CDW Documentation

Project Summary

Project Summary

Model Name

Loan Default Predictor

Purpose

Predict whether a loan applicant is likely to default on a loan based on personal, financial, and credit attributes.

Model Type

Classification

Algorithm Source

AutoML (Azure Machine Learning)

—

What the Model Does

Given applicant data, the model predicts:

- **0** → Will likely default on loan
- **1** → Will likely not default on loan

Azure AutoML trained and selected the best performing algorithm based on metrics like AUC-weighted and accuracy.

—

Steps Taken

1. Create Dataset in Azure ML

Used the Azure ML Studio UI to upload a CSV dataset (e.g., `loan_applicants.csv`), containing labeled data with these columns:

```
["ApplicantID", "Age", "MaritalStatus", "Dependents", "Education",
```

```
 "AnnualIncome", "EmploymentType", "JobSector", "CreditScore",
 "LoanAmount", "LoanTermMonths", "LoanPurpose", "InterestRate",
 "PastDefaults", "LatePayments", "BankCustomerYears", "OwnsHome",
"LoanDefault"]
```

Where `LoanDefault` is the target label (0 or 1).

2. Train with AutoML via GUI

Used the Azure ML Studio UI to:

- Start a new AutoML classification run
- Select `LoanDefault` as the target column
- Choose compute cluster and enable explanation and metrics logging
- Let AutoML train multiple models and select the best one

3. Register the Model

After training:

- Selected the best run
- Clicked **Register Model**
- Model is now available in the workspace under **Assets > Models**

4. Deploy the Model

Deployed the model from Azure ML Studio:

- Clicked **Deploy > Real-time endpoint**
- Selected compute target
- Endpoint URL was generated and secured with an API key

5. Test the REST Endpoint

Access keys from Key Vault (Python):

```
api_key = secret_client.get_secret("max-ml-key").value
url = secret_client.get_secret("max-ml-endpoint").value

if not api_key or not url:
    raise Exception("Missing key or URL from Key Vault")
```

Sample Prediction Request:

```
import urllib.request
import json
```

```
data = {
  "input_data": {
    "columns": [
      "Age", "MaritalStatus", "Dependents", "Education", "AnnualIncome",
      "EmploymentType", "JobSector", "CreditScore", "LoanAmount",
      "LoanTermMonths", "LoanPurpose", "InterestRate", "PastDefaults",
      "LatePayments", "BankCustomerYears", "OwnsHome"
    ],
    "index": [0],
    "data": [[35, "Married", 2, "Graduate", 55000, "Full-time", "Private",
710,
              25000, 60, "Home", 5.2, 0, 1, 6, "Yes"]]
  }
}

body = json.dumps(data).encode("utf-8")
headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {api_key}"
}

req = urllib.request.Request(url, body, headers)

try:
    with urllib.request.urlopen(req) as response:
        decoded = response.read().decode("utf-8")
        prediction = json.loads(decoded)[0]

        if prediction == 0:
            print("Prediction: Will likely default on loan")
        elif prediction == 1:
            print("Prediction: Will likely not default on loan")
        else:
            print(f"Unexpected prediction value: {prediction}")

except urllib.error.HTTPError as error:
    print("HTTP error:", error.code)
    print(error.read().decode())
```

—

Input Format Expected by Endpoint

```
{
  "input_data": {
    "columns": [
      "Age", "MaritalStatus", "Dependents", "Education", "AnnualIncome",
      "EmploymentType", "JobSector", "CreditScore", "LoanAmount",
      "LoanTermMonths", "LoanPurpose", "InterestRate", "PastDefaults",
      "LatePayments", "BankCustomerYears", "OwnsHome"
```

```
    ],
    "index": [0],
    "data": [[35, "Married", 2, "Graduate", 55000, "Full-time", "Private",
710,
              25000, 60, "Home", 5.2, 0, 1, 6, "Yes"]]
  }
}
```

—

Outputs

The endpoint returns either:

- [0] or [1]

Mapped in your script to readable text:

- **0** → “Will likely default on loan”
- **1** → “Will likely not default on loan”