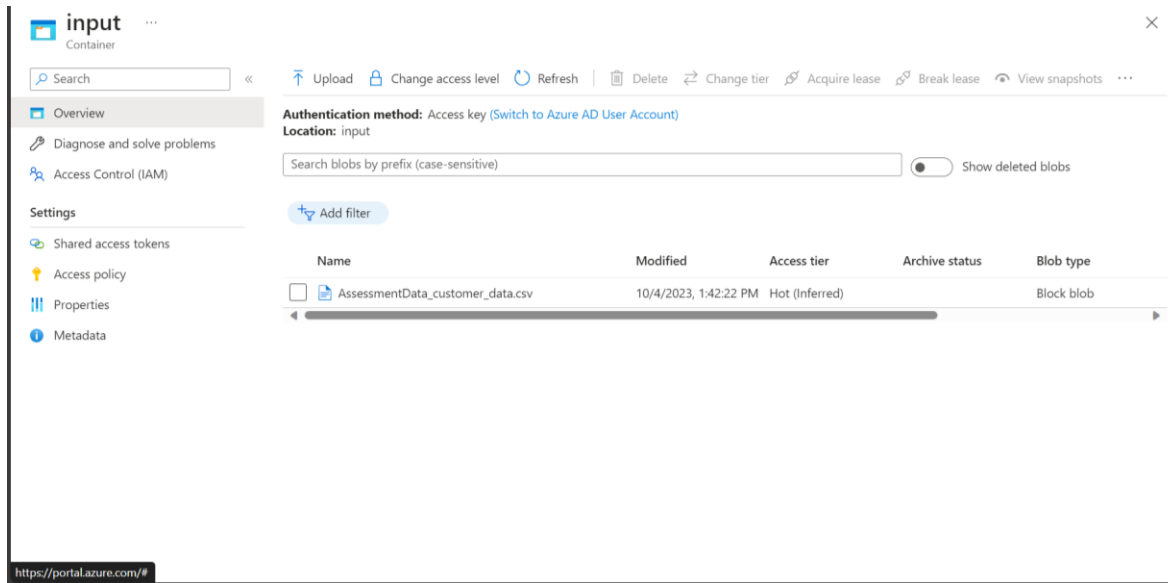


Purushottam Behera

Azure ML Studio

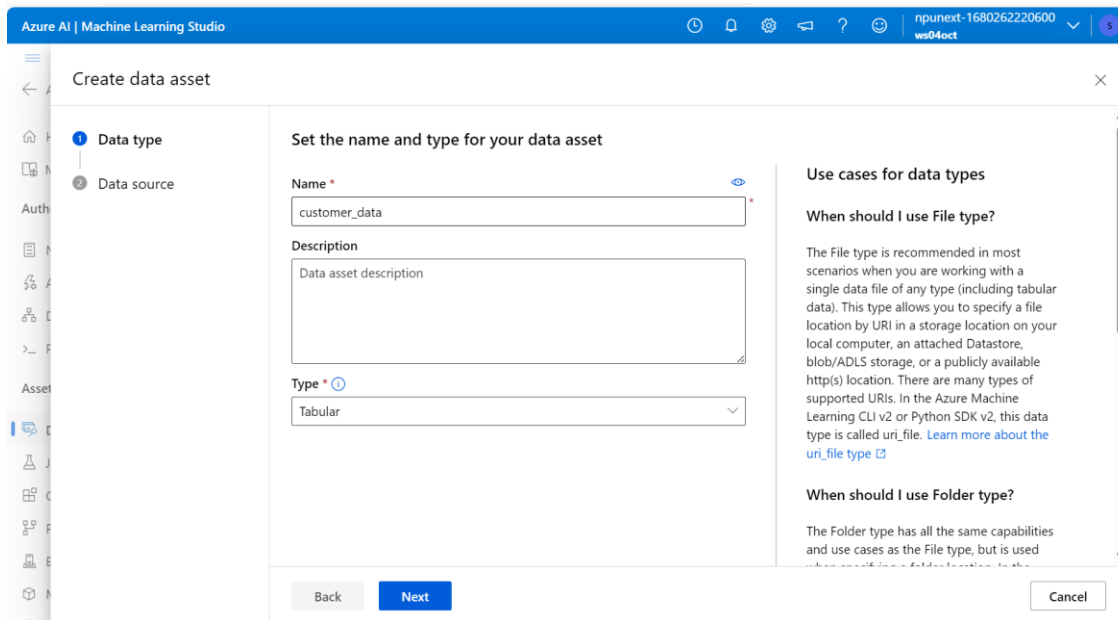
UPLOADING DATA INTO STORAGE ACCOUNT



The screenshot shows the 'input' container in Azure ML Studio. The left sidebar contains navigation links: Overview, Diagnose and solve problems, Access Control (IAM), Settings, Shared access tokens, Access policy, Properties, and Metadata. The main area displays the 'Authentication method' as 'Access key (Switch to Azure AD User Account)' and the 'Location' as 'input'. Below this is a search bar for blobs by prefix and a toggle for 'Show deleted blobs'. A table lists the blobs:

Name	Modified	Access tier	Archive status	Blob type
AssessmentData_customer_data.csv	10/4/2023, 1:42:22 PM	Hot (Inferred)		Block blob

CONNECTING ML STUDIO WITH STORAGE ACCOUNT



The screenshot shows the 'Create data asset' dialog in Azure ML Studio. The left sidebar has a 'Data type' section with 'Data source' selected. The main area is titled 'Set the name and type for your data asset'. It contains a 'Name' field with 'customer_data', a 'Description' field with 'Data asset description', and a 'Type' dropdown menu set to 'Tabular'. On the right, there is a 'Use cases for data types' section with two subsections: 'When should I use File type?' and 'When should I use Folder type?'. The 'File type' section explains that it is recommended for single data files and provides a link to 'Learn more about the uri_file type'. The 'Folder type' section explains that it has the same capabilities as the File type. At the bottom, there are 'Back', 'Next', and 'Cancel' buttons.

Create data asset

2 Data source

Choose a source for your data asset
Choose the data source you want to create your asset from. A data source can be from a local storage location on your computer, from an attached datastore, from Azure storage, or from a publicly available web location.

From Azure storage
Create a data asset from registered data storage services including Azure Blob Storage, Azure file share, and Azure Data Lake.

From local files
Create a data asset by uploading files from your local drive.

From SQL databases
Create a dataset from Azure SQL database and Azure PostgreSQL database.

From web files
Create a data asset from a single file located at a public web URL.

From Azure Open Datasets

Back **Next** **Cancel**

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

Create data asset

3 Source storage type

Select a datastore
Choose a storage type and a datastore that...

Datastore type *
Azure Blob Storage

Search datastore

Name ↓

workspaceblobstore

workspaceartifactstore

Back **Next**

New datastore

Subscription ID *
npunext-1673505141486 (4d0de9b1-2d51-4e1d-95ca-99adb3d6358a)

Storage account *
sa04octansuman (rg04octansuman)

Blob container *
input

☒ Save credentials with the datastore for data access

Authentication type *
Account key

Account key *
.....

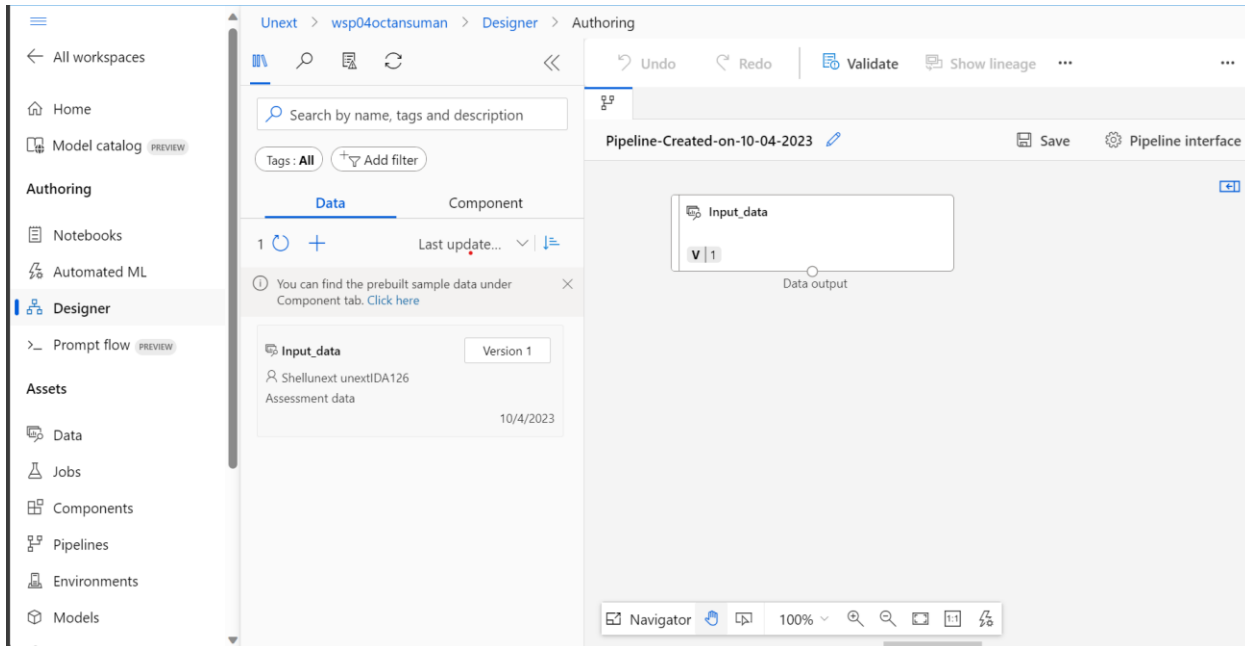
☐ Use workspace managed identity for data preview and profiling in Azure Machine Learning studio

Note: Azure Machine Learning service does not validate whether the underlying data source...

Create **Cancel**

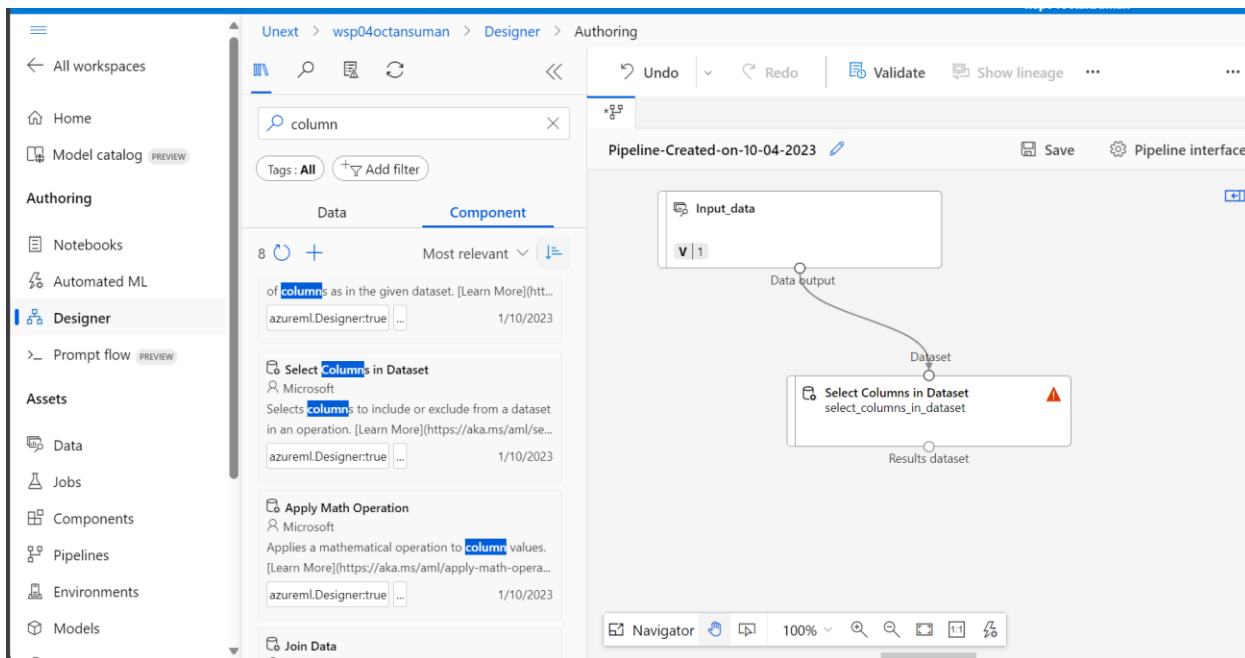
- Choosing a storage path

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 to select specific column



The image displays two screenshots of the Azure ML Designer interface, showing the configuration and pipeline integration of the 'Select Columns in Dataset' component.

Top Screenshot: Component Configuration

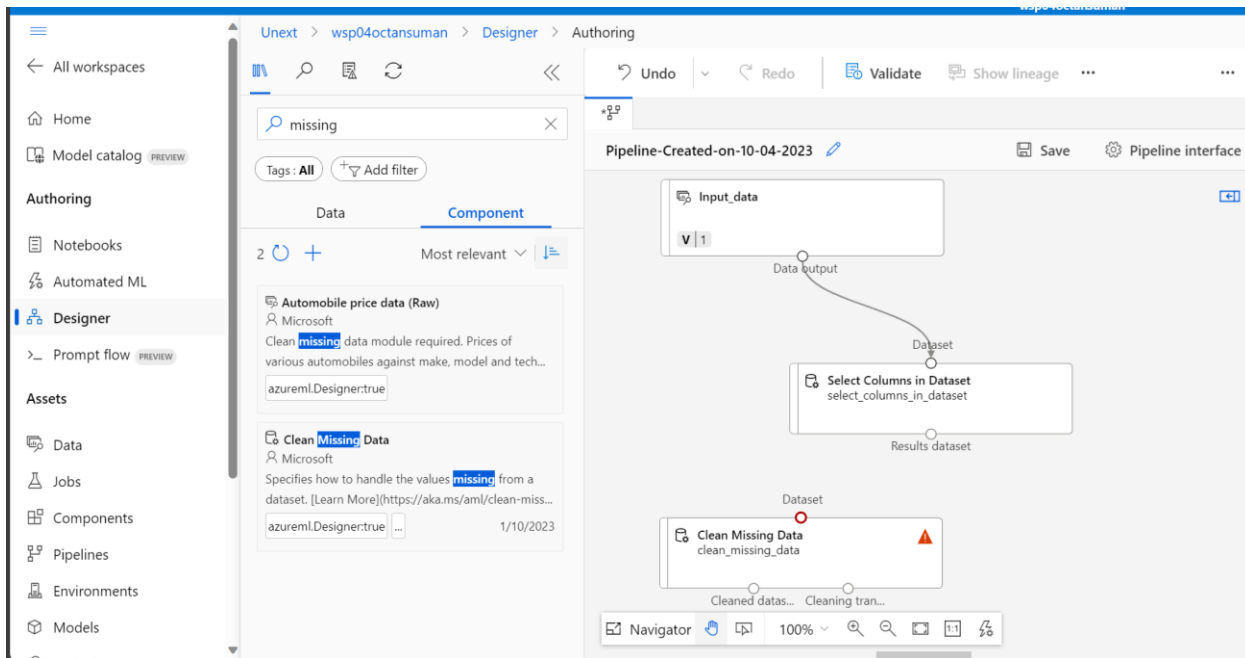
- Search:** The search bar contains 'column'.
- Tags:** All tags are visible, with 'Add filter' option.
- Data Component:** The 'Select Columns in Dataset' component is selected. It is described as 'Selects columns to include or exclude from a dataset in an operation.' The configuration shows 'Column names: Age, AnnualIncome, SpendingScore'.
- Output settings:** The output is set to 'Results dataset'.
- Input settings:** The input is set to 'Input_data'.
- Run settings:** The run settings are visible.
- Node information:** The node information is visible.
- Component information:** The component information is visible.

Bottom Screenshot: Pipeline Integration

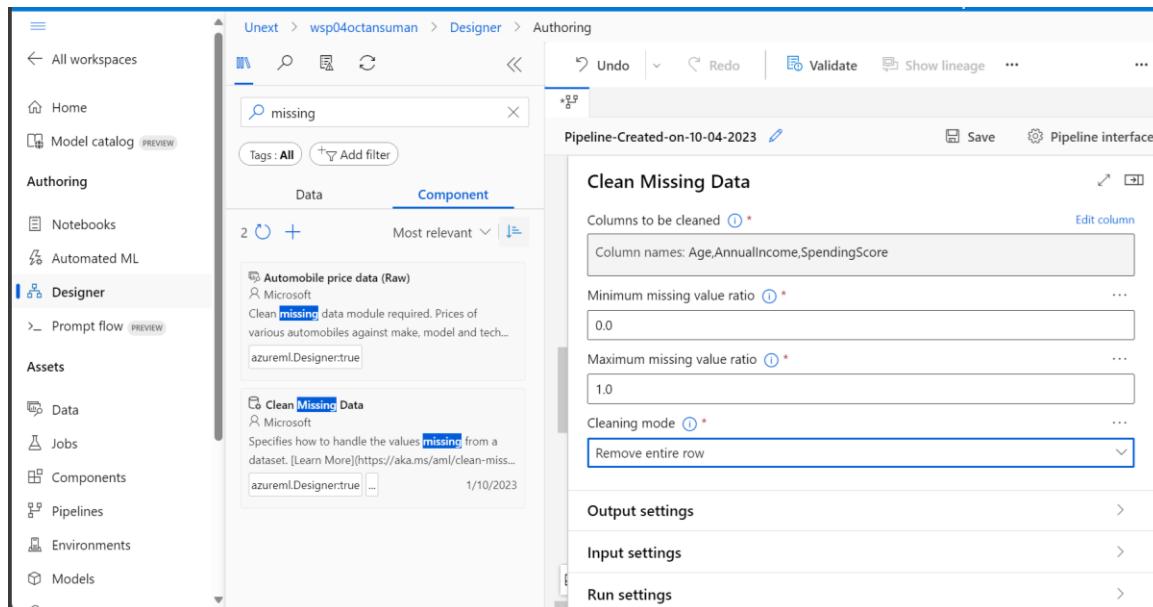
- Pipeline:** The pipeline is titled 'Pipeline-Created-on-10-04-2023'.
- Input:** The pipeline starts with 'Input_data'.
- Component:** The 'Select Columns in Dataset' component is connected to the pipeline. The output is labeled 'Results dataset'.
- Parameters:** The 'Parameters' tab is visible on the right.

CLEANING MISSING DATA

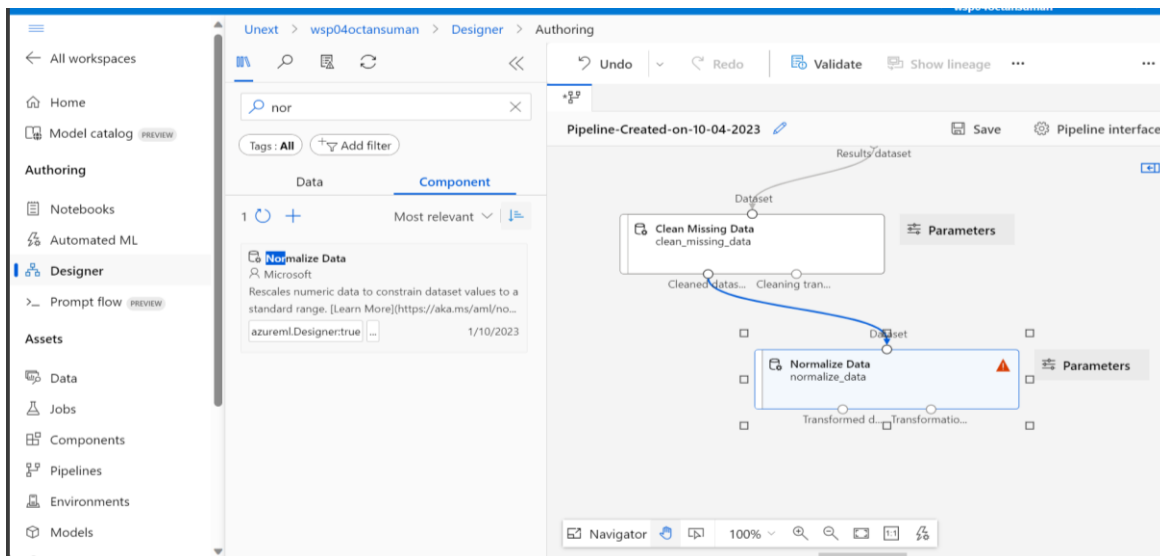
- Adding a clean missing data component



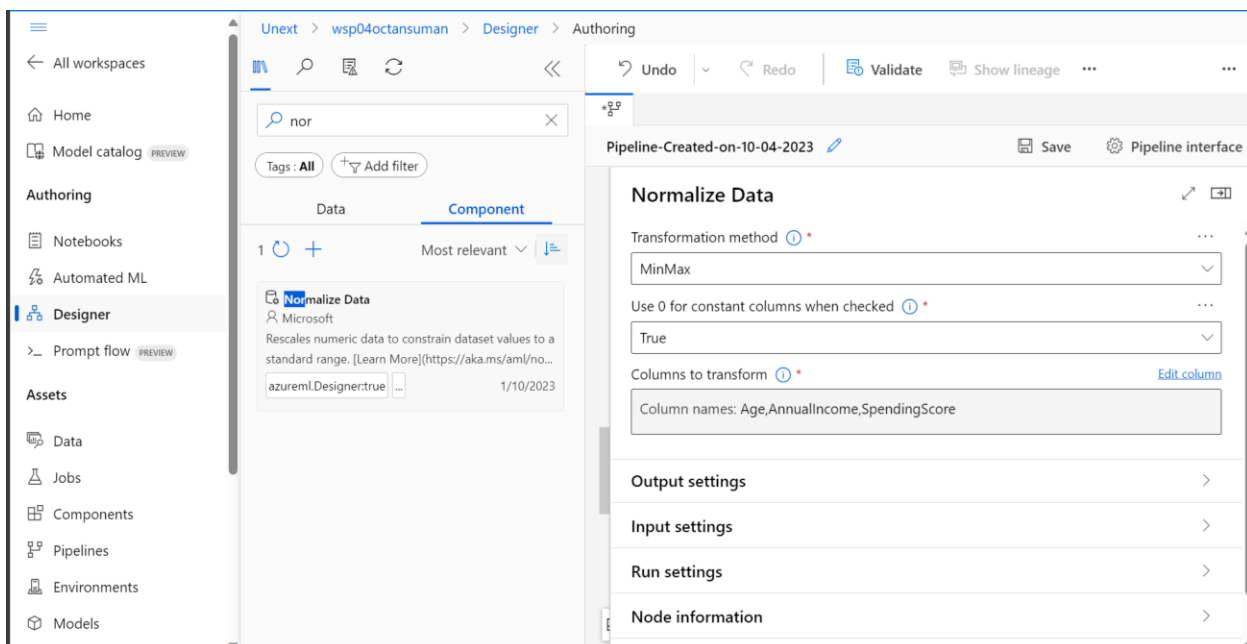
- Removing entire row if data is missing.



NORMALISING



- MinMax Normalisation



The screenshot shows the Azure ML Designer interface. On the left, the 'Designer' tab is selected in the sidebar. The main workspace displays a pipeline titled 'Pipeline-Created-on-10-04-2023'. The pipeline consists of two components: 'Clean Missing Data' (clean_missing_data) and 'Normalize Data' (normalize_data). The 'Clean Missing Data' component is connected to the 'Normalize Data' component. The 'Normalize Data' component has a 'Parameters' panel on the right. The interface includes a search bar at the top with the text 'nor', a 'Tags: All' filter, and a 'Component' tab. The pipeline is visualized with a flow from a 'Dataset' input to the 'Clean Missing Data' component, then to the 'Normalize Data' component, and finally to a 'Transformed d...' output.

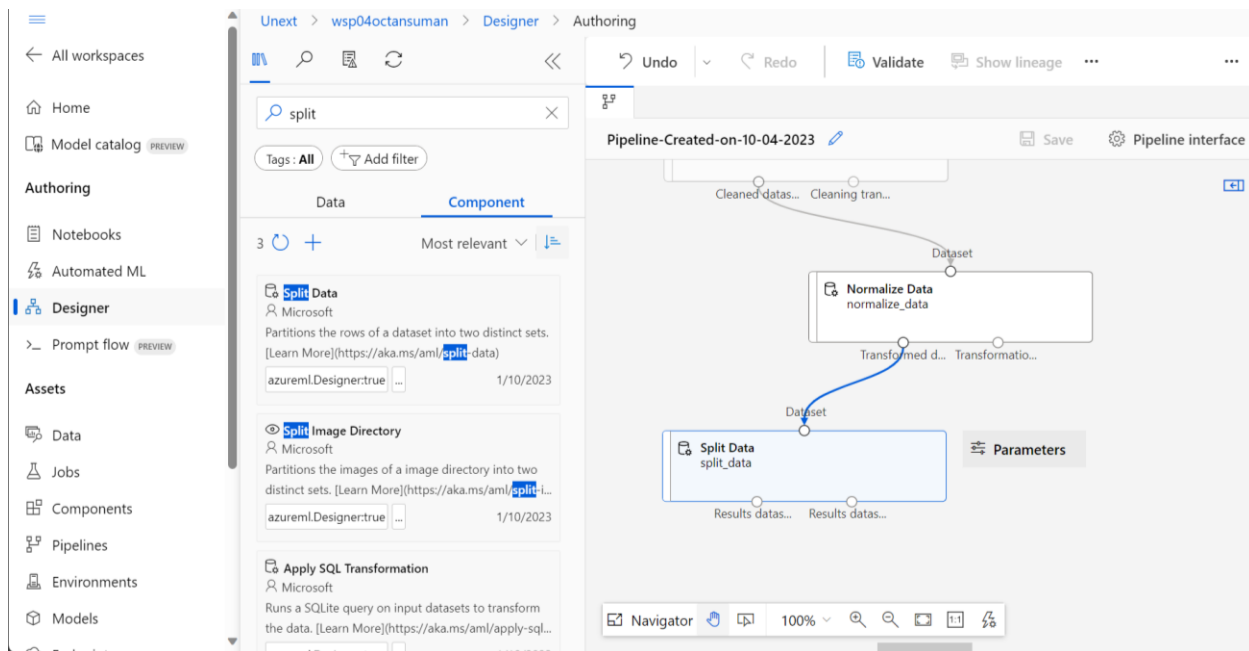
SPLITTING DATA

80:20 Split

The screenshot shows the Azure ML Designer interface with the 'Split Data' component selected. The 'Split Data' component is shown in the 'Component' tab of the sidebar. The main workspace displays the 'Split Data' component's configuration panel. The configuration includes the following settings:

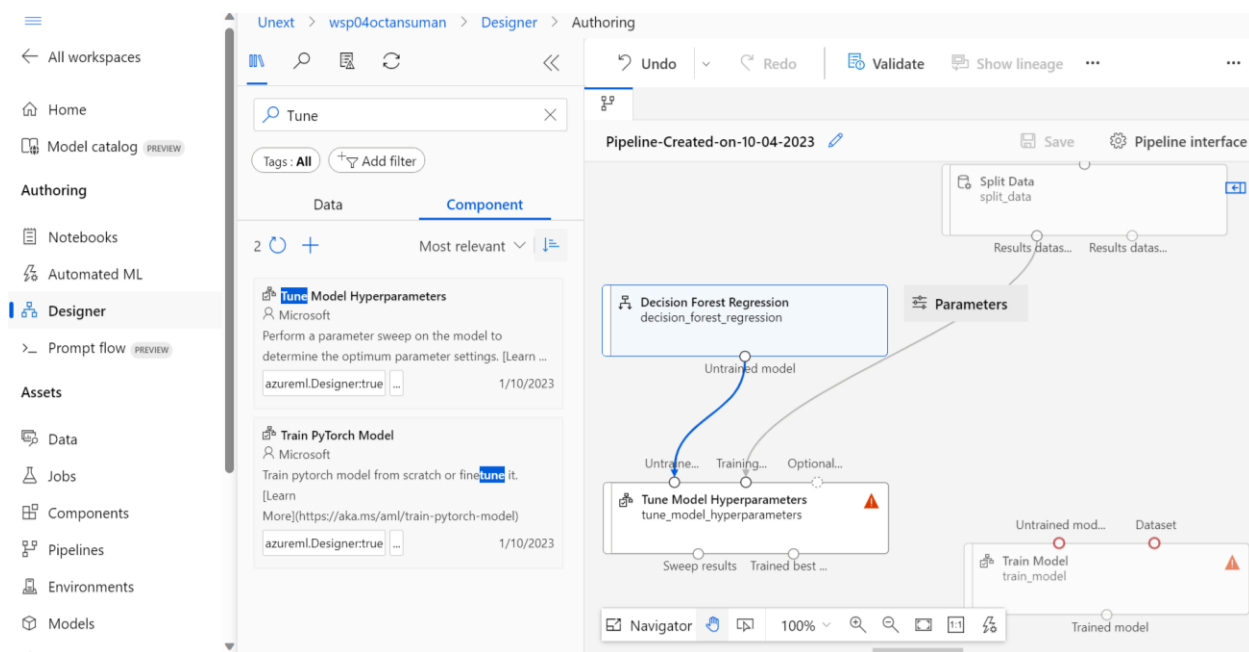
- Splitting mode:** Split Rows
- Fraction of rows in the first output dataset:** 0.80
- Randomized split:** True
- Random seed:** 1234
- Stratified split:** False

The 'Output settings' and 'Input settings' sections are also visible at the bottom of the configuration panel. The pipeline is visualized with a flow from a 'Dataset' input to the 'Split Data' component, and then to two output datasets.



MODEL

Decision Tree Regressor



HYPER PARAMETER TUNNING

- Selecting range of hyperparameters

Unext > wsp04octansuman > Designer > Authoring

Tags: All Add filter

Data Component

2 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 fine-tune it. [Learn More](https://aka.ms/aml/train-pytorch-model)
azureml.Designer:true 1/10/2023

Pipeline-Created-on-10-04-2023 Save Pipeline interface

Decision Forest Regression

Create trainer mode ⓘ *

ParameterRange

Number of decision trees ⓘ *

1; 8; 32

Maximum depth of the decision trees ⓘ *

1; 16; 64

Minimum number of samples per leaf node ⓘ *

1; 4; 16

Resampling method ⓘ *

Bagging Resampling

Output settings >

Input settings >

Providing hyper parameters

Unext > wsp04octansuman > Designer > Authoring

Tags: All Add filter

Data Component

2 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 fine-tune it. [Learn More](https://aka.ms/aml/train-pytorch-model)
azureml.Designer:true 1/10/2023

Pipeline-Created-on-10-04-2023 Save Pipeline interface

Tune Model Hyperparameters

Random sweep

Maximum number of runs on random sweep ⓘ *

5

Random seed ⓘ *

0

Metric for measuring performance for classification ⓘ *

AUC

Metric for measuring performance for regression ⓘ *

Coefficient of determination

Label column ⓘ *

Column names: SpendingScore [Edit column](#)

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

2 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 fine-tune it. [Learn ...]
More[https://aka.ms/aml/train-pytorch-model]
azureml.Designer:true 1/10/2023

Pipeline-Created-on-10-04-2023

Save Pipeline interface

split_data

Results datas... Results datas...

Decision Forest Regression
decision_forest_regression

Untrained model

Untrain... Training... Optional...

Tune Model Hyperparameters
tune_model_hyperparameters

Sweep results Trained best ...

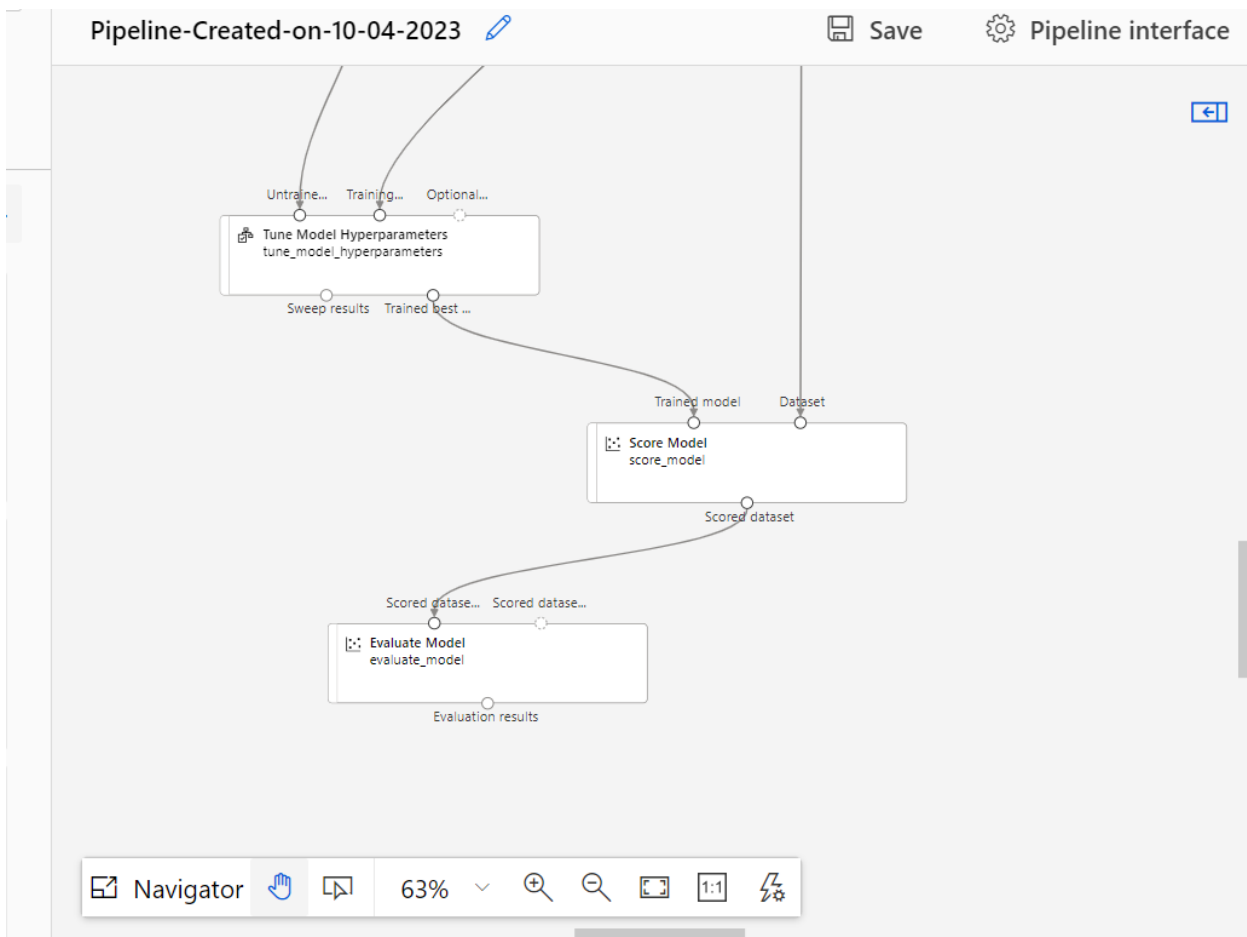
Untrained mod... Dataset

Train Model
train_model

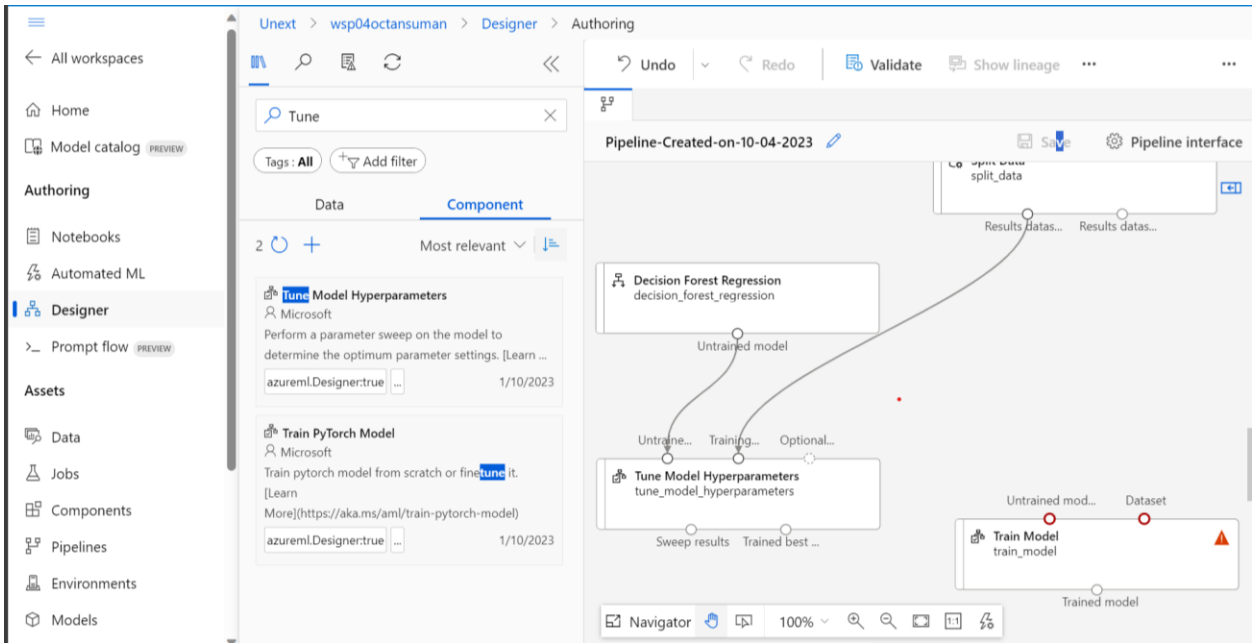
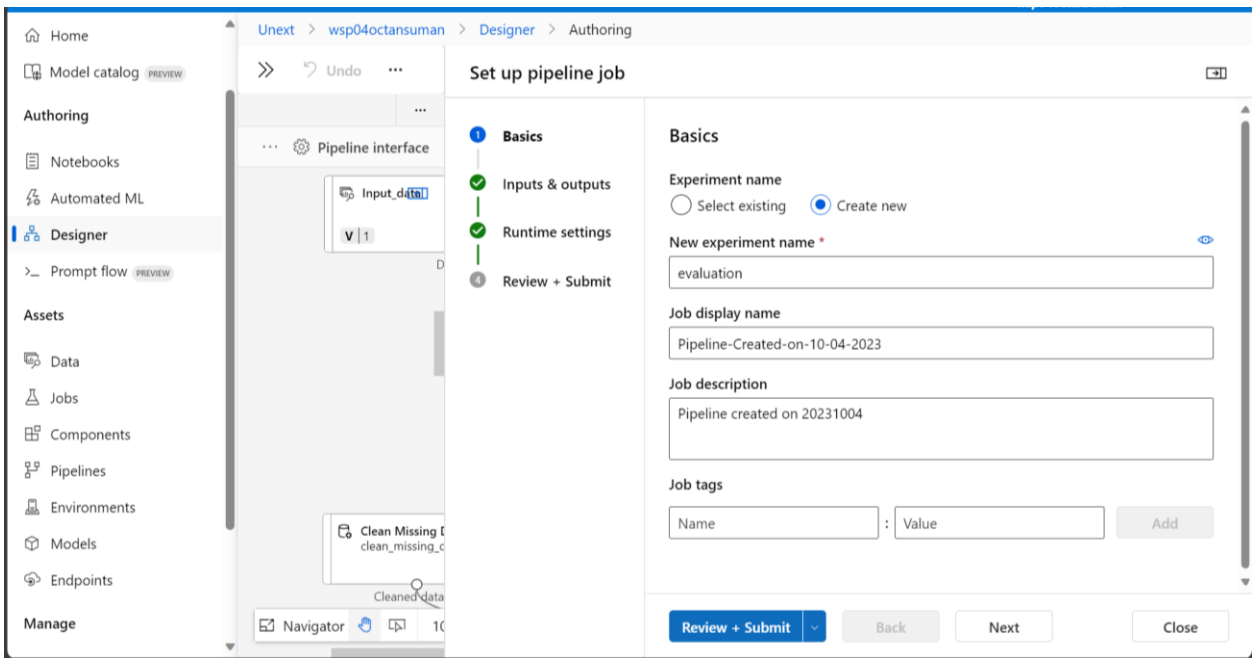
Trained model

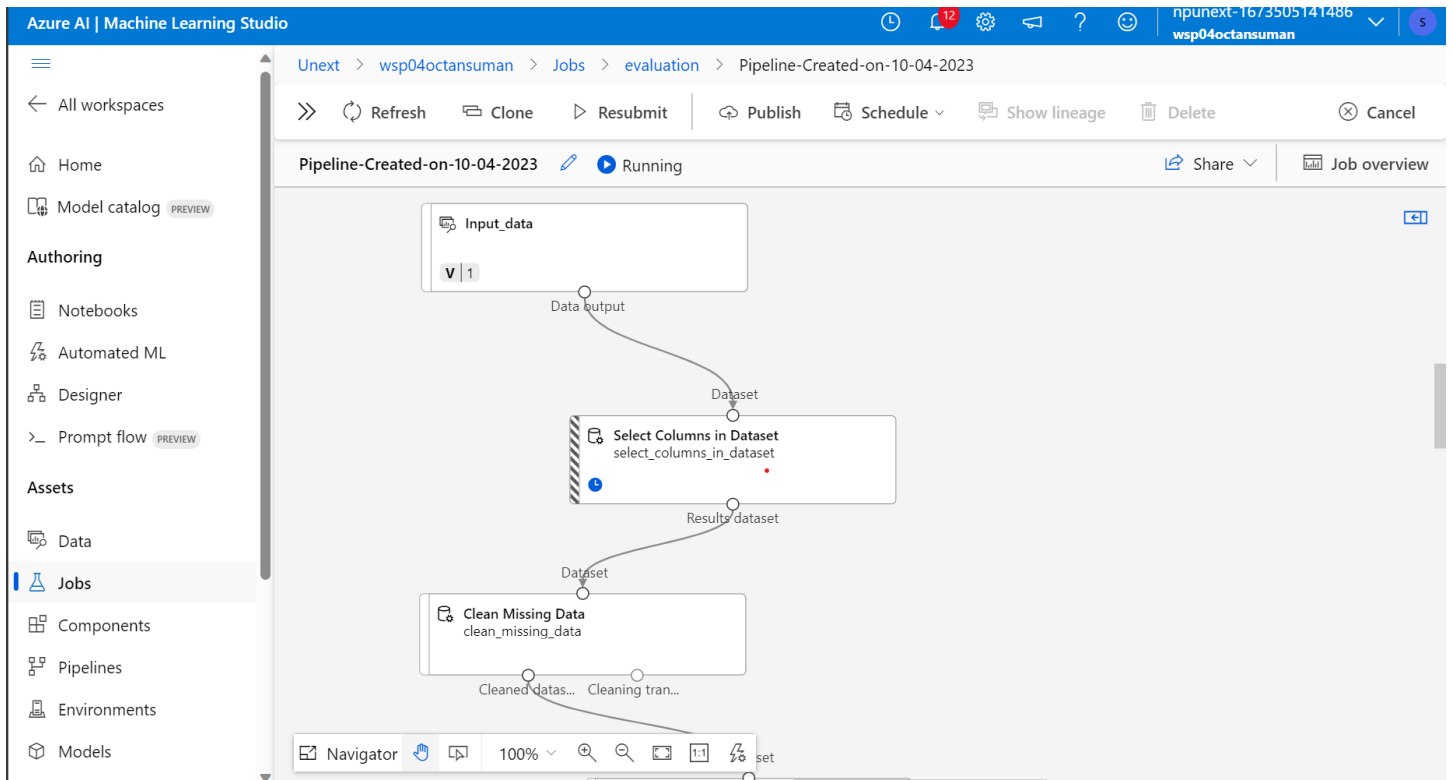
Navigator 100% 1:1

SCORING AND EVALUATING MODEL



RUNNING PIPELINE





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.