Model catalog PREVIEW
Authoring
Notebooks
Automated ML
Designer
Prompt flow PREVIEW
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Undo
Pipeline interface
Input_data
Clean Missing D
clean_missing_c
Cleaned data
Navigator
Set up pipeline job
Basics
Inputs & outputs
Runtime settings
Review + Submit
Basics
Experiment name
Select existing
Create new
New experiment name *
evaluation
Job display name
Pipeline-Created-on-10-04-2023
Job description
Pipeline created on 20231004
Job tags
Name
Value
Add
Review + Submit
Back
Next
Close


Unext > wsp04octansuman > Designer > Authoring
All workspaces
Home
Model catalog PREVIEW
Authoring
Notebooks
Automated ML
Designer
Prompt flow PREVIEW
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Tune
Tags : All
Add filter
Data
Component
Most relevant
Tune Model Hyperparameters
Microsoft
Perform a parameter sweep on the model to determine the optimum parameter settings. [Learn ...
azureml.Designer:true
1/10/2023
Train PyTorch Model
Microsoft
Train pytorch model from scratch or finetune it. [Learn More](https://aka.ms/aml/train-pytorch-model)
azureml.Designer:true
1/10/2023
Undo
Redo
Validate
Show lineage
Pipeline-Created-on-10-04-2023
Save
Pipeline interface
split_data
Results datas...
Results datas...
Decision Forest Regression
decision_forest_regression
Untrained model
Untraine...
Training...
Optional...
Tune Model Hyperparameters
tune_model_hyperparameters
Sweep results
Trained best ...
Untrained mod...
Dataset
Train Model
train_model
Trained model
Navigator
100%



Q. Preparing a dataset for training a machine learning model using Azure Machine Learning involves several key steps.

1. Data Collection and Ingestion:

   Gather the data from various sources, such as databases, files, or external APIs.

   Ingest the data into Azure Machine Learning using tools like Azure Data Factory, Azure Data Lake Storage, or Azure Blob Storage.

2. Data Cleaning and Preprocessing:

   Handle missing data by either imputing values or removing rows/columns with missing values.

   Perform data transformations such as scaling, normalization, or encoding categorical variables.

   Detect and handle outliers or anomalies in the data.

3. Data Splitting:

   Split the dataset into training, validation, and test sets.

The training set is used to train the model, the validation set is used for hyperparameter tuning, and the test set is used for final model evaluation.

4. Feature Engineering:

   Create or modify new features to enhance the model's predictive power.

   Feature selection may also be performed to choose the most relevant features.

5. Data Validation:

   Check the quality of your dataset to ensure it meets the requirements for model training.

   Validate that the data is correctly formatted, and there are no unexpected issues.

6. Data Pipeline Creation:

   Build a data pipeline that automates the data preprocessing steps and makes it easy to repeat the process consistently.

7. Data Profiling and Visualization:

   Visualize and analyze the dataset to gain insights into its characteristics.

   Understand the distribution of features and relationships within the data.

8. Data Serialization and Storage:

   Serialize and store the preprocessed data in a format suitable for training (e.g., Parquet, CSV) and save it to Azure storage services for easy access during model training.

9. Data Registration:

   Register the prepared dataset in Azure Machine Learning's dataset registry for easy access and sharing with others in your team.

Q. **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?**

1. Assessing Generalization:

The primary goal of a machine learning model is to make accurate predictions on new, unseen data. Splitting the dataset helps you assess how well your model generalizes to unseen data.

2. Model Evaluation:

The testing set serves as a holdout dataset that is not used during training. After the model is trained on the training set, you can evaluate its performance on the testing set to estimate its predictive accuracy.

3. Preventing Overfitting:

Overfitting occurs when a model learns to memorize the training data but fails to generalize to new data. The testing set helps you detect overfitting. If a model performs significantly worse on the testing set than the training set, it may be overfitting.

4. Hyperparameter Tuning:

We can use the testing set to perform hyperparameter tuning. By trying different combinations of hyperparameters and evaluating their performance on the testing set, you can optimize your model for better results.

5. Quantitative Performance Metrics:

Splitting the data allows you to compute various quantitative performance metrics, such as accuracy, precision, recall, F1 score, and others, to measure how well the model performs on different aspects of prediction.

6. Model Selection:

we can compare the performance of multiple models (e.g., different algorithms or architectures) using the same testing set. This helps you choose the best model for your specific problem.

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

I used **Decision Tree Regression** in the scenario of predicting customer purchasing behavior.

- Decision trees are capable of capturing nonlinear relationships between independent variables (age and annual income) and the dependent variable (purchase behavior). This flexibility can be advantageous when the true relationship is not strictly linear.

- decision trees provide a relatively interpretable model. we can visualize the tree structure to understand the decision-making process, making it useful for explaining the factors influencing purchasing behavior to non-technical stakeholders.

- This model can naturally handle interactions between features. For instance, the algorithm can identify cases where a combination of age and income has a specific impact on purchase behavior that may not be evident in a linear model.

- Automatic Feature Selection: Decision trees can perform implicit feature selection by deciding which features are most important for the prediction task. This can be helpful in scenarios where we suspect that only certain aspects of age and income are relevant predictors.

- Robustness to Outliers: Decision trees are robust to outliers and anomalies in the data.

- Decision trees require minimal data preprocessing and are relatively easy to implement. They can handle missing values without extensive imputation techniques.

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

Hyperparameter tuning, also known as hyperparameter optimization, is the process of finding the optimal set of hyperparameters for a machine learning model. Hyperparameters are configuration settings for a model that are not learned from the data but are set before the training process begins.

Hyperparameter tuning is crucial in machine learning for several reasons:

1. **Model Performance**: The choice of hyperparameters can significantly impact the performance of a machine learning model. The right hyperparameters can lead to improved accuracy and generalization, while poor choices can result in underfitting or overfitting.
2. **Generalization:** Optimizing hyperparameters helps the model generalize well to unseen data, which is the ultimate goal of any machine learning model.
3. **Efficiency**: Hyperparameter tuning can help achieve better model performance without the need for a larger dataset or a more complex model, thus improving efficiency.
4. **Robustness**: Properly tuned hyperparameters can make the model more robust to variations in the data and changes in the problem domain.

One common technique for hyperparameter tuning in decision trees is Grid Search:

Grid Search:

Grid Search is a systematic technique that involves defining a grid of hyperparameter values to explore. It then trains and evaluates the model for each combination of hyperparameters within the defined grid. The benefits of using Grid Search include:

1. Comprehensive Search: Grid Search exhaustively explores the hyperparameter space by trying all possible combinations within the specified grid. This ensures that you don't miss potential optimal configurations.
2. Reproducibility: Grid Search provides a systematic and reproducible way to find the best hyperparameters, making it easy to document and replicate experiments.
3. Ease of Implementation: Grid Search is straightforward to implement and can be used with various machine learning libraries and frameworks.