

**Department of Computer and Information Sciences**

**KV4004 – AI Fundamentals**

**Workshop 3**

**October 2024**

**1**

**Explore Classification with Azure Machine Learning Designer**

## In this workshop, you will learn how to develop a classification model with Azure Machine Learning designer that predicts whether the patient has a diabetes or not based on some characteristics given in the dataset about each subject. You will accomplish this by using the designer as we did in the previous workshop with regression. This time we will need to create the dataset first from the datasets provided in Microsoft Azure assets.

*Before proceeding, make sure you have created an Azure Machine Learning workspace and com- putting instance.*

A

# Exercise 1: Creating the Diabetes Dataset from Azure assets

1. Sign in to Azure VM via [**https://msle.learnDondemand.net**](https://msle.learnDondemand.net) using your Microsoft VM account credentials.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a login box

Description automatically generated

1. Click ‘My Training’, and choose ‘AI Fundamentals 1 KV4004 (DP-100)’ class

A screenshot of a computer

Description automatically generated

1. Launch the first VM ‘Explore the Azure Machine Learning workspace’

A screenshot of a computer

Description automatically generated

1. Log into the MS Azure ( portal.azure.com )with the provided username & pwd. Create an Azure ML studio and launch it. (If you cannot remember how to create it, please check the instruction **in ‘AI Fundamentals Lab 1-part 1.pdf’** in week 1.)
2. In Azure Machine Learning studio, expand the left pane by selecting the menu icon at the top left of the screen. Select the Data page (under Assets).

The Data page contains specific data files or tables that you plan to work with in Azure ML. You can create datasets from this page as well.

1. Download the ‘**Diabetes.csv’** file from week3 on Blackboard
2. On the Data page, under the Data assets tab, select + Create. Then configure a data asset with the following settings:

## Data type:

Name: diabetes-data

Description: Diabetes data

Dataset type: tabular

* + **Data source**: From Local

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

## Upload ‘Diabetes.csv’

## A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated

## A screenshot of a computer Description automatically generated

## Settings:

File format: Delimited Delimiter: Comma Encoding: UTF-8

Column headers: Only first file has headers Skip rows: None

Dataset contains multi-line data: do not select

A screenshot of a computer

Description automatically generated

## Schema:

Include all columns other than Path

Review the automatically detected types

A screenshot of a computer

Description automatically generated

## Review

* + Select Create

1. After the dataset has been created, open it and view the Explore page to see a sample of the data. This data represents details from patients who have been tested for diabetes.

As you can see from the data explore tab, some characteristics about each patient such as number of pregnancies, plasma glucose, blood pressure, etc. These are the clinical variables that will be used to predict whether the subject is diabetic or not. The output variable we want to predict is Diabetic that contains the values 1s and 0s as this is a categorical variable not numeric as in regression. The model predictions will also be different from regression and this model prediction will be probabilities of whether it’s a class 1 or class 0. If the *probability <* 0*.*5, then the predicted class is 0. If it is *≥* 0*.*5, then the predicted class is 1.

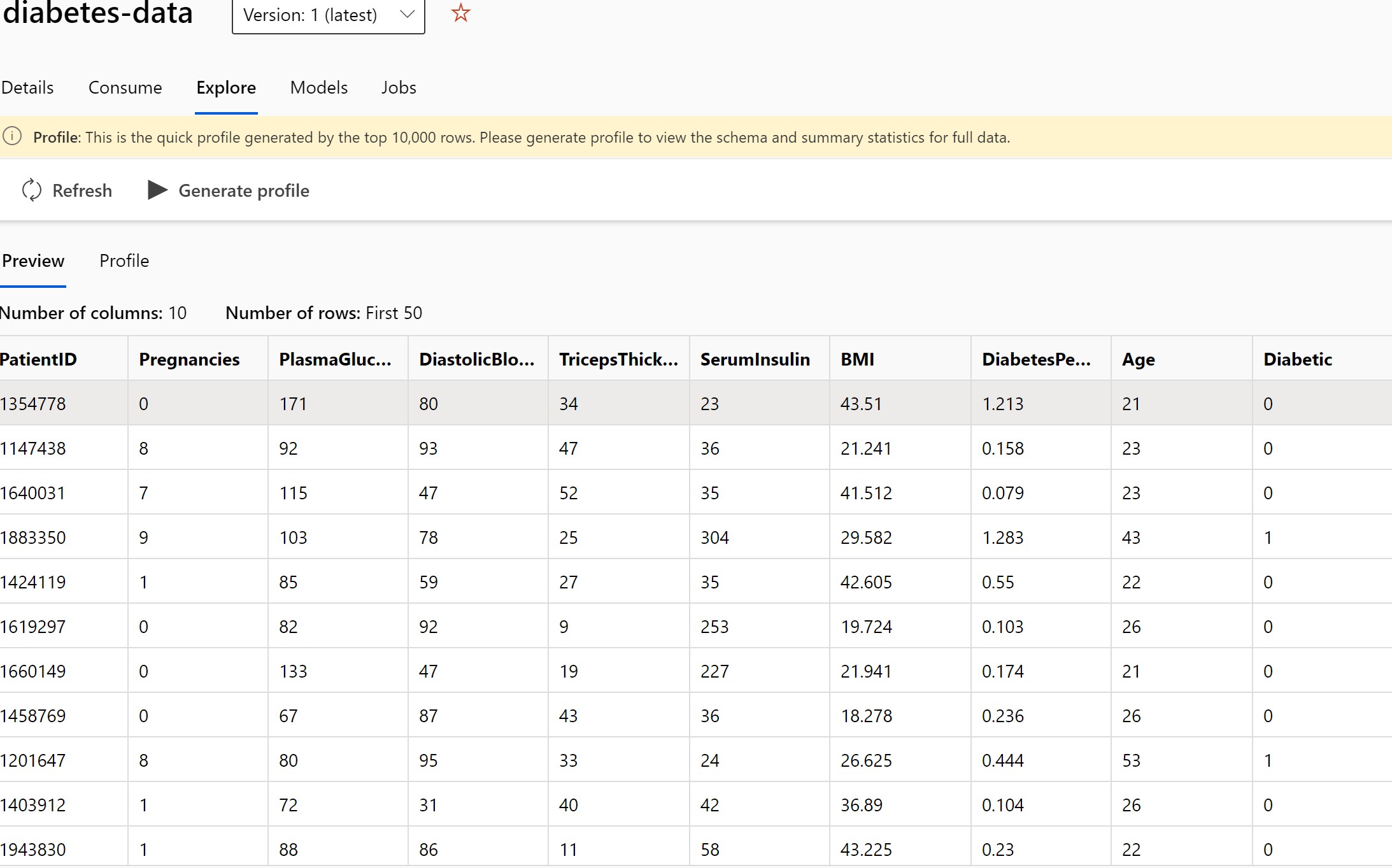
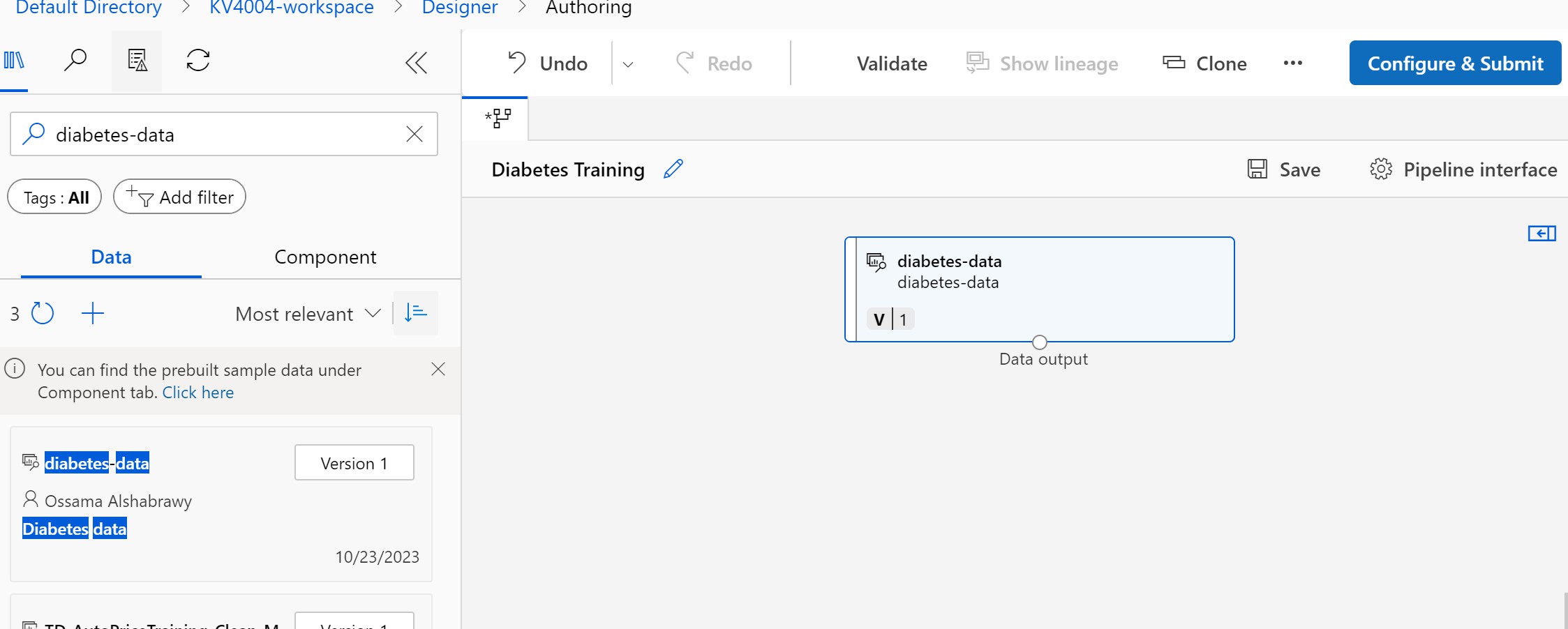


Fig. 1.1: Diabetic dataset features and label.

# Exercise 2: Creating the Diabetes Dataset from Azure assets

1. In Azure Machine Learning studio, on the left pane select the Designer item (under Authoring), and then select + to create a new pipeline.
2. Change the draft name from **Pipeline-Created-on-date** to **Diabetes Training**.
3. Add the dataset into the canvas. Select Data. Search for and place the **diabetes-data** dataset onto the canvas.



adding the dataset we created into the canvas.

1. Right-click the diabetes-data dataset on the canvas, and select **Preview** data.
2. Review the schema of the data in the Profile tab, noting that you can see the distributions of the various columns as histograms.
3. Scroll back up and review the other columns, which represent the features that will be used to predict the label. Note that most of these columns are numeric, but each feature is on its own scale. For example, **Age** values range from 21 to 77, while **DiabetesPedigree** values range from 0.078 to 2.3016. When training a machine learning model, it is sometimes possible for larger values to dominate the resulting predictive function, reducing the influence of features that on a smaller scale. Typically, data scientists mitigate this possible bias by normalizing the numeric columns so they’re on the similar scales.
4. Close the **DataOutput** tab so that you can see the dataset on the canvas.

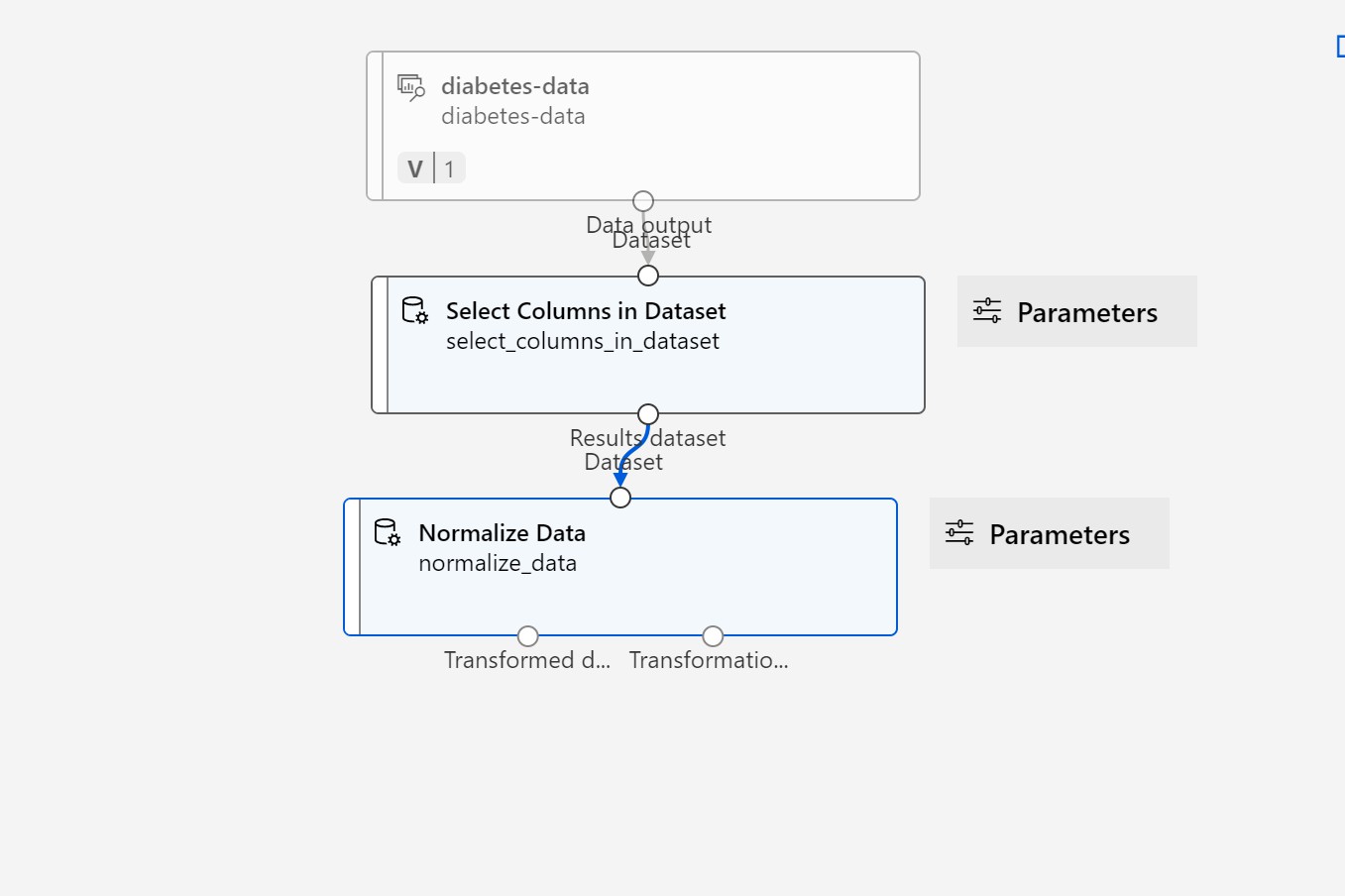
# Exercise 3: Adding Transformation

1. In the **Asset library** pane on the left, select **Component**
2. Find the **Select Columns in Dataset** module and place it on the canvas below the **diabetes-data** dataset. Then connect the output from the bottom of the **diabetes-data** dataset to the input at the top of the Select Columns in Dataset module.
3. Double click on the Select Columns in Dataset module to access a settings pane on the right. Select **Edit column**. Then in the Select columns window, select **By name** and **Add all** the columns. Then remove **PatientID** and click **Save**.

A screenshot of a computer

Description automatically generated

1. Find the **Normalize Data** module and place it on the canvas below the **Select Columns in Dataset** module. Then connect the output from the bottom of the **Select Columns** in Dataset module to the input at the top of the Normalize Data module as in the following figure.

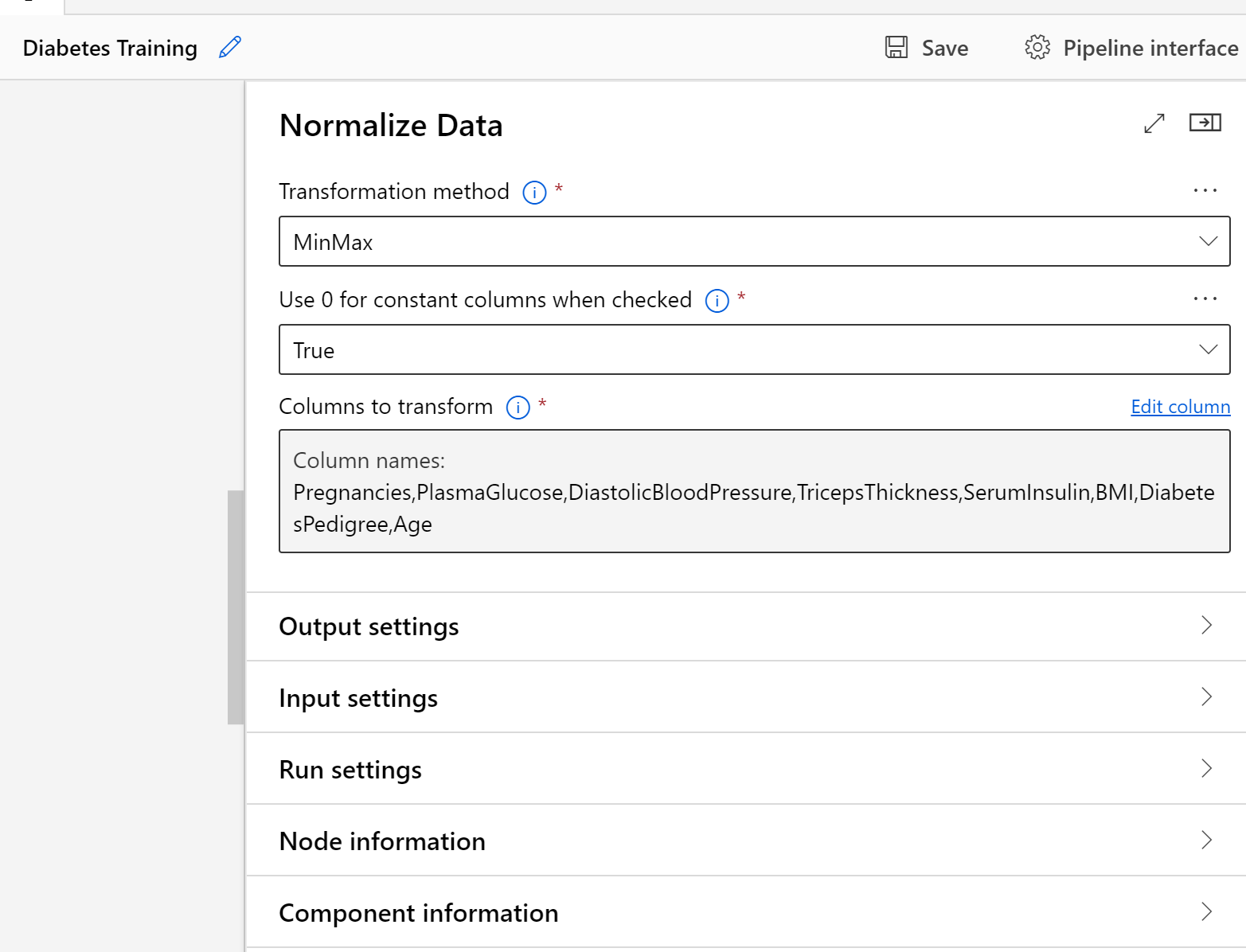


the pipeline so far.

1. Double-click the Normalize Data module to view its settings, noting that it requires you to specify the transformation method and the columns to be transformed.
2. Set the Transformation method to **MinMax** and the Use 0 for constant columns when checked to **True**. Edit the columns to transform with Edit columns. Select columns **With Rules** and copy and paste the following list under include column names:

Pregnancies, PlasmaGlucose, DiastolicBloodPressure, TricepsThickness, SerumInsulin, BMI, DiabetesPedigree, Age

1. Click Save



Normalize Data settings.

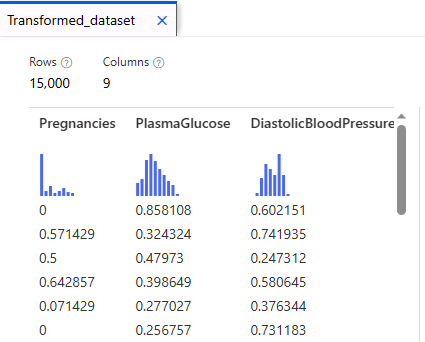
The data transformation is normalizing the numeric columns to put them on the same scale, which should help prevent columns with large values from dominating model training.

# Run the pipeline

1. Select Configure and Submit at the top of the page to open the Set-up pipeline job dialogue.
2. On the Basics page select **Create new** and set the name of the experiment to **diabetes-training** then select **Next**.
3. On the Inputs and outputs page select **Next** without making any changes.
4. On the Runtime settings page an error appears as you don´t have a default compute to run the pipeline. In the Select compute type drop-down select Compute instance you’ve just create it or create new if none there.

# View the transformed data

1. From Jobs, choose the job that contains the pipeline so far. Right-click the **Normalize Data** module on the canvas, and select Preview data. Select Transformed dataset.
2. View the data, noting that the numeric columns you selected have been normalized to a common scale.



Preview of the normalized data

# Exercise 4: Add training modules

Return to the Designer and select the **Diabetes Training pipeline**. Add the components as the figure and steps below:

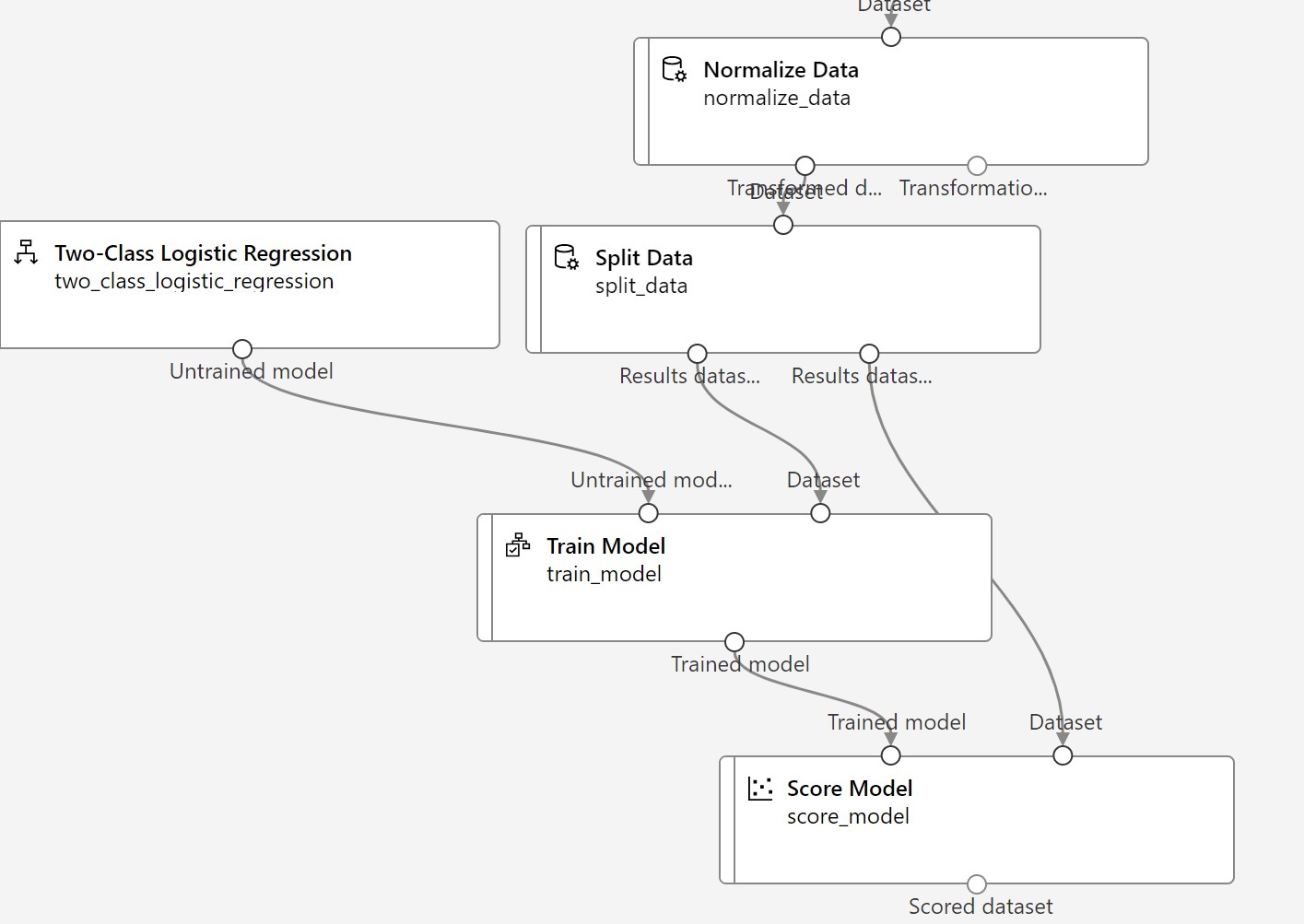


Fig. 1.6: training pipeline

1. In the Asset library pane on the left, in Component, search for and place a **Split Data** module onto the canvas under the Normalize Data module. Then connect the Transformed Dataset (left) output of the Normalize Data module to the input of the Split Data module.
2. Select the **Split Data** module, and configure its settings as follows: Splitting mode: Split Rows

Fraction of rows in the first output dataset: 0.7 Randomized split: True

Random seed: 123 Stratified split: False

1. In the Asset library, search for and place a **Train Model** module to the canvas, under the Split Data module. Then connect the Results dataset1 (left) output of the Split Data module to the Dataset (right) input of the Train Model module.
2. The model we’re training will predict the **Diabetic** value, so select the Train Model module and modify its settings to set the Label column to **Diabetic**.
3. In the Asset library, search for and place a **Two-Class Logistic Regression** module to the canvas, to the left of the Split Data module and above the Train Model module. Then connect its output to the Untrained model (left) input of the Train Model module.
4. To test the trained model, we need to use it to score the validation dataset we held back when we split the original data - in other words, predict labels for the features in the validation dataset.

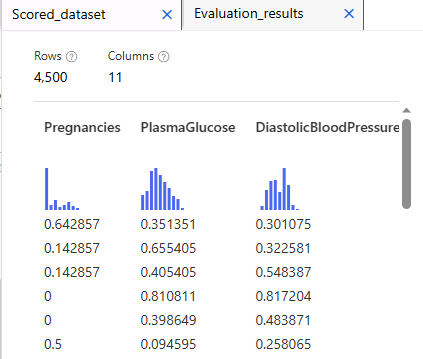
In the **Asset library**, search for and place a **Score Model** module to the canvas, below the Train Model module. Then connect the output of the Train Model module to the Trained model (left) input of the Score Model module; and connect the Results dataset2 (right) output of the Split Data module to the Dataset (right) input of the Score Model module.

# Run the training pipeline

1. Select **Configure and Submit**, and run the pipeline using the existing experiment named **diabetes- training**.
2. Check the status of the job by selecting **Jobs** under the Assets. From there, select the **diabetes-training**

experiment and then select the latest *Diabetes Training job*.

1. On the new tab, right-click the **Score Model** module on the canvas, select **Preview data** and then select Scored dataset to view the results.



score model output. The probabilities and class predictions

As you can see from the preview data that the output of the model is just probabilities of whether the data point is class 0 or class 1 as explained previously. What did you notice?

# Validation stage: Add an Evaluate Model module

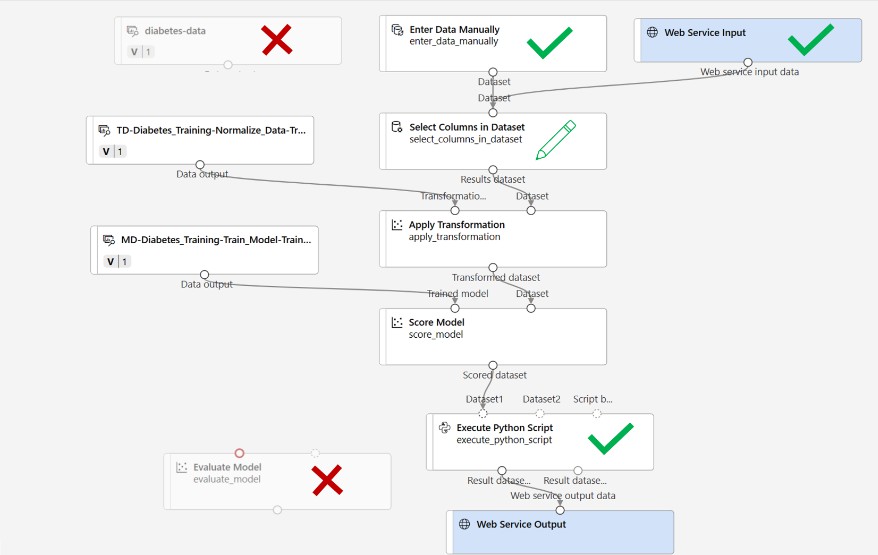
1. Return to **Designer** and open the **Diabetes Training** pipeline you created.
2. In the **Asset library**, search for and place an **Evaluate Model** module to the canvas, under the Score Model module, and connect the output of the Score Model module to the Scored dataset (left) input of the Evaluate Model module.
3. Select **Configure and Submit**, and run the pipeline using the existing experiment named diabetes-training.
4. Check the status of the job by selecting **Jobs** under the Assets. From there, select the **diabetes-training** experiment and then select the latest Diabetes Training job.
5. On the new tab, right-click the **Evaluate Model** module on the canvas, select **Preview data** then select **Evaluation results** to view the performance metrics. These metrics can help data scientists assess how well the model predicts based on the validation data.
6. Review the metrics to the left of the confusion matrix, which include:

* **Accuracy**: In other words, what proportion of diabetes predictions did the model get right?
* **Precision**: In other words, out of all the patients that the model predicted as having diabetes, the per- centage of time the model is correct.
* **Recall**: In other words, out of all the patients who actually have diabetes, how many diabetic cases did the model identify correctly?
* **F1 Score**: is kind of averaging for both of the precision and recall.

# Exercise 5: Create an inference pipeline

1. Locate the menu above the canvas and select **Create inference pipeline**.
2. In the **Create inference pipeline** drop-down list, select **Real-time inference pipeline**. After a few seconds, a new version of your pipeline named Diabetes Training-real time inference will be opened.
3. Rename the new pipeline to **Predict Diabetes**, and then review the new pipeline. Design it as below:

* Add a **web service input** component for new data to be submitted.
* Replace the **diabetes-data** dataset with an **Enter Data Manually** module that doesn’t include the label column (Diabetic).
* Edit the columns selected in the **Select Columns in Dataset** module and remove **Diabetic** column. Remove the **Evaluate Model** module.
* Insert an **Execute Python Script** module before the **web service output** to return only the patient ID, predicted label value, and probability.



Inference designer

1. In **Enter Data Manually**, insert the following data to be used for the inference where it does not include the label this time as we want to predict it.

PatientID, Pregnancies, PlasmaGlucose, DiastolicBloodPressure, TricepsThickness, SerumInsulin, BMI, DiabetesPedigree, Age

1882185,9,104,51,7,24,27.36983156,1.350472047,43

1662484,6,73,61,35,24,18.74367404,1.074147566,75

1228510,4,115,50,29,243,34.69215364,0.741159926,59

Please note that we you insert the data, it’s better to add line by line in the Enter Data Manually. The reason is sometimes when you copy the data, you copy it as one line which will be read as headers but you want the first line as headers followed by lines of the data which represent the rows.

1. Connect the new **Enter Data Manually** module to the same Dataset input of the Select Columns in Dataset module as the **Web Service Input**.
2. Edit the **Select Columns in Dataset** module. Remove Diabetic from the Selected Columns.
3. Delete the connection between the **Score Model** module and the **Web Service Output**. Add an **Execute Python Script** module, replacing all of the default python script with the following code (which selects only the PatientID, Scored Labels and Scored Probabilities columns and renames them appropriately). Add the following code below to **Execute Python Script** by double clicking on it.

import pandas as pd

def azureml\_main(dataframe1 = None, dataframe2 = None):

scored\_results = dataframe1[['Scored Labels', 'Scored Probabilities']] scored\_results.rename(columns={'Scored Labels':'DiabetesPrediction',

'Scored Probabilities':'Probability'}, inplace=True)

return scored\_results

1. Connect the output from the Score Model module to the Dataset1 (left-most) input of the Execute Python Script, and connect the Result dataset (left) output of the Execute Python Script module to the Web Service Output.
2. Run the pipeline as a **new experiment** named **diabetes-inference** on your compute cluster. The experiment may take a while to run.
3. Return to the Jobs tab. From there, select the **diabetes-inference** experiment and then select the **Predict Diabetes** job.
4. When the pipeline has completed, select the **Execute Python Script** module. Select the **Preview data** and **select Result** dataset to *see the predicted labels and probabilities for the three patient observations in the input data*.

