Challenging Task – 4

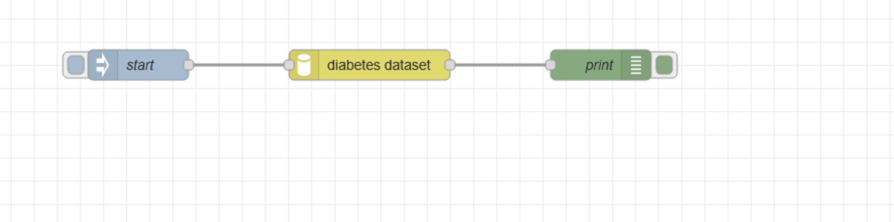
Name: Gangireddy Madhurima

Reg Number: 21MIS1155

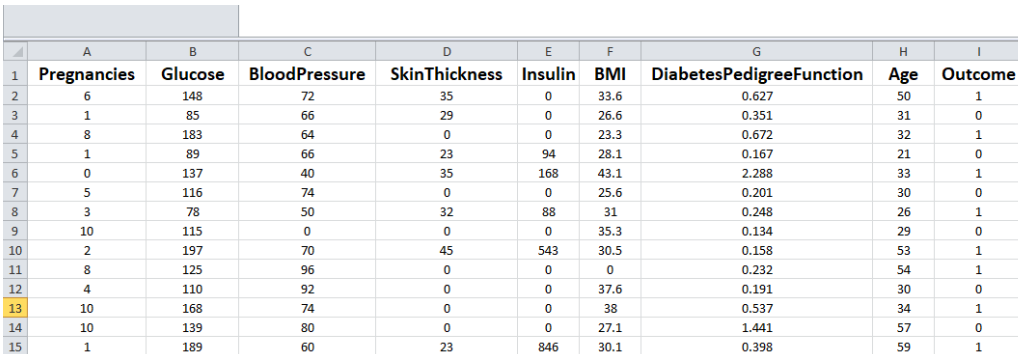
Step 1: Import Data Set into Node-Red

1. Install node-red-contrib-machine-learning in Node-Red

2. Set up the nodes to read dataset.

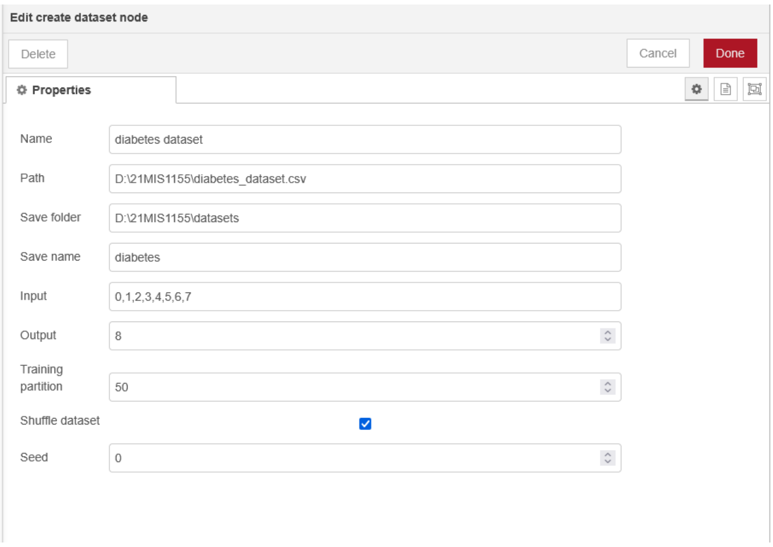


3. Dataset – diabetes_dataset.csv



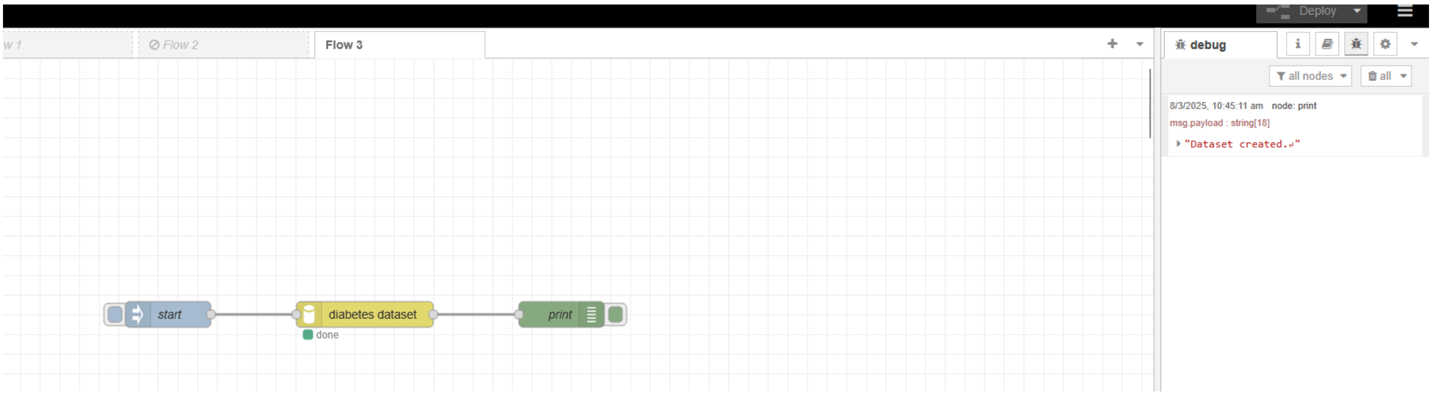
	A	B	C	D	E	F	G	H	I
1	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1
11	8	125	96	0	0	0	0.232	54	1
12	4	110	92	0	0	37.6	0.191	30	0
13	10	168	74	0	0	38	0.537	34	1
14	10	139	80	0	0	27.1	1.441	57	0
15	1	189	60	23	846	30.1	0.398	59	1

4. Add the details of dataset in dataset node



Name	diabetes dataset
Path	D:\21MIS1155\diabetes_dataset.csv
Save folder	D:\21MIS1155\datasets
Save name	diabetes
Input	0,1,2,3,4,5,6,7
Output	8
Training partition	50
Shuffle dataset	☑
Seed	0

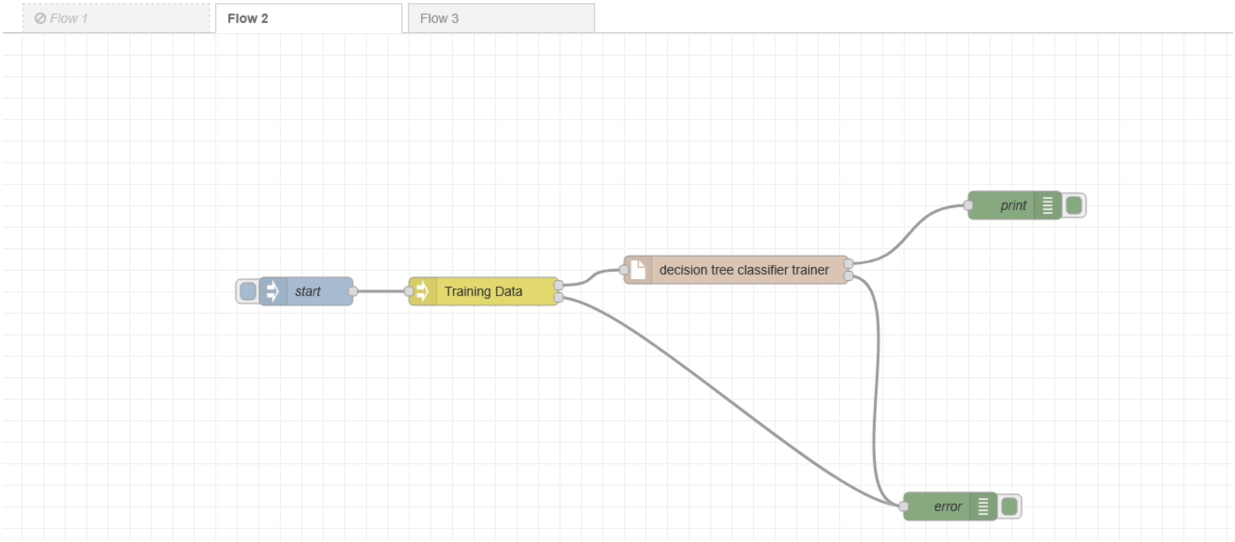
5. Dataset imported to Dataset Node

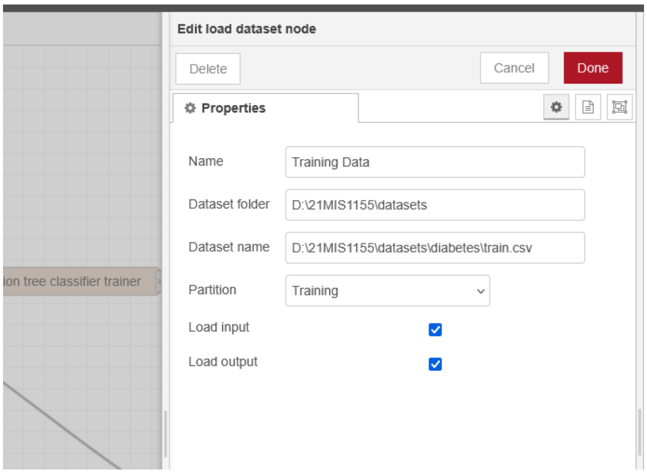


```
8/3/2025, 10:45:11 am   node: print
msg.payload : string[18]
"Dataset created.↵"
```

Step 2: Training the Data

1. Set up the Nodes and keep train.csv for Training





Edit load dataset node

Name: Training Data

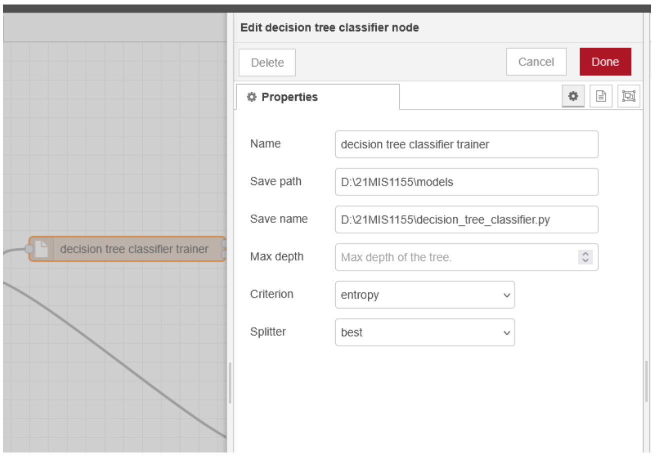
Dataset folder: D:\21MIS1155\datasets

Dataset name: D:\21MIS1155\datasets\diabetes\train.csv

Partition: Training

Load input ☑

Load output ☑



Edit decision tree classifier node

Name: decision tree classifier trainer

Save path: D:\21MIS1155\models

Save name: D:\21MIS1155\decision_tree_classifier.py

Max depth: Max depth of the tree.

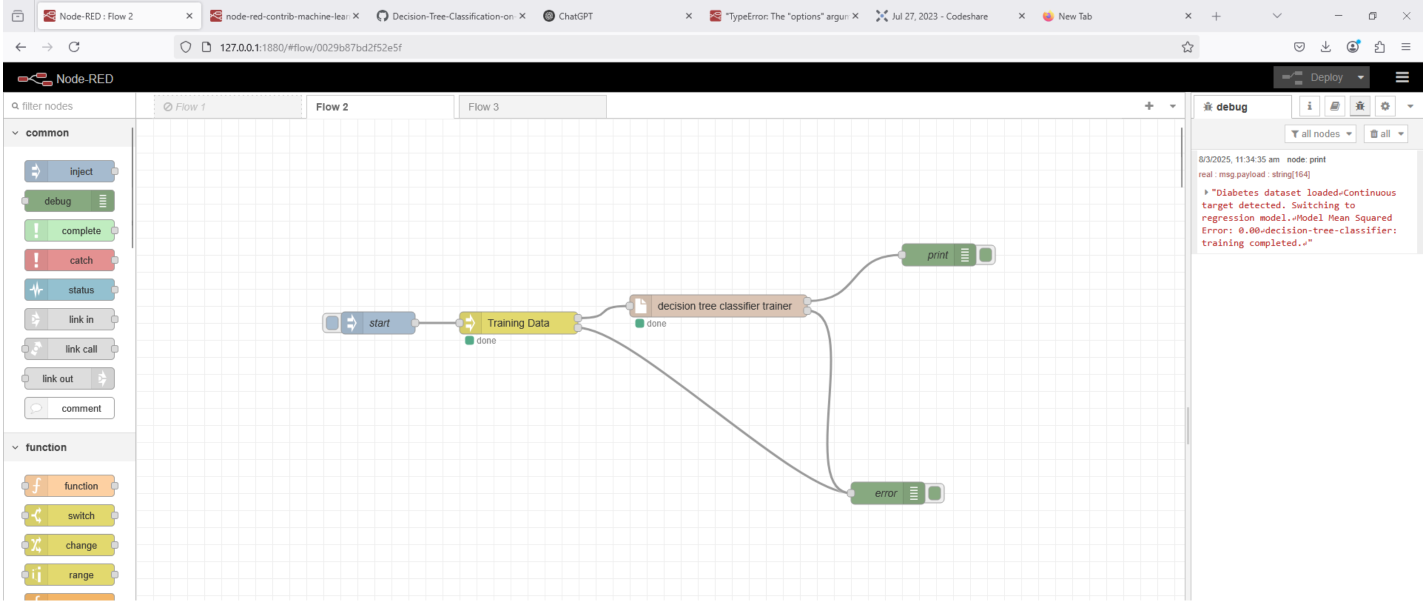
Criterion: entropy

Splitter: best

2. trainer.py code:

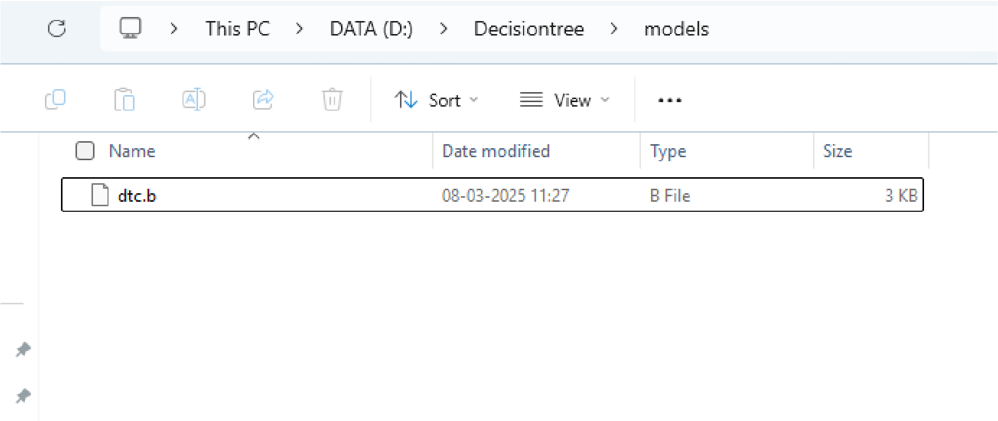
```
import json
import pandas as pd
import os
import sys
from io import StringIO  # Import StringIO to handle the literal JSON string
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Set paths and imports
sys.path.append(os.path.dirname(os.path.realpath(__file__)) + '/../../utils')
from sklw import SKLW

OUTLIER_DETECTORS = ['elliptic-envelope-classifier', 'isolation-forest-classifier', 'one-class-support-vector-classifier']

# Read configurations
config = json.loads(input())
save = config['save']

while True:
    # Read request (if provided JSON or file)
    data = input()

    try:
        # Convert the literal JSON string to a StringIO object and read it as JSON
        json_data = StringIO(data)  # Wrap the JSON string in StringIO
        df = pd.read_json(json_data, orient='values')
    except Exception as e:
        # Handle the case where the request is a file
        try:
            df = pd.read_csv(json.loads(data)['file'], header=None)
        except Exception as e2:
            print(f"Error loading data: {e2}")
            continue  # Skip to the next iteration if the data is not loaded properly

    # Load Iris dataset if we are dealing with it specifically
    iris = load_iris()
    df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
    df['species'] = iris.target
    print("Diabetes dataset loaded")

    # Separate features (X) and labels (y)
    x = df.iloc[:, :-1]  # All columns except the last one
    y = df.iloc[:, -1]   # Last column as the label (species)

    # Check if target variable (y) is continuous (regression task) or categorical (classification)
    if y.dtype.kind in 'iuf':  # If y is integer or float (continuous)
        # If it’s continuous, check if we should treat it as a regression problem
        print("Continuous target detected. Switching to regression model.")
        is_regression = True
    else:
        # If y is categorical (discrete), treat it as a classification problem
        print("Categorical target detected. Using classification model.")
        is_regression = False

    # Initialize the classifier or regressor based on the task type
    model = None
    if is_regression:
        # For regression, check if 'criterion' is part of kwargs
        kwargs = config.get('kwargs', {})
        # Ensure that 'criterion' is not in kwargs if we're passing it explicitly
        if 'criterion' in kwargs:
            del kwargs['criterion']  # Remove 'criterion' from kwargs if it's set manually
        model = SKLW(path=save, model=DecisionTreeRegressor(criterion='squared_error', **kwargs))
    else:
        # Use classification models if the target is categorical
        classifier = None
        if config['classifier'] == 'decision-tree-classifier':
            model = DecisionTreeClassifier(criterion='gini', **config['kwargs'])

    # Split dataset into train and test
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

    try:
        # Train the model
        model.fit(x_train, y_train)

        # Predict using the trained model
        y_pred = model.predict(x_test)

        # For regression: Evaluate the model using Mean Squared Error (MSE)
        if is_regression:
            mse = mean_squared_error(y_test, y_pred)
            print(f"Model Mean Squared Error: {mse:.2f}")
        else:
            # For classification: Evaluate the model using accuracy
            from sklearn.metrics import accuracy_score
            accuracy = accuracy_score(y_test, y_pred)
            print(f"Model accuracy: {accuracy:.2f}")

    except Exception as e:
        print(f"Error during model training: {e}")
        continue  # Skip this iteration if training fails

    print(f"{config['classifier']}: training completed.")
```

3. Training Dataset



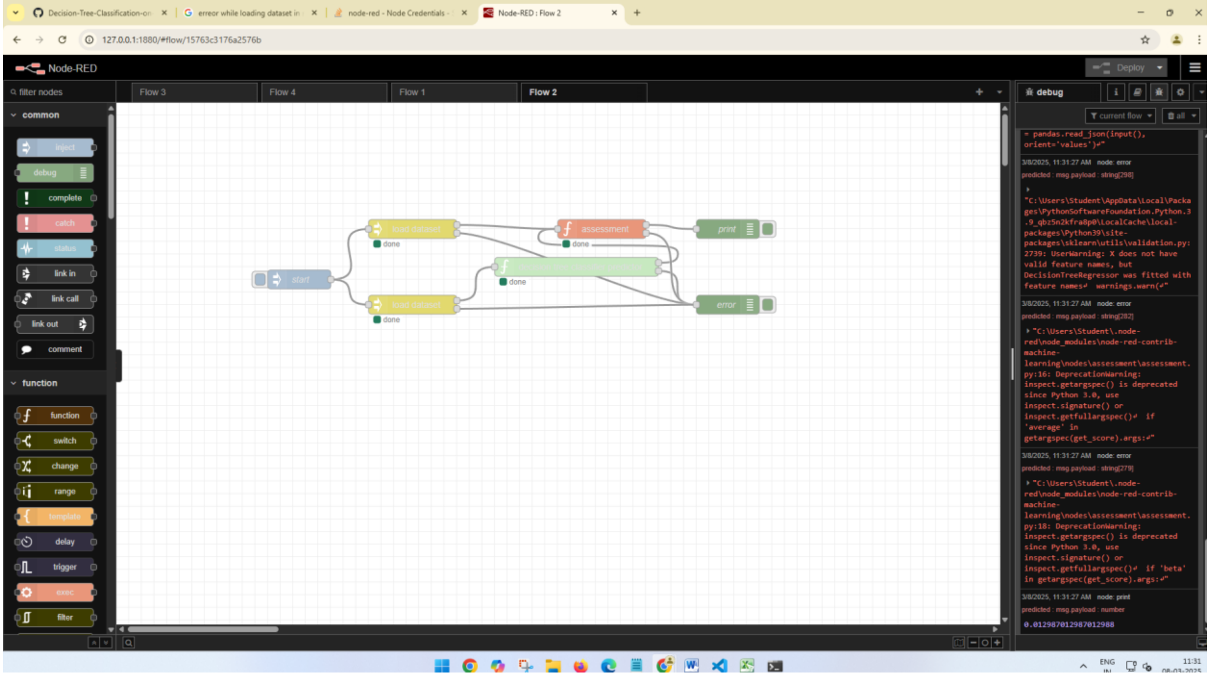
```
8/3/2025, 11:34:35 am   node: print
real : msg.payload : string[164]
"Diabetes dataset loaded↵Continuous
target detected. Switching to
regression model.↵Model Mean Squared
Error: 0.00↵decision-tree-classifier:
training completed.↵"
```

Trained Model

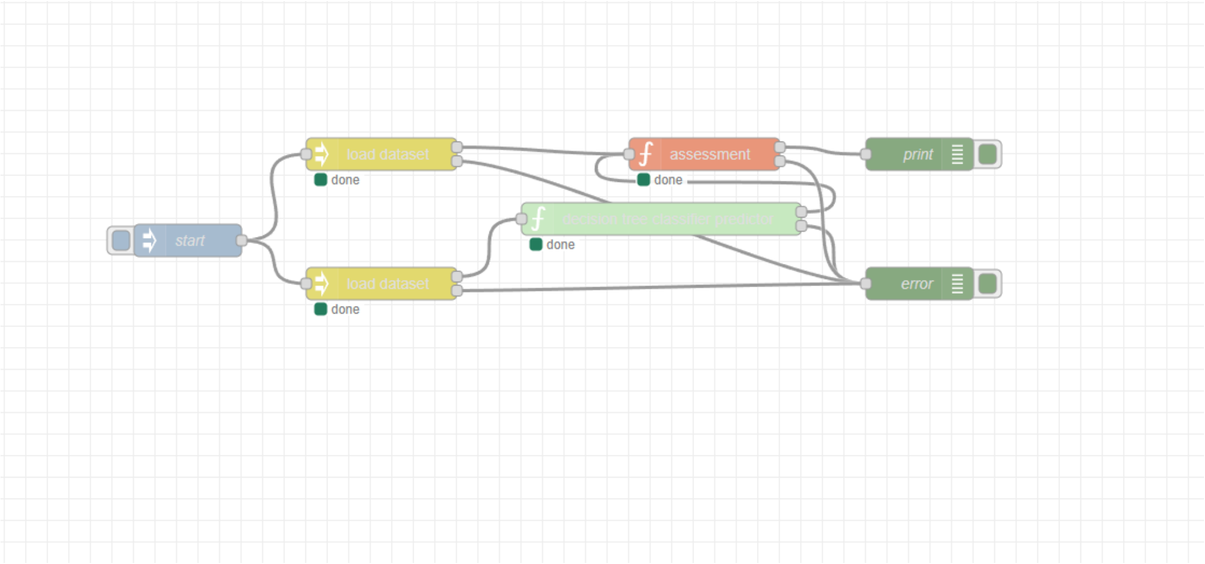


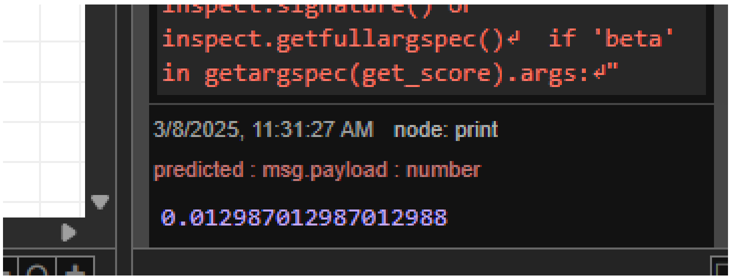
Name	Date modified	Type	Size
dtc.b	08-03-2025 11:27	B File	3 KB

Step 3: Testing the Model



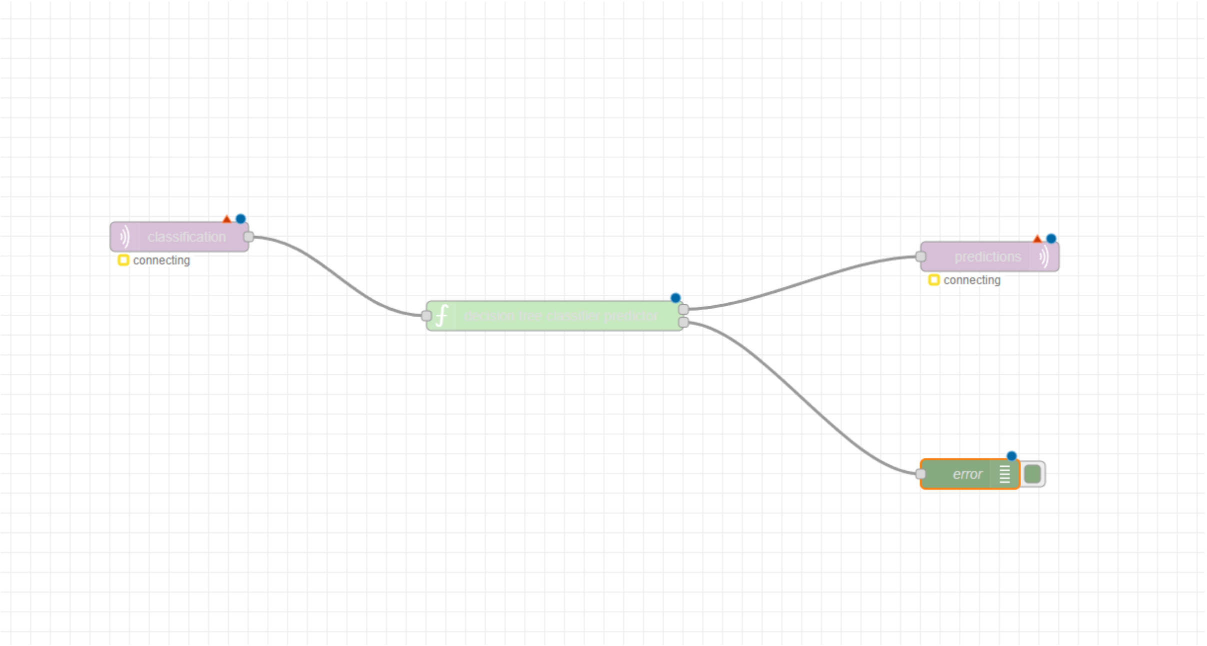
```
= pandas.read_json(input(),
orient='values')↵"

3/8/2025, 11:31:27 AM   node: error
predicted : msg.payload : string[298]
▸
"C:\Users\Student\AppData\Local\Packa
ges\PythonSoftwareFoundation.Python.3
.9_qbz5n2kfra8p0\LocalCache\local-
packages\Python39\site-
packages\sklearn\utils\validation.py:
2739: UserWarning: X does not have
valid feature names, but
DecisionTreeRegressor was fitted with
feature names↵  warnings.warn(↵"

3/8/2025, 11:31:27 AM   node: error
predicted : msg.payload : string[282]
▸ "C:\Users\Student\.node-
red\node_modules\node-red-contrib-
machine-
learning\nodes\assessment\assessment.
py:16: DeprecationWarning:
inspect.getargspec() is deprecated
since Python 3.0, use
inspect.signature() or
inspect.getfullargspec()↵  if
'average' in
getargspec(get_score).args:↵"

3/8/2025, 11:31:27 AM   node: error
predicted : msg.payload : string[279]
▸ "C:\Users\Student\.node-
red\node_modules\node-red-contrib-
machine-
learning\nodes\assessment\assessment.
py:18: DeprecationWarning:
inspect.getargspec() is deprecated
since Python 3.0, use
inspect.signature() or
inspect.getfullargspec()↵  if 'beta'
in getargspec(get_score).args:↵"

3/8/2025, 11:31:27 AM   node: print
predicted : msg.payload : number
0.012987012987012988
```

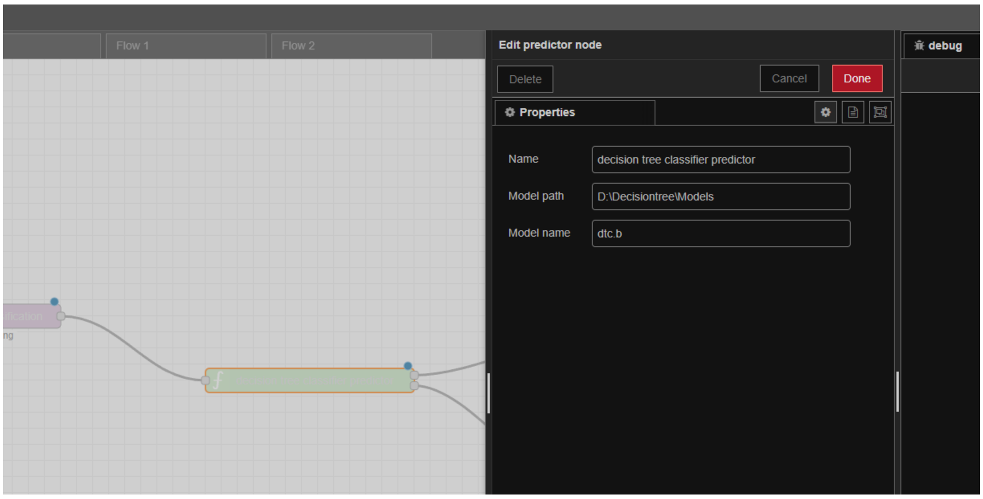




```
inspect.signature() or
inspect.getfullargspec()↵  if 'beta'
in getargspec(get_score).args:↵"

3/8/2025, 11:31:27 AM   node: print
predicted : msg.payload : number
0.012987012987012988
```

Step 4: MQTT Protocol Set Up





Name: decision tree classifier predictor
Model path: D:\Decisiontree\Models
Model name: dtc.b

JSON CODE:

```
[
  {
    "id": "da8ca300.2dfe6",
    "type": "create dataset",
     "z": "21ce826.2ff977e",
    "name": "",
    "path": "test/iris.data",
    "saveFolder": "test/datasets",
    "saveName": "iris",
    "input": "0,1,2,3",
    "output": "4",
    "trainingPartition": "",
        "shuffle": true,
        "seed": "",
        "x": 340,
        "y": 80,
        "wires": [
            [
                "4fb0a8dc.f6baf8"
            ]
        ]
    },
    {
        "id": "44b6f4b0.34d7dc",
        "type": "load dataset",
        "z": "21ce826.2ff977e",
        "name": "",
        "datasetFolder": "test/datasets",
        "datasetName": "iris",
        "partition": "train.csv",
        "input": true,
        "output": true,
        "x": 290,
        "y": 200,
        "wires": [
            [
                "26110acb.cbf526"
            ],
            [
                "86385870.9f6b88"
            ]
        ]
    },
    {
        "id": "4f7cc53d.87a22c",
        "type": "inject",
        "z": "21ce826.2ff977e",
        "name": "start",
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "x": 110,
        "y": 80,
        "wires": [
            [
                "da8ca300.2dfe6"
            ]
        ]
    },
    {
        "id": "d3e9e7ab.a06d68",
        "type": "inject",
        "z": "21ce826.2ff977e",
        "name": "start",
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "x": 110,
        "y": 200,
        "wires": [
            [
                "44b6f4b0.34d7dc"
            ]
        ]
    },
    {
        "id": "b21982e2.99cf1",
        "type": "inject",
        "z": "21ce826.2ff977e",
        "name": "start",
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "x": 110,
        "y": 440,
        "wires": [
            [
                "f1b47338.aab82",
                "1ea9f445.89d0bc"
            ]
        ]
    },
    {
        "id": "4fb0a8dc.f6baf8",
        "type": "debug",
        "z": "21ce826.2ff977e",
        "name": "print",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "x": 570,
        "y": 80,
        "wires": []
    },
    {
        "id": "86385870.9f6b88",
        "type": "debug",
        "z": "21ce826.2ff977e",
        "name": "error",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "x": 770,
        "y": 240,
        "wires": []
    },
    {
        "id": "2270c854.c34e08",
        "type": "debug",
        "z": "21ce826.2ff977e",
        "name": "print",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "x": 750,
        "y": 160,
        "wires": []
    },
    {
        "id": "e69a3271.c7cab",
        "type": "predictor",
        "z": "21ce826.2ff977e",
        "name": "decision tree classifier predictor",
        "modelPath": "test/models",
        "modelName": "dtc.b",
        "x": 550,
        "y": 420,
        "wires": [
            [
                "b8f2ab19.e693a8"
            ],
            [
                "f7c59de2.be773"
            ]
        ]
    },
    {
        "id": "26110acb.cbf526",
        "type": "decision tree classifier",
        "z": "21ce826.2ff977e",
        "name": "decision tree classifier trainer",
        "savePath": "test/models",
        "saveName": "dtc.b",
        "maxDepth": "",
        "criterion": "gini",
        "splitter": "best",
        "x": 540,
        "y": 200,
        "wires": [
    [
      "2270c854.c34e08"
    ],
    [
      "86385870.9f6b88"
    ]
  ]
},
{
  "id": "b8f2ab19.e693a8",
  "type": "assessment",
  "z": "21ce826.2ff977e",
  "name": "",
  "score": "accuracy_score",
  "x": 590,
  "y": 360,
  "wires": [
    [
      "808a0c93.8ee38"
    ],
    [
      "f7c59de2.be773"
    ]
  ]
},
{
  "id": "f1b47338.aab82",
  "type": "load dataset",
  "z": "21ce826.2ff977e",
  "name": "",
  "datasetFolder": "test/datasets",
  "datasetName": "iris",
  "partition": "test.csv",
  "input": false,
  "output": true,
  "x": 290,
  "y": 360,
  "wires": [
    [
      "b8f2ab19.e693a8"
    ],
    [
      "f7c59de2.be773"
    ]
  ]
},
{
  "id": "1ea9f445.89d0bc",
  "type": "load dataset",
  "z": "21ce826.2ff977e",
  "name": "",
  "datasetFolder": "test/datasets",
  "datasetName": "iris",
  "partition": "test.csv",
  "input": true,
        "output": false,
        "x": 290,
        "y": 480,
        "wires": [
            [
                "e69a3271.c7cab"
            ],
            [
                "f7c59de2.be773"
            ]
        ]
    },
    {
        "id": "f7c59de2.be773",
        "type": "debug",
        "z": "21ce826.2ff977e",
        "name": "error",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "x": 790,
        "y": 480,
        "wires": []
    },
    {
        "id": "808a0c93.8ee38",
        "type": "debug",
        "z": "21ce826.2ff977e",
        "name": "print",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "x": 790,
        "y": 360,
        "wires": []
    },
    {
        "id": "8a4ea95c.f860b8",
        "type": "predictor",
        "z": "21ce826.2ff977e",
        "name": "decision tree classifier predictor",
        "modelPath": "test/models",
        "modelName": "dtc.b",
        "x": 450,
        "y": 580,
        "wires": [
            [
                "e967043f.480868"
            ],
            [
                "e66df10b.40ba8"
        ]
    ]
},
{
    "id": "e967043f.480868",
    "type": "mqtt out",
    "z": "21ce826.2ff977e",
    "name": "",
    "topic": "predictions",
    "qos": "",
    "retain": "",
    "broker": "cb216faf.d9136",
    "x": 730,
    "y": 540,
    "wires": []
},
{
    "id": "e66df10b.40ba8",
    "type": "debug",
    "z": "21ce826.2ff977e",
    "name": "error",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "x": 710,
    "y": 620,
    "wires": []
},
{
    "id": "3cd1a442.2bc73c",
    "type": "mqtt in",
    "z": "21ce826.2ff977e",
    "name": "",
    "topic": "classification",
    "qos": "2",
    "broker": "cb216faf.d9136",
    "x": 140,
    "y": 580,
    "wires": [
        [
            "8a4ea95c.f860b8"
        ]
    ]
},
{
    "id": "cb216faf.d9136",
    "type": "mqtt-broker",
    "z": "",
    "name": "",
    "broker": "iot.eclipse.org",
    "port": "1883",
    "clientid": "",
    "usetls": false,
    "compatmode": true,
    "keepalive": "60",
    "cleansession": true,
    "willTopic": "",
    "willQos": "0",
    "willPayload": "",
    "birthTopic": "",
    "birthQos": "0",
    "birthPayload": ""
  }
]
```