pip show azure-ai-ml

Name: azure-ai-ml  
Version: 1.26.0  
Summary: Microsoft Azure Machine Learning Client Library for Python  
Home-page: https://github.com/Azure/azure-sdk-for-python  
Author: Microsoft Corporation  
Author-email: azuresdkengsysadmins@microsoft.com  
License: MIT License  
Location: /anaconda/envs/azureml\_py38/lib/python3.10/site-packages  
Requires: azure-common, azure-core, azure-mgmt-core, azure-monitor-opentelemetry, azure-storage-blob, azure-storage-file-datalake, azure-storage-file-share, colorama, isodate, jsonschema, marshmallow, msrest, pydash, pyjwt, pyyaml, strictyaml, tqdm, typing-extensions  
Required-by:   
Note: you may need to restart the kernel to use updated packages.

Connect to your workspace With the required SDK packages installed, now you're ready to connect to your workspace.

To connect to a workspace, we need identifier parameters - a subscription ID, resource group name, and workspace name. Since you're working with a compute instance, managed by Azure Machine Learning, you can use the default values to connect to the workspace.

from azure.identity import DefaultAzureCredential, InteractiveBrowserCredential  
from azure.ai.ml import MLClient  
  
try:  
 credential = DefaultAzureCredential()  
 # Check if given credential can get token successfully.  
 credential.get\_token("https://management.azure.com/.default")  
except Exception as ex:  
 # Fall back to InteractiveBrowserCredential in case DefaultAzureCredential not work  
 credential = InteractiveBrowserCredential()

# Get a handle to workspace  
ml\_client = MLClient.from\_config(credential=credential)

Found the config file in: /config.json

Use the Python SDK to train a model To train a model, you'll first create the diabetes\_training.py script in the src folder. The script uses the diabetes.csv file in the same folder as the training data.

## Creating a Linear regression model python file

%%writefile src/expenses-training-linear-regression.py  
  
# import libraries  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import roc\_auc\_score  
from sklearn.metrics import roc\_curve  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
  
  
# load the diabetes dataset  
print("Loading Data...")  
expenses = pd.read\_csv('azureml://subscriptions/56fbbfce-6dab-4d62-af16-ccd107f4d9d3/resourcegroups/DEV-EA-EASTUS-DF-RG/workspaces/mlw-uel-cn-7000/datastores/ds\_healthdata\_2/paths/data/ons\_nhs\_expenditure.csv')  
  
print(expenses)  
  
# separate features and labels  
X, y = expenses[['Year']].values, expenses['Total\_current\_healthcare\_expenditure'].values  
  
# set regularization hyperparameter  
reg = 0.01  
  
# split data into training set and test set  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=0)  
  
# train a linear regression model  
print('Training a linear regression model with regularization rate of', reg)  
model = LinearRegression().fit(X\_train, y\_train)  
  
# calculate accuracy  
y\_hat = model.predict(X\_test)  
acc = model.score(X\_test, y\_test)  
print('R^2 Score:', acc)  
  
# calculate AUC (not applicable for regression, but you can calculate RMSE or MAE)  
from sklearn.metrics import mean\_squared\_error  
rmse = np.sqrt(mean\_squared\_error(y\_test, y\_hat))  
print('RMSE:', rmse)  
  
# continuous labels to discrete classes, you can use the following approach, Classification

Overwriting src/expenses-training-linear-regression.py

## Creating a logistic regression model python file

%%writefile src/expenses-training-LogisticRegression-classification.py  
  
# import libraries  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import roc\_auc\_score  
from sklearn.metrics import roc\_curve  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
  
  
# load the diabetes dataset  
print("Loading Data...")  
expenses = pd.read\_csv('azureml://subscriptions/56fbbfce-6dab-4d62-af16-ccd107f4d9d3/resourcegroups/DEV-EA-EASTUS-DF-RG/workspaces/mlw-uel-cn-7000/datastores/ds\_healthdata\_2/paths/data/ons\_nhs\_expenditure.csv')  
  
print(expenses)  
  
# separate features and labels  
X = expenses[['Year']].values  
y = pd.cut(expenses['Total\_current\_healthcare\_expenditure'], bins=3, labels=[0, 1, 2])  
  
# set regularization hyperparameter  
reg = 0.01  
  
# split data into training set and test set with stratification  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=0, stratify=y)  
  
# train a logistic regression model  
print('Training a logistic regression model with regularization rate of', reg)  
model = LogisticRegression(C=1/reg, solver="liblinear").fit(X\_train, y\_train)  
  
# calculate accuracy  
y\_hat = model.predict(X\_test)  
acc = np.average(y\_hat == y\_test)  
print('Accuracy:', acc)  
  
# calculate AUC  
y\_scores = model.predict\_proba(X\_test)  
auc = roc\_auc\_score(y\_test, y\_scores, multi\_class='ovr', labels=[0, 1, 2])  
print('AUC: ' + str(auc))

Overwriting src/expenses-training-LogisticRegression-classification.py

Run the cell below to submit the job that trains a classification model to predict diabetes.

## Classification Modeling Run using Logistic regression

from azure.ai.ml import MLClient, command, Input  
from azureml.core import Workspace, Environment  
from azure.identity import DefaultAzureCredential  
import fsspec  
from azureml.core import Workspace  
  
from azureml.core.conda\_dependencies import CondaDependencies  
  
# Load the workspace from the saved config file  
ws = Workspace.from\_config()  
  
# Retrieve the existing curated environment  
curated\_env = Environment.get(workspace=ws, name="AzureML-sklearn-0.24-ubuntu18.04-py37-cpu")  
  
# Create a copy of the curated environment  
env = curated\_env.clone("my-custom-sklearn-env")  
  
  
  
# Create a CondaDependencies object if it doesn't exist  
if env.python.conda\_dependencies is None:  
 env.python.conda\_dependencies = CondaDependencies()  
  
# Add fsspec to the environment  
env.python.conda\_dependencies.add\_pip\_package("fsspec")  
  
# Set the name and version on the environment object  
env.name = "my-custom-sklearn-env"  
env.version = "1"  
  
# Register the updated environment with a version  
env.register(workspace=ws)  
  
# configure job  
job = command(  
 code="./src",  
 command="python expenses-training-LogisticRegression-classification.py",  
 environment="my-custom-sklearn-env:1",  
 compute="amlclcn7000",  
 display\_name="expense-pythonv2-train-log\_reg",  
 experiment\_name="expense-training-log-reg"  
)  
  
# submit job  
returned\_job = ml\_client.create\_or\_update(job)  
aml\_url = returned\_job.studio\_url  
print("Monitor your job at", aml\_url)

Environment version is set. Attempting to register desired version. To auto-version, reset version to None.  
Class AutoDeleteSettingSchema: This is an experimental class, and may change at any time. Please see https://aka.ms/azuremlexperimental for more information.  
Class AutoDeleteConditionSchema: This is an experimental class, and may change at any time. Please see https://aka.ms/azuremlexperimental for more information.  
Class BaseAutoDeleteSettingSchema: This is an experimental class, and may change at any time. Please see https://aka.ms/azuremlexperimental for more information.  
Class IntellectualPropertySchema: This is an experimental class, and may change at any time. Please see https://aka.ms/azuremlexperimental for more information.  
Class ProtectionLevelSchema: This is an experimental class, and may change at any time. Please see https://aka.ms/azuremlexperimental for more information.  
Class BaseIntellectualPropertySchema: This is an experimental class, and may change at any time. Please see https://aka.ms/azuremlexperimental for more information.

Monitor your job at https://ml.azure.com/runs/happy\_star\_31d7kdr556?wsid=/subscriptions/56fbbfce-6dab-4d62-af16-ccd107f4d9d3/resourcegroups/dev-ea-eastus-df-rg/workspaces/mlw-uel-cn-7000&tid=9258a771-d6bc-4486-adea-e939c450d791

## Linear regression

# configure job  
job = command(  
 code="./src",  
 command="python expenses-training-linear-regression.py",  
 environment="my-custom-sklearn-env:1",  
 compute="amlclcn7000",  
 display\_name="expense-pythonv2-train-lin-reg",  
 experiment\_name="expense-training-lin-reg"  
)  
  
# submit job  
returned\_job = ml\_client.create\_or\_update(job)  
aml\_url = returned\_job.studio\_url  
print("Monitor your job at", aml\_url)

Monitor your job at https://ml.azure.com/runs/jovial\_fork\_d08mtwz2dy?wsid=/subscriptions/56fbbfce-6dab-4d62-af16-ccd107f4d9d3/resourcegroups/dev-ea-eastus-df-rg/workspaces/mlw-uel-cn-7000&tid=9258a771-d6bc-4486-adea-e939c450d791