Leveraging MindSpore-Based Machine and Deep Learning Models for Autism Classification on Structured and Image Data

Suggested classes: 3 classes

Class 1:

Class 1: Detailed description on Machine/Deep Learning Models for Structured .csv and Image datasets

Class 2: Introduction to MindSpore, Packages and Development Environment

Class 3: Hands-on Practice for students with sample datasets.

Target Audience: Master of Science in Artificial Intelligence (Semester 2)/Bachelor of Science in Data Analytics (Year 3)/ Bachelor in ICT (Year 3)

Huawei Course Material used: Artificial Intelligence Technology and Applications

I. Tasks

1. Task Description

Machine learning (ML), deep learning, automated machine learning, fine-tuning and optimization are different facets of Artificial Intelligence (AI), focusing on imbuing intelligence into machines, thereby turning them into smart systems that can assist and enhance human decision-making and output with reduced human exhaustion and errors. MindSpore is an open-source, complete AI computing framework that provides supervised and unsupervised learning models on diverse datasets. Moreover, it includes powerful packages for image processing using MindCV that can be executed across different platforms (CPU, GPU, Ascend). Parallelism for quicker execution on big datasets can also be seamlessly done using the CANN architecture with the possibility of running the ML models on cloud using ModelArts Cloud Platform.

Given the multifarious possibilities with MindSpore, this experiment covers the entire process, from dataset preparation and processing using structured datasets in Comma Separated Version (.csv) files and image datasets using MindCV, to model fine-tuning, training and testing. Every course piques student engagement when they can see the impact of their learning and efforts on real-world data. This course targets one of the leading causes of concern in health care today, autism. This is one medical challenge that despite intense research is still a mystery yet to be unraveled. Data is available but the means to make that data meaningful to predict, identify causes, suggest therapy using technology is demonstrated through this experiment.

On completion of this task, students will have gained an in-depth understanding of the working principles of machine learning on structured data and deep learning model on image data, mastered the art of visualizing model performance, and recognize how machine learning influences decision making in other disciplines of study. They will be able to identify how the Mindspore platform can further be extended to run ML models by integrating MindSpore with ModelArts platform for a no-code training and deployment using ExeML.

2. Objectives

After completing this task, you will be able to:

(1) Recognize the importance of data preparation for learning models and the impact of data on model performance.

(2) Use MindSpore and related packages and know how to configure the development environment with MiniConda.

(3) Label, pre-process, and build models on both structured and image datasets.

(4) Understand the role of evaluation metrics and methods in ranking ML models.

(5) Design and implement effective data processing pipelines.

(6) Optimize hyperparameters.

(7) Visualize the performance of ML models.

(8) Understand the complete workflow of MindSpore learning Framework.

II. Task Preparation

1. Background Knowledge

This experiment requires basic knowledge on programming with Python, machine learning/deep learning pipeline, evaluation methods and metrics. Fundamental knowledge of image processing would be an added advantage to complete the experiment in 3 classes.

(1) Python Programming

This includes the course material on Python Programming Basics – Covering Section 2 in the Huawei course material.

(2) Machine learning/Deep Learning Pipeline

The course material on Machine Learning Overview (Section 3: 3.1 and 3.3) and Deep Learning Overview (Section 4:4.1 to 4.5) in the Huawei ICT Academy tutorial trains the students on in-depth knowledge of the machine learning pipeline. Section 3.2.2 also covers the introduction and explanation of hyper-parameters, their role and process of tuning (using search methods) and optimization for enhancing model performance.

(3) Evaluation Methods and Metrics

The model errors and performance are covered in detail in Section 3.2 (Machine Learning Process). The complexity and errors of models are neatly described using multiple metrics – MAE, MSE, Precision, Recall, F1-Score based on the Confusion Matrix along with Cross-validation as an evaluation method.

(4) Image Processing Fundamentals

Image processing basics are included in Section 4 under Deep learning Overview for Neural Network types (Section 4.6) in the Huawei course material. This focusses on the basics of images – colors, channels, filters, size, scaling, and normalization.

(5) Software Platforms and Packages

*The **software platform** used for autism classification is*

Miniconda with Python 3.11

Mindspore 2.7.1

CPU – Device Deployment

Jupyter Notebook

Packages installed:

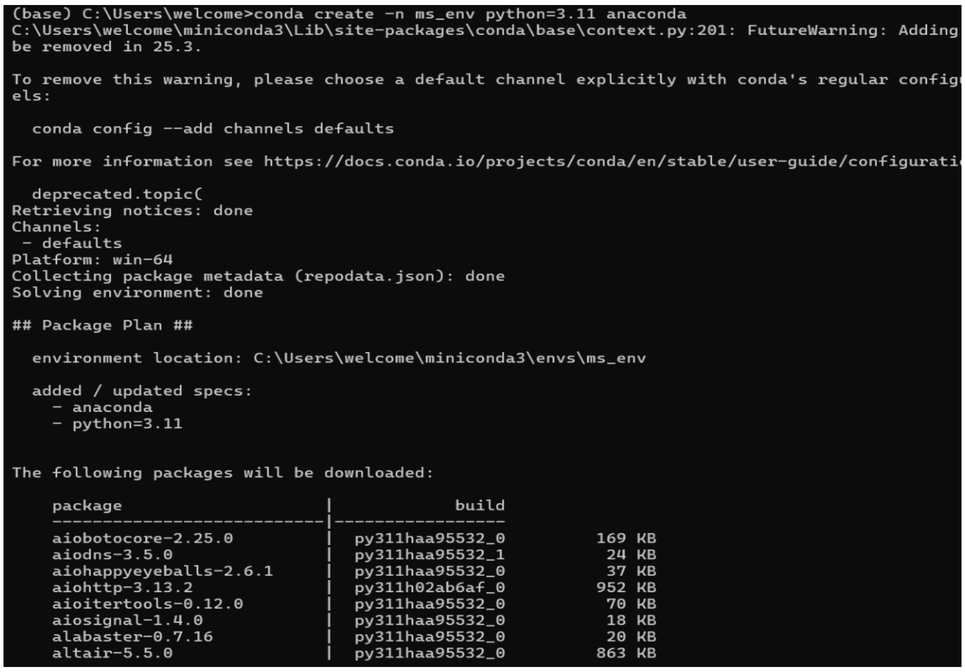
Pandas, Scikit-Learn, MindCV, Matplotlib and other packages as portrayed *in the screenshots in the next section.*

2. Experiment Environment Preparation

This experiment requires the initial setup of Python environment with MiniConda and Mindspore computing platform. Compatibility and dependencies are to be met to ensure that the execution is conflict-free. Mindspore is a one-stop AI development platform provided by Huawei Cloud. It provides powerful computing resources and convenient development tools for training and deploying deep learning models. Preparation of the experiment environment includes download and installation of Python version (Miniconda) compatible with Mindspore AI development framework 2.7.1, and configuring jupyter notebook environment for a graphical user interface. Since the data to be used are not in-built, they need to be organized in the appropriate folder structure for MindSpore to access and read. The set up ensures that data upload, data preparation, model training and testing, optimization and visualization of results takes place in a seamless fashion.

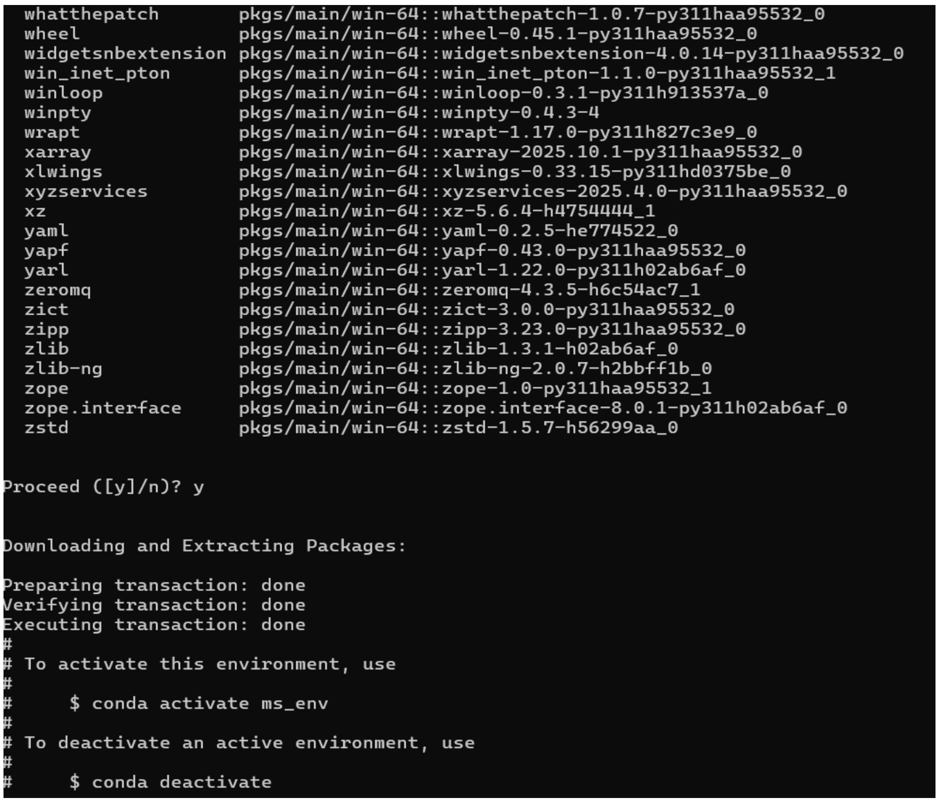
(1) Basic environment configuration:

This experiment is based on the Mindspore AI Deep Learning Platform. The environment requirements are uploaded as Environment Setup.docx at the Github repository: ***https://github.com/gracia2026/Autism-classificatio-with-MindSpore.git***



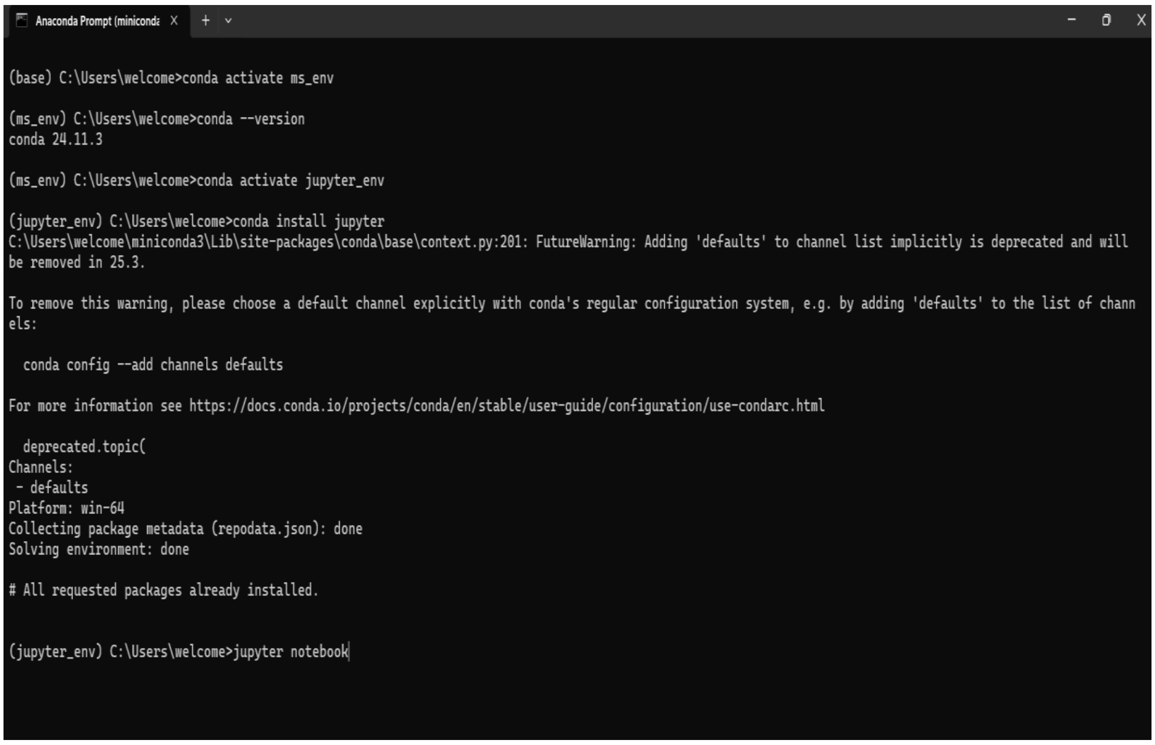
```
(base) C:\Users\welcome>conda create -n ms_env python=3.11 anaconda
C:\Users\welcome\miniconda3\Lib\site-packages\conda\base\context.py:201: FutureWarning: Adding
be removed in 25.3.

To remove this warning, please choose a default channel explicitly with conda's regular config
els:

  conda config --add channels defaults

For more information see https://docs.conda.io/projects/conda/en/stable/user-guide/configurati

  deprecated.topic(
Retrieving notices: done
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\welcome\miniconda3\envs\ms_env

  added / updated specs:
    - anaconda
    - python=3.11


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    aiobotocore-2.25.0         |  py311haa95532_0         169 KB
    aiodns-3.5.0               |  py311haa95532_1          24 KB
    aiohappyeyeballs-2.6.1     |  py311haa95532_0          37 KB
    aiohttp-3.13.2             |  py311h02ab6af_0         952 KB
    aioitertools-0.12.0        |  py311haa95532_0          70 KB
    aiosignal-1.4.0            |  py311haa95532_0          18 KB
    alabaster-0.7.16           |  py311haa95532_0          20 KB
    altair-5.5.0               |  py311haa95532_0         863 KB
```

Figure 1. Installation of Python 3.11



```
  whatthepatch        pkgs/main/win-64::whatthepatch-1.0.7-py311haa95532_0
  wheel               pkgs/main/win-64::wheel-0.45.1-py311haa95532_0
  widgetsnbextension  pkgs/main/win-64::widgetsnbextension-4.0.14-py311haa95532_0
  win_inet_pton       pkgs/main/win-64::win_inet_pton-1.1.0-py311haa95532_1
  winloop             pkgs/main/win-64::winloop-0.3.1-py311h913537a_0
  winpty              pkgs/main/win-64::winpty-0.4.3-4
  wrapt               pkgs/main/win-64::wrapt-1.17.0-py311h827c3e9_0
  xarray              pkgs/main/win-64::xarray-2025.10.1-py311haa95532_0
  xlwings             pkgs/main/win-64::xlwings-0.33.15-py311hd0375be_0
  xyzservices         pkgs/main/win-64::xyzservices-2025.4.0-py311haa95532_0
  xz                  pkgs/main/win-64::xz-5.6.4-h4754444_1
  yaml                pkgs/main/win-64::yaml-0.2.5-he774522_0
  yapf                pkgs/main/win-64::yapf-0.43.0-py311haa95532_0
  yarl                pkgs/main/win-64::yarl-1.22.0-py311h02ab6af_0
  zeromq              pkgs/main/win-64::zeromq-4.3.5-h6c54ac7_1
  zict                pkgs/main/win-64::zict-3.0.0-py311haa95532_0
  zipp                pkgs/main/win-64::zipp-3.23.0-py311haa95532_0
  zlib                pkgs/main/win-64::zlib-1.3.1-h02ab6af_0
  zlib-ng             pkgs/main/win-64::zlib-ng-2.0.7-h2bbff1b_0
  zope                pkgs/main/win-64::zope-1.0-py311haa95532_1
  zope.interface      pkgs/main/win-64::zope.interface-8.0.1-py311h02ab6af_0
  zstd                pkgs/main/win-64::zstd-1.5.7-h56299aa_0


Proceed ([y]/n)? y


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate ms_env
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

Figure 2. Activation of Conda Environment for Installing Packages

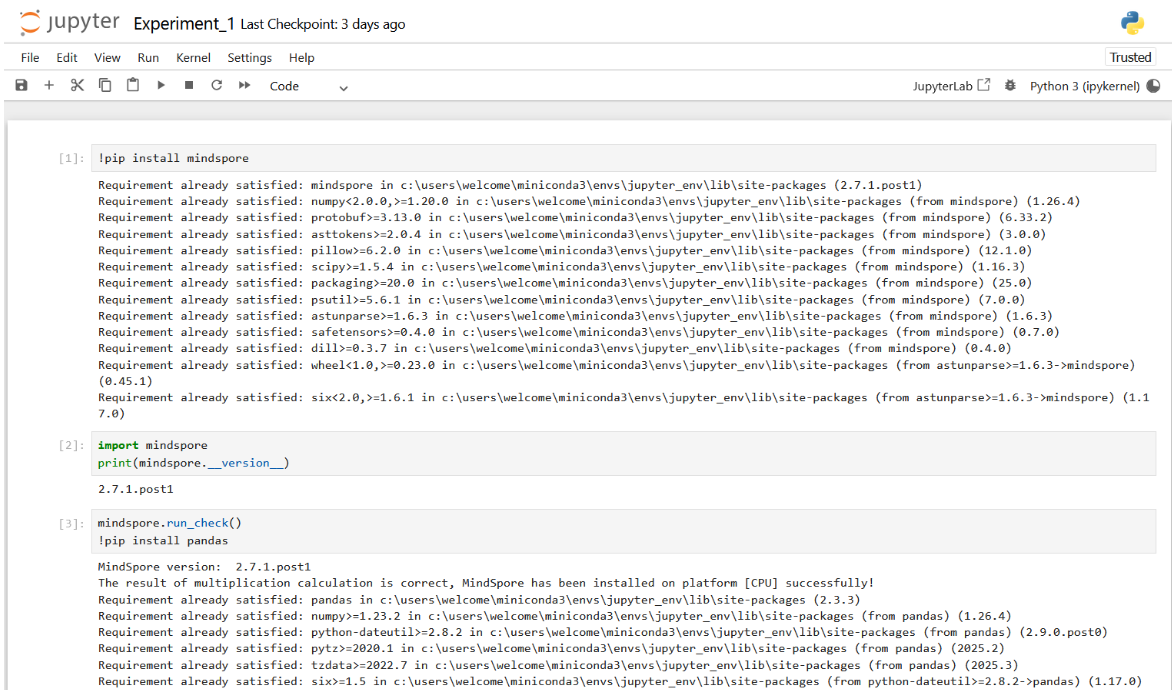


```
(base) C:\Users\welcome>conda activate ms_env

(ms_env) C:\Users\welcome>conda --version
conda 24.11.3

(ms_env) C:\Users\welcome>conda activate jupyter_env

(jupyter_env) C:\Users\welcome>conda install jupyter
C:\Users\welcome\miniconda3\Lib\site-packages\conda\base\context.py:201: FutureWarning: Adding 'defaults' to channel list implicitly is deprecated and will
be removed in 25.3.

To remove this warning, please choose a default channel explicitly with conda's regular configuration system, e.g. by adding 'defaults' to the list of chann
els:

  conda config --add channels defaults

For more information see https://docs.conda.io/projects/conda/en/stable/user-guide/configuration/use-condarc.html

  deprecated.topic(
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

# All requested packages already installed.


(jupyter_env) C:\Users\welcome>jupyter notebook
```

Figure 3. Installation of Jupyter Notebook

(2) Software libraries:

The main software packages to be installed in this experiment are:

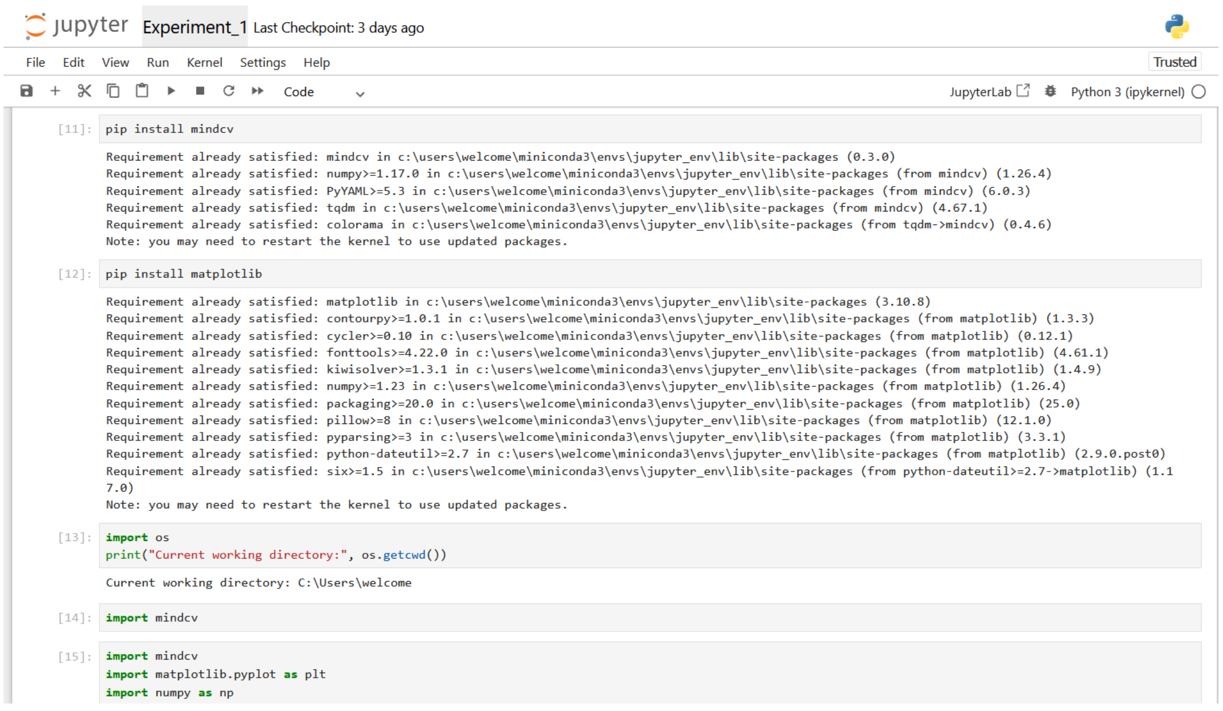

jupyter Experiment_1 Last Checkpoint: 3 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[1]: !pip install mindspore

Requirement already satisfied: mindspore in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (2.7.1.post1)
Requirement already satisfied: numpy<2.0.0,>=1.20.0 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (1.26.4)
Requirement already satisfied: protobuf>=3.13.0 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (6.33.2)
Requirement already satisfied: asttokens>=2.0.4 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (3.0.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (12.1.0)
Requirement already satisfied: scipy>=1.5.4 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (1.16.3)
Requirement already satisfied: packaging>=20.0 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (25.0)
Requirement already satisfied: psutil>=5.6.1 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (7.0.0)
Requirement already satisfied: astunparse>=1.6.3 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (1.6.3)
Requirement already satisfied: safetensors>=0.4.0 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (0.7.0)
Requirement already satisfied: dill>=0.3.7 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindspore) (0.4.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from astunparse>=1.6.3->mindspore)
(0.45.1)
Requirement already satisfied: six<2.0,>=1.6.1 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from astunparse>=1.6.3->mindspore) (1.1
7.0)

[2]: import mindspore
     print(mindspore.__version__)

2.7.1.post1

[3]: mindspore.run_check()
     !pip install pandas

MindSpore version:  2.7.1.post1
The result of multiplication calculation is correct, MindSpore has been installed on platform [CPU] successfully!
Requirement already satisfied: pandas in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (2.3.3)
Requirement already satisfied: numpy>=1.23.2 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from pandas) (2025.3)
Requirement already satisfied: six>=1.5 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

Figure 4. Installation and Activation of Mindspore and Pandas

```python
import pandas as pd
import numpy as np
import mindspore as ms
from mindspore import nn, ops, Tensor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

Figure 5. Installation of Packages for Classification of .csv Structured Data

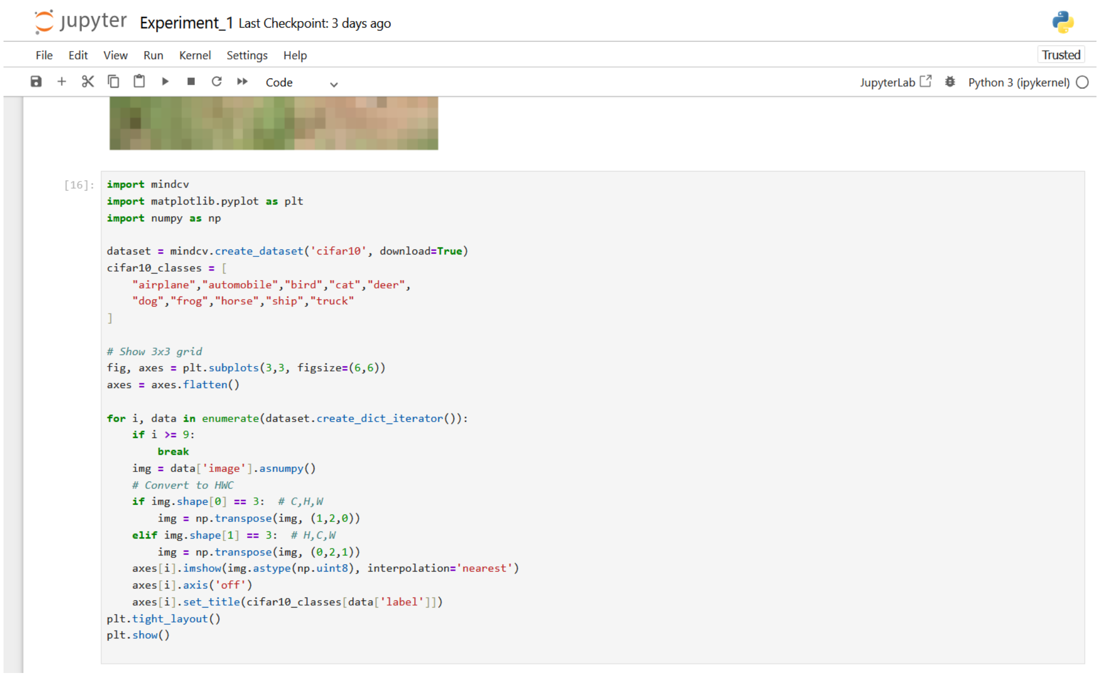

jupyter Experiment_1 Last Checkpoint: 3 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[11]: pip install mindcv

Requirement already satisfied: mindcv in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (0.3.0)
Requirement already satisfied: numpy>=1.17.0 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindcv) (1.26.4)
Requirement already satisfied: PyYAML>=5.3 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindcv) (6.0.3)
Requirement already satisfied: tqdm in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from mindcv) (4.67.1)
Requirement already satisfied: colorama in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from tqdm->mindcv) (0.4.6)
Note: you may need to restart the kernel to use updated packages.

[12]: pip install matplotlib

Requirement already satisfied: matplotlib in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (3.10.8)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from matplotlib) (12.1.0)
Requirement already satisfied: pyparsing>=3 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from matplotlib) (3.3.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\welcome\miniconda3\envs\jupyter_env\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.1
7.0)
Note: you may need to restart the kernel to use updated packages.

[13]: import os
      print("Current working directory:", os.getcwd())

Current working directory: C:\Users\welcome

[14]: import mindcv

[15]: import mindcv
      import matplotlib.pyplot as plt
      import numpy as np
```

Figure 6. Installation of Packages for Image Classification

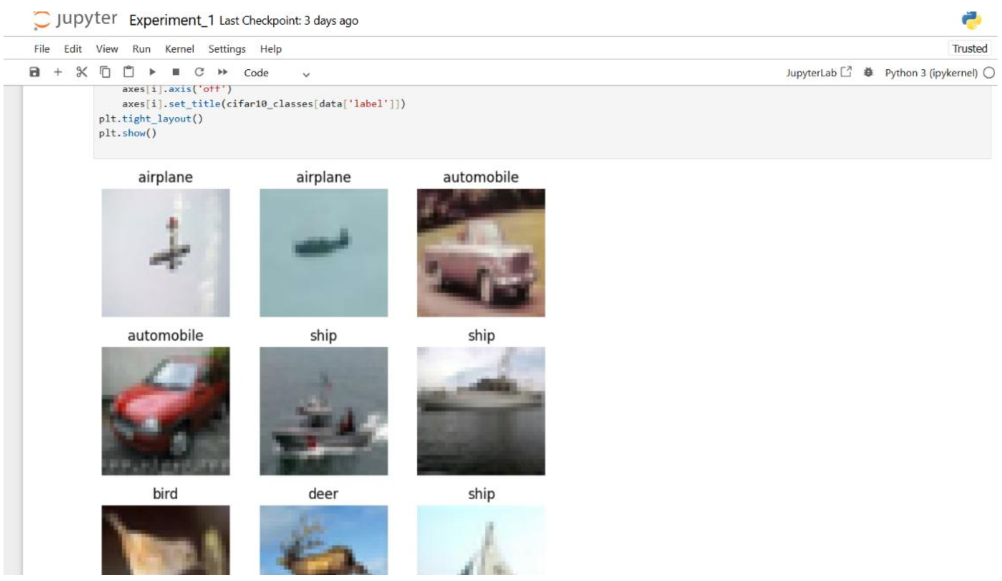
(3) Preparing the MindSpore Development Platform for Image Processing:

The following code shows the packages to be installed and verified for the successful implementation of this experiment in jupyter notebook.


jupyter Experiment_1 Last Checkpoint: 3 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

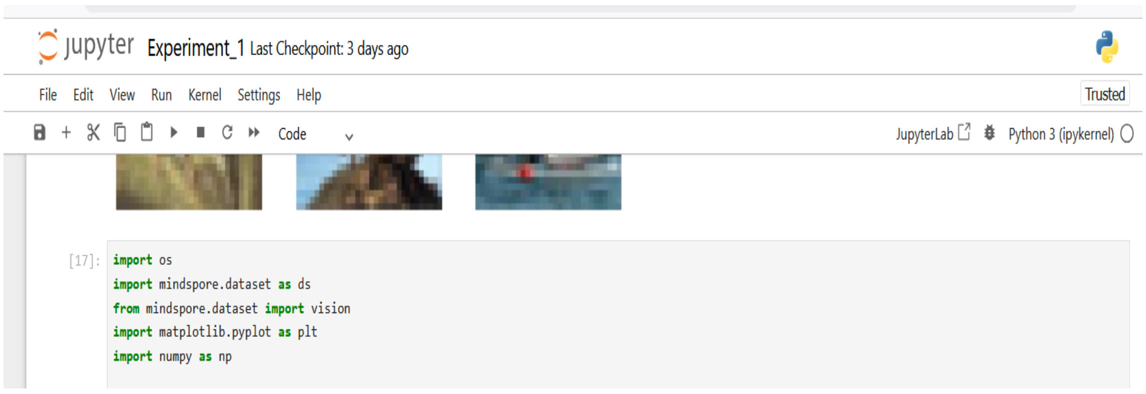
```
[16]: import mindcv
      import matplotlib.pyplot as plt
      import numpy as np

      dataset = mindcv.create_dataset('cifar10', download=True)
      cifar10_classes = [
          "airplane","automobile","bird","cat","deer",
          "dog","frog","horse","ship","truck"
      ]

      # Show 3x3 grid
      fig, axes = plt.subplots(3,3, figsize=(6,6))
      axes = axes.flatten()

      for i, data in enumerate(dataset.create_dict_iterator()):
          if i >= 9:
              break
          img = data['image'].asnumpy()
          # Convert to HWC
          if img.shape[0] == 3:  # C,H,W
              img = np.transpose(img, (1,2,0))
          elif img.shape[1] == 3:  # H,C,W
              img = np.transpose(img, (0,2,1))
          axes[i].imshow(img.astype(np.uint8), interpolation='nearest')
          axes[i].axis('off')
          axes[i].set_title(cifar10_classes[data['label']])
      plt.tight_layout()
      plt.show()
```

Figure 7. Preparation of Mindspore Platform for Cifar-10 Dataset Display



```
        axes[i].axis('off')
        axes[i].set_title(cifar10_classes[data['label']])
    plt.tight_layout()
    plt.show()
```

Figure 8. Cifar-10 Dataset Display


Jupyter Experiment_1 Last Checkpoint: 3 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[17]: import os
      import mindspore.dataset as ds
      from mindspore.dataset import vision
      import matplotlib.pyplot as plt
      import numpy as np
```

Figure 9. Dataset and Vision Package for Stand-alone Image dataset Classification

```
# ===============================
# 1. IMPORTS
# ===============================
import os
import shutil
import numpy as np
from PIL import Image

import mindspore as ms
from mindspore import nn, Tensor

from mindcv import create_model

import mindspore.dataset.vision as vision
import mindspore.dataset.transforms as transforms

from sklearn.cluster import KMeans
```

Figure 10. Packages for Image Data Decoding, Transformation and Clustering

```python
# ============================================================
# MindSpore 2.7 - ResNet18 Transfer Learning (SAFE VERSION)
# ============================================================

import os
import numpy as np
import mindspore as ms
from mindspore import nn, Tensor
from mindspore.dataset import ImageFolderDataset, vision
from mindcv.models import resnet18

import matplotlib.pyplot as plt
from PIL import Image
from collections import defaultdict

ms.set_context(mode=ms.PYNATIVE_MODE)
```

Figure 11. Packages to Ensure Image Quality and Normalization

III. Task Implementation

1. Key Points

(1) Data Folder Organization and Access:

Folder Structure for Image Data Access, Classification and Clustering:

D:

- ➔ *data*
  - ○ *train*
    - ▪ *autistic*
    - ▪ *non_autistic*
  - ○ *val*
    - ▪ *autistic*
    - ▪ *non-autistic*

Data storage for .csv files:

- ➔ *Data*
  - ○ *Format data1.csv*

(2) Data Processing and Visualization: *To ensure the success of any learning model, the type-quality-size-count of the data needs to be verified. Poor quality, incomplete, unstructured data will require processing methods like:*

- *Removal of redundant/missing information*
- *Mean/Median/Mode for averaging the missing information*
- *Use of matplotlib and seaborn libraries for visualization*

This experiment includes the following data preparation for 2 types of data classification on:

Format data1.csv

- *Checking for missing data*
- *Label Encoding*

For classification and clustering on:

Autism Image Dataset

- *Image Decoding*
- *Image Transformation (Resizing, Scaling, Normalization)*

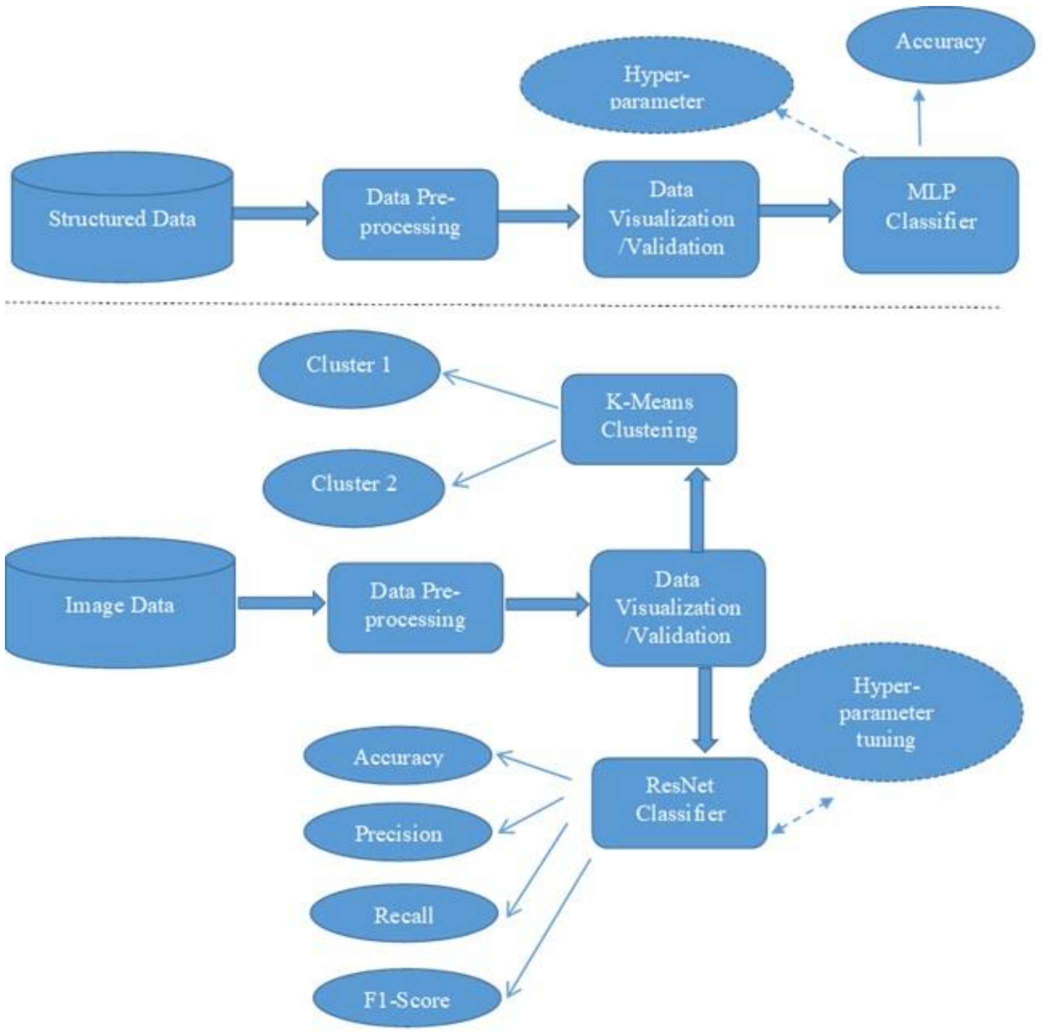


Figure 12. Mindspore – Based ML_DL framework for Autism Classification

(3) **Execution of ML/DL models on both datasets**: *This experiment follows the approach below to perform learning on the datasets:*

a) Set context and device target to CPU.

b) Structured .CSV data (Classification using Multi-Layer Perceptron)

i. Organize the data in folder with the correct path.

ii. Upload and visualize the data.

iii. Train-Test Evaluation Method

iv. Apply the Models – Optimize

v. Display the metrics

c) Autism Image data

Clustering

i. Organize the data in folders (train and validation) with the correct path.

ii. Upload and visualize the data.

iii. Decode, transform (Resize, Rescale, Normalize), verify image quality

iv. Apply K-Means clustering algorithm

v. Visualize the results

vi. Upload different images for validation and test again.

Classification

vii. Organize the data in folders (train and validation) with the correct path.

viii. Upload and visualize the data.

ix. Decode, transform (Resize, Rescale, Normalize), verify image quality

x. Apply ResNet learning algorithm

xi. Visualize the results

xii. Upload different images for validation and test again.

(4) **Hyper-parameter Optimization and Evaluation**: *Each of the above models has specific parameters that were tuned as follows. Explore and experiment with other parameters to study their impact on performance (time, efficiency)*

K-Means: number of clusters, random_state

ResNet: Batch size, epoch, loss_fn, optimizer, learning rate,

Multi-Layer Perceptron: Batch size, epoch, loss_fn, optimizer

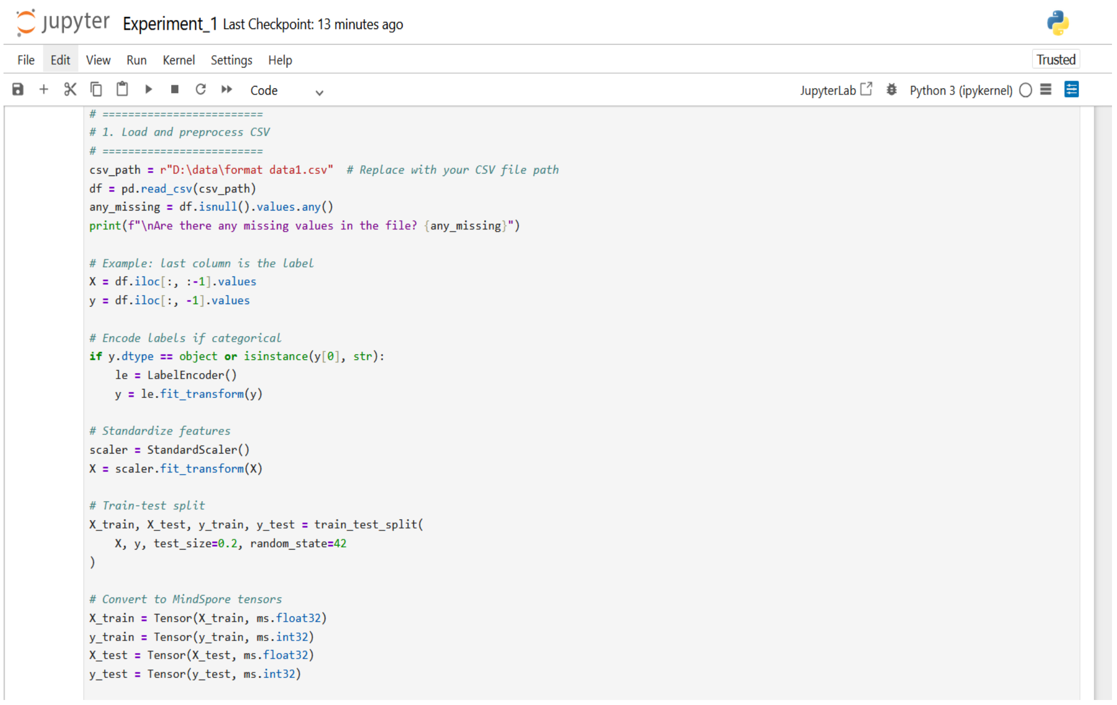
(5) Result Visualization:

Visualization of results that portray the different metrics used to evaluate the performance is an efficient way to observe the difference in speed of execution with the number of epochs, size and number of images. Potential decline in model performance is one of the key factors to develop and execute deep learning models on the cloud platforms for a low-code and faster execution.

2. Key Steps

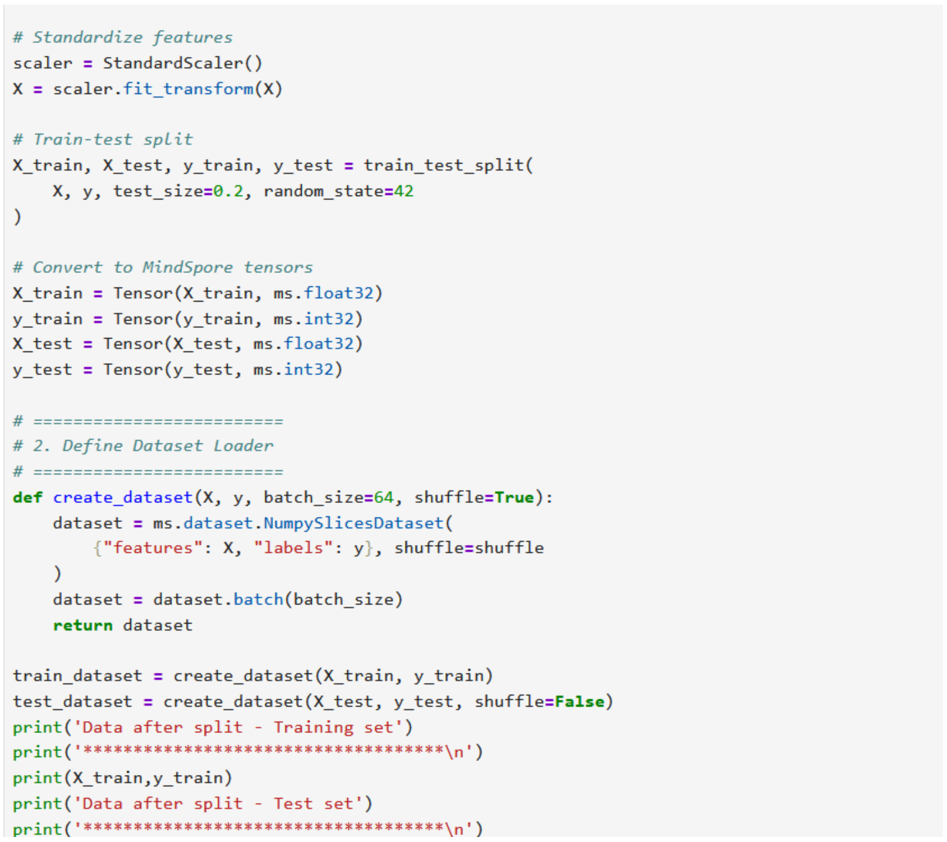
Step 1: Data Preparation and Visualization:

This experiment required Label encoding for running the Multilayer perceptron on the csv file. The train and test set for evaluation was set at 80% - 20%. Tensor is the basic data structure in mindspore and all data is converted to a standard format for easier mathematical processing.


jupyter Experiment_1 Last Checkpoint: 13 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

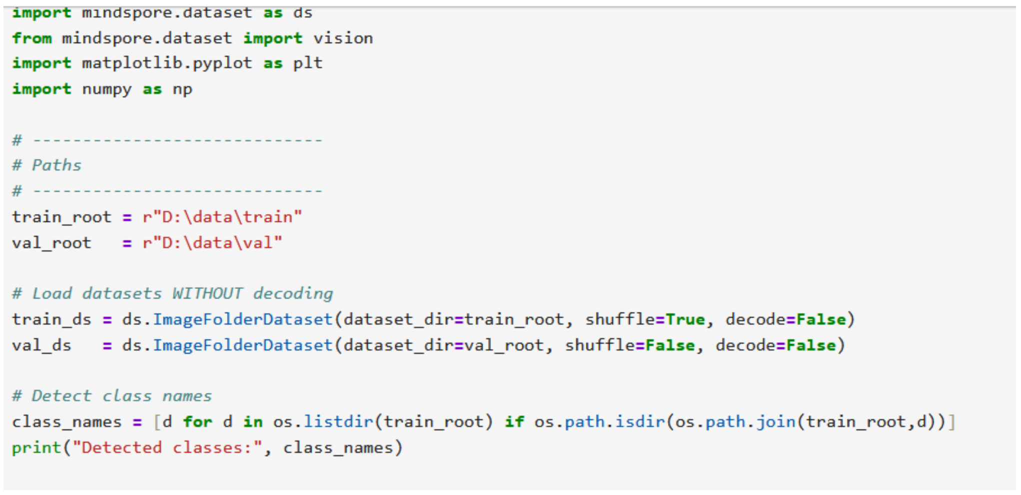
```python
# ========================
# 1. Load and preprocess CSV
# ========================
csv_path = r"D:\data\format data1.csv"  # Replace with your CSV file path
df = pd.read_csv(csv_path)
any_missing = df.isnull().values.any()
print(f"\nAre there any missing values in the file? {any_missing}")

# Example: last column is the label
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

# Encode labels if categorical
if y.dtype == object or isinstance(y[0], str):
    le = LabelEncoder()
    y = le.fit_transform(y)

# Standardize features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Convert to MindSpore tensors
X_train = Tensor(X_train, ms.float32)
y_train = Tensor(y_train, ms.int32)
X_test = Tensor(X_test, ms.float32)
y_test = Tensor(y_test, ms.int32)
```

Figure 13. Data upload and Pre-processing



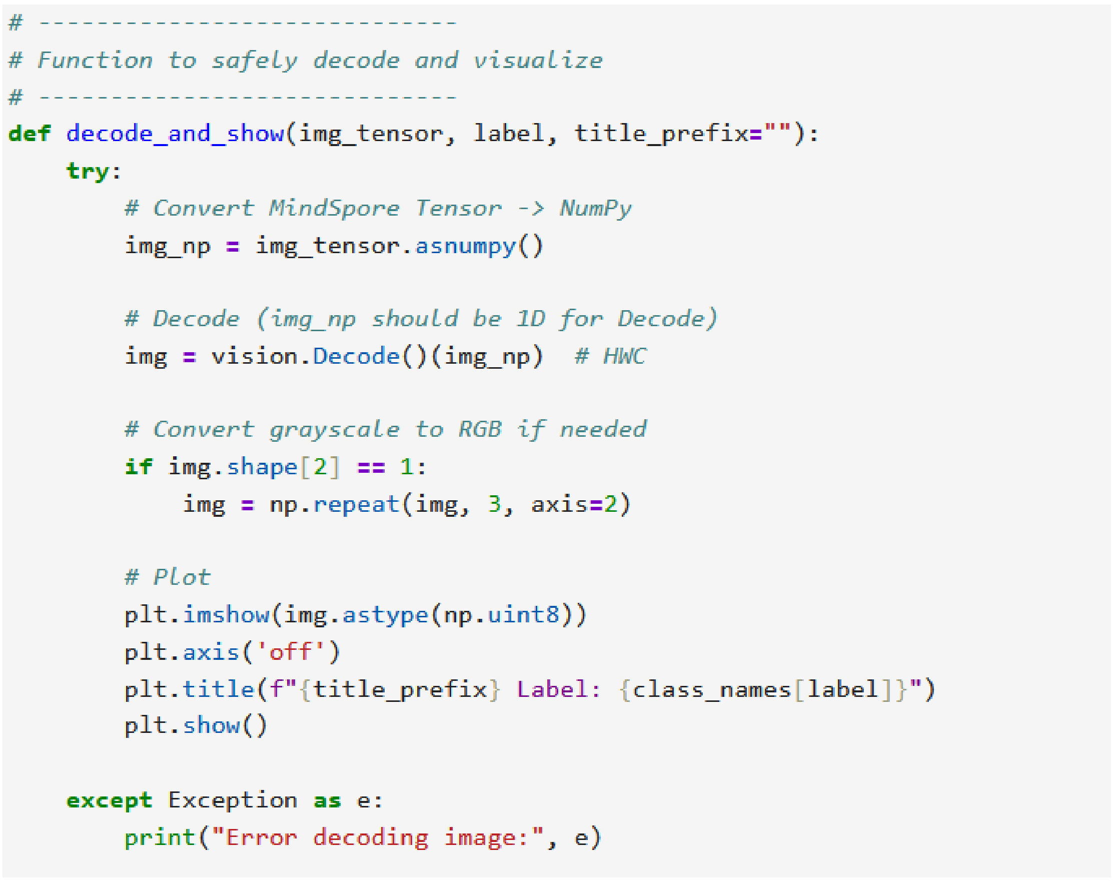
```
# Standardize features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Convert to MindSpore tensors
X_train = Tensor(X_train, ms.float32)
y_train = Tensor(y_train, ms.int32)
X_test = Tensor(X_test, ms.float32)
y_test = Tensor(y_test, ms.int32)

# ========================
# 2. Define Dataset Loader
# ========================
def create_dataset(X, y, batch_size=64, shuffle=True):
    dataset = ms.dataset.NumpySlicesDataset(
        {"features": X, "labels": y}, shuffle=shuffle
    )
    dataset = dataset.batch(batch_size)
    return dataset

train_dataset = create_dataset(X_train, y_train)
test_dataset = create_dataset(X_test, y_test, shuffle=False)
print('Data after split - Training set')
print('************************************\n')
print(X_train,y_train)
print('Data after split - Test set')
print('************************************\n')
```

Figure 14 – Data pre-processing and Visualization on .csv file



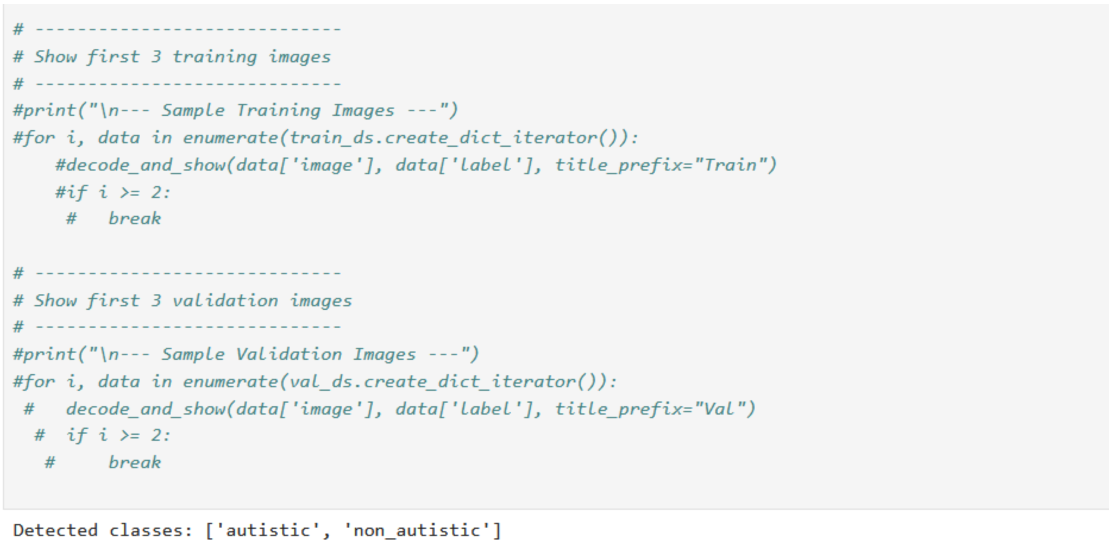
```
import mindspore.dataset as ds
from mindspore.dataset import vision
import matplotlib.pyplot as plt
import numpy as np

# -----------------------------
# Paths
# -----------------------------
train_root = r"D:\data\train"
val_root   = r"D:\data\val"

# Load datasets WITHOUT decoding
train_ds = ds.ImageFolderDataset(dataset_dir=train_root, shuffle=True, decode=False)
val_ds   = ds.ImageFolderDataset(dataset_dir=val_root, shuffle=False, decode=False)

# Detect class names
class_names = [d for d in os.listdir(train_root) if os.path.isdir(os.path.join(train_root,d))]
print("Detected classes:", class_names)
```

Figure 15 – Data Access and Verification of Class Labels



```python
# -----------------------------
# Function to safely decode and visualize
# -----------------------------
def decode_and_show(img_tensor, label, title_prefix=""):
    try:
        # Convert MindSpore Tensor -> NumPy
        img_np = img_tensor.asnumpy()

        # Decode (img_np should be 1D for Decode)
        img = vision.Decode()(img_np)  # HWC

        # Convert grayscale to RGB if needed
        if img.shape[2] == 1:
            img = np.repeat(img, 3, axis=2)

        # Plot
        plt.imshow(img.astype(np.uint8))
        plt.axis('off')
        plt.title(f"{title_prefix} Label: {class_names[label]}")
        plt.show()

    except Exception as e:
        print("Error decoding image:", e)
```

Figure 16 – Data Decoding for Color Images



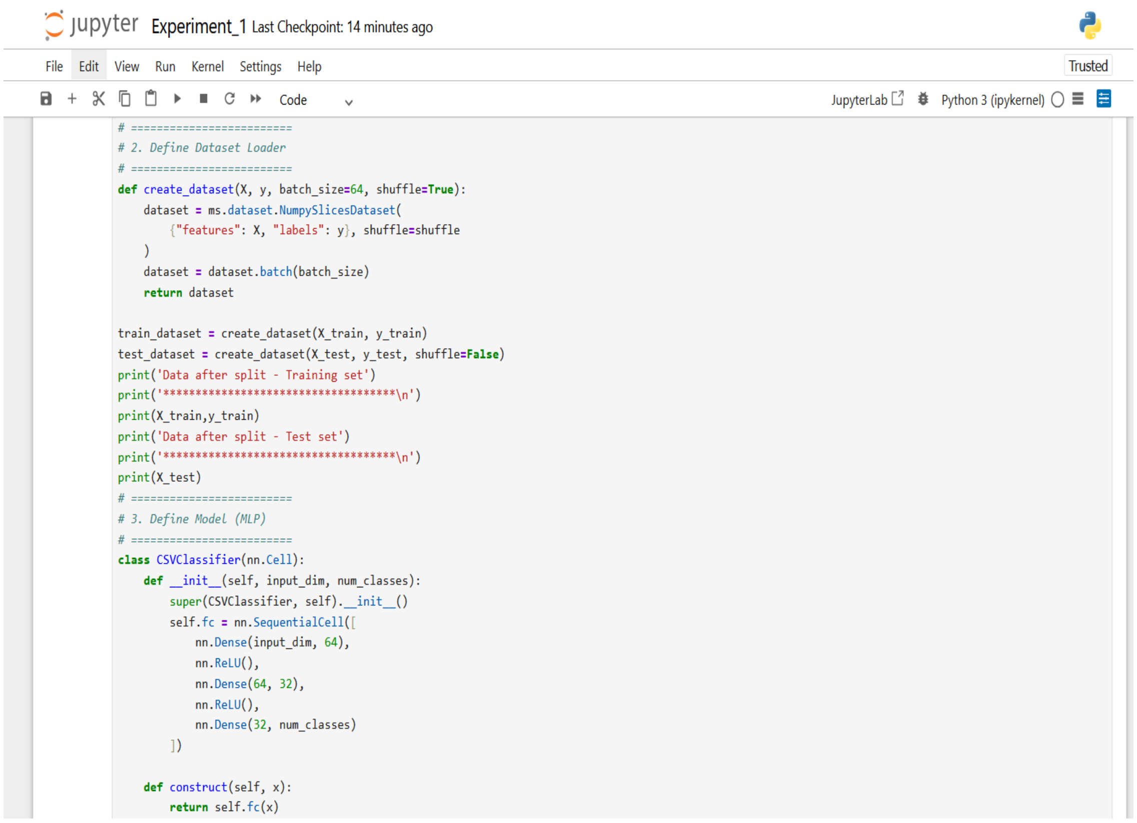
```python
# -----------------------------
# Show first 3 training images
# -----------------------------
#print("\n--- Sample Training Images ---")
#for i, data in enumerate(train_ds.create_dict_iterator()):
    #decode_and_show(data['image'], data['label'], title_prefix="Train")
    #if i >= 2:
     #   break

# -----------------------------
# Show first 3 validation images
# -----------------------------
#print("\n--- Sample Validation Images ---")
#for i, data in enumerate(val_ds.create_dict_iterator()):
 #   decode_and_show(data['image'], data['label'], title_prefix="Val")
  #  if i >= 2:
   #     break
```

```
Detected classes: ['autistic', 'non_autistic']
```

Figure 17 – Visualization of Image Quality

Step 2: Default parameters – Fine tuning and Execution*:*

The learning models are initially run with the default parameters. In the next run of the models, learning rate, epochs, random_seed values are modified as trial-error or based on previous experimental performances. The results are compared to identify the optimal combination of parameters for the given model.

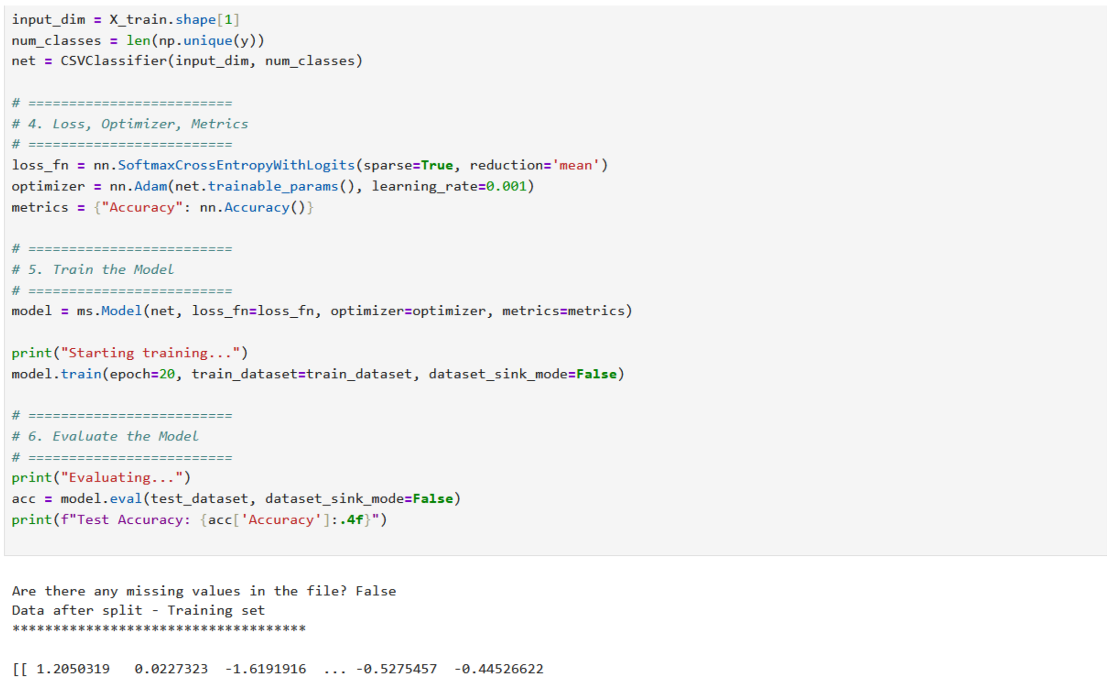

Jupyter Experiment_1 Last Checkpoint: 14 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
# ========================
# 2. Define Dataset Loader
# ========================
def create_dataset(X, y, batch_size=64, shuffle=True):
    dataset = ms.dataset.NumpySlicesDataset(
        {"features": X, "labels": y}, shuffle=shuffle
    )
    dataset = dataset.batch(batch_size)
    return dataset

train_dataset = create_dataset(X_train, y_train)
test_dataset = create_dataset(X_test, y_test, shuffle=False)
print('Data after split - Training set')
print('************************************\n')
print(X_train,y_train)
print('Data after split - Test set')
print('************************************\n')
print(X_test)
# ========================
# 3. Define Model (MLP)
# ========================
class CSVClassifier(nn.Cell):
    def __init__(self, input_dim, num_classes):
        super(CSVClassifier, self).__init__()
        self.fc = nn.SequentialCell([
            nn.Dense(input_dim, 64),
            nn.ReLU(),
            nn.Dense(64, 32),
            nn.ReLU(),
            nn.Dense(32, num_classes)
        ])

    def construct(self, x):
        return self.fc(x)
```

Figure 18 – Hyper-parameter setting

Step 3: Supervised and Unsupervised learning:

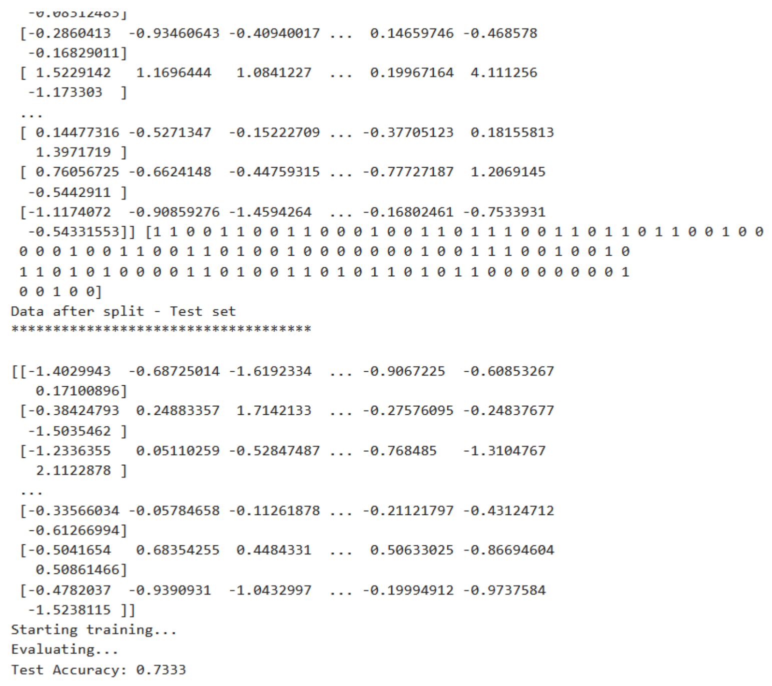
The MLP, K-Means and ResNet Models are executed on the .csv and image datasets respectively. The parameters of the pre-trained models are further fine-tuned. Moreover, the image data required decoding, scaling, resizing and normalization along with verification of the image quality before the models could be executed on the uploaded data.



```
input_dim = X_train.shape[1]
num_classes = len(np.unique(y))
net = CSVClassifier(input_dim, num_classes)

# ========================
# 4. Loss, Optimizer, Metrics
# ========================
loss_fn = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')
optimizer = nn.Adam(net.trainable_params(), learning_rate=0.001)
metrics = {"Accuracy": nn.Accuracy()}

# ========================
# 5. Train the Model
# ========================
model = ms.Model(net, loss_fn=loss_fn, optimizer=optimizer, metrics=metrics)

print("Starting training...")
model.train(epoch=20, train_dataset=train_dataset, dataset_sink_mode=False)

# ========================
# 6. Evaluate the Model
# ========================
print("Evaluating...")
acc = model.eval(test_dataset, dataset_sink_mode=False)
print(f"Test Accuracy: {acc['Accuracy']:.4f}")
```

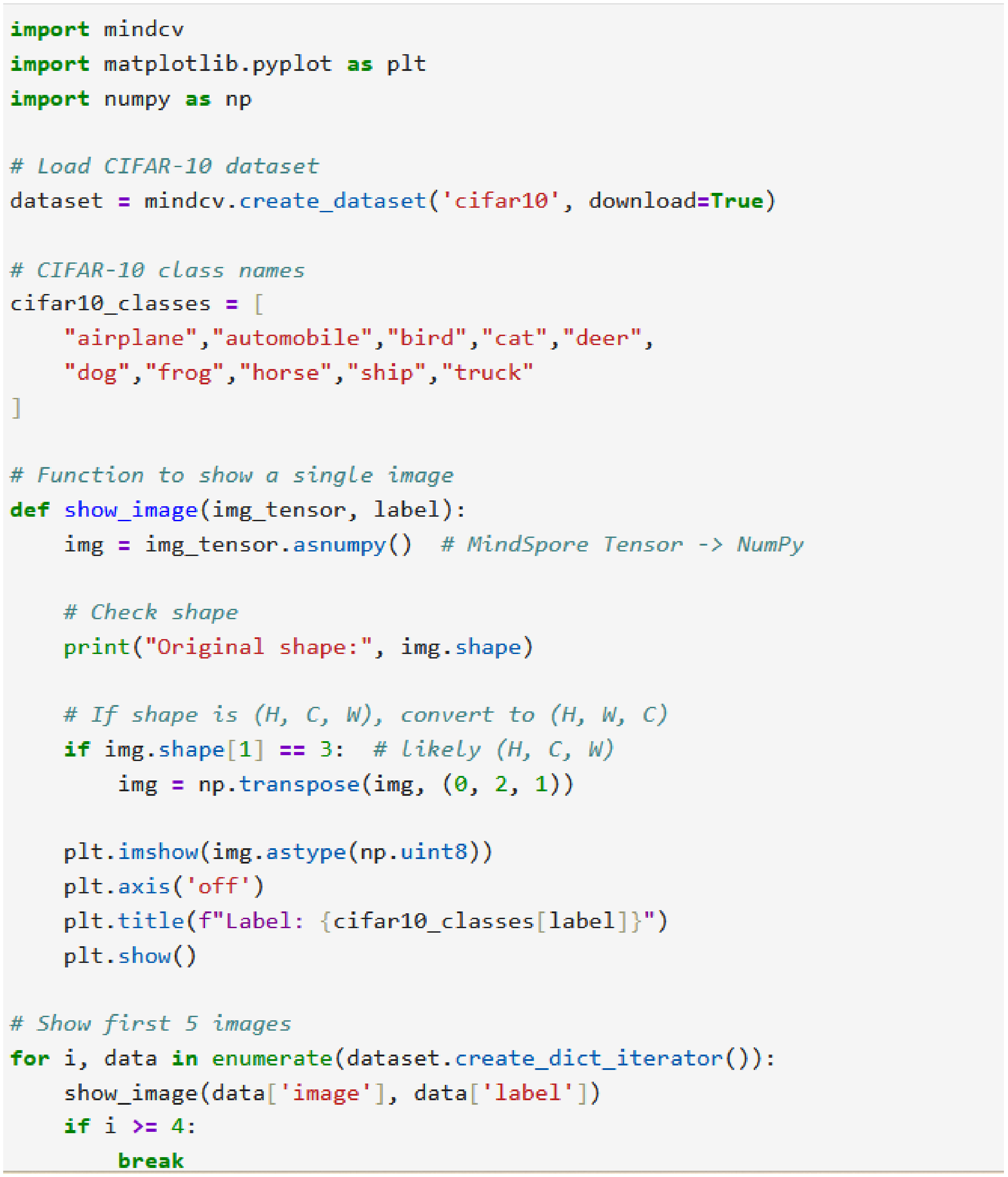
```
Are there any missing values in the file? False
Data after split - Training set
************************************

[[ 1.2050319   0.0227323  -1.6191916  ... -0.5275457  -0.44526622
```

Figure 19 – Evaluation Metrics and Visualization of results – Missing Data



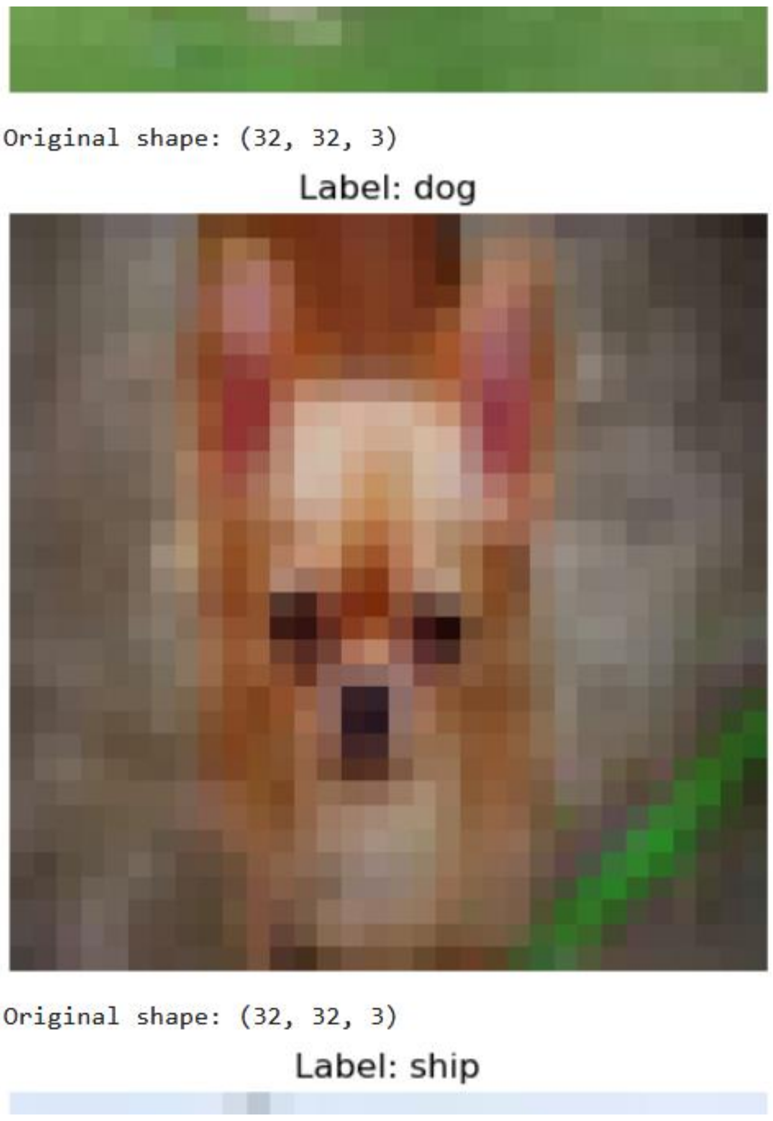
```
  -0.08312483]
 [-0.2860413  -0.93460643 -0.40940017 ...  0.14659746 -0.468578
  -0.16829011]
 [ 1.5229142   1.1696444   1.0841227  ...  0.19967164  4.111256
  -1.173303  ]
 ...
 [ 0.14477316 -0.5271347  -0.15222709 ... -0.37705123  0.18155813
   1.3971719 ]
 [ 0.76056725 -0.6624148  -0.44759315 ... -0.77727187  1.2069145
  -0.5442911 ]
 [-1.1174072  -0.90859276 -1.4594264  ... -0.16802461 -0.7533931
  -0.54331553]] [1 1 0 0 1 1 0 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 0 1 1 0 0 1 0 0
 0 0 0 1 0 0 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0
 1 1 0 1 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1
 0 0 1 0 0]
Data after split - Test set
************************************

[[-1.4029943  -0.68725014 -1.6192334  ... -0.9067225  -0.60853267
   0.17100896]
 [-0.38424793  0.24883357  1.7142133  ... -0.27576095 -0.24837677
  -1.5035462 ]
 [-1.2336355   0.05110259 -0.52847487 ... -0.768485   -1.3104767
   2.1122878 ]
 ...
 [-0.33566034 -0.05784658 -0.11261878 ... -0.21121797 -0.43124712
  -0.61266994]
 [-0.5041654   0.68354255  0.4484331  ...  0.50633025 -0.86694604
   0.50861466]
 [-0.4782037  -0.9390931  -1.0432997  ... -0.19994912 -0.9737584
  -1.5238115 ]]
Starting training...
Evaluating...
Test Accuracy: 0.7333
```

Figure 20 – Evaluation Metrics and Visualization of results – Model Accuracy



```python
import mindcv
import matplotlib.pyplot as plt
import numpy as np

# Load CIFAR-10 dataset
dataset = mindcv.create_dataset('cifar10', download=True)

# CIFAR-10 class names
cifar10_classes = [
    "airplane","automobile","bird","cat","deer",
    "dog","frog","horse","ship","truck"
]

# Function to show a single image
def show_image(img_tensor, label):
    img = img_tensor.asnumpy()  # MindSpore Tensor -> NumPy

    # Check shape
    print("Original shape:", img.shape)

    # If shape is (H, C, W), convert to (H, W, C)
    if img.shape[1] == 3:  # likely (H, C, W)
        img = np.transpose(img, (0, 2, 1))

    plt.imshow(img.astype(np.uint8))
    plt.axis('off')
    plt.title(f"Label: {cifar10_classes[label]}")
    plt.show()

# Show first 5 images
for i, data in enumerate(dataset.create_dict_iterator()):
    show_image(data['image'], data['label'])
    if i >= 4:
        break
```

Figure 21 – Data upload of In-built Cifar-10 data and Visualization of Images



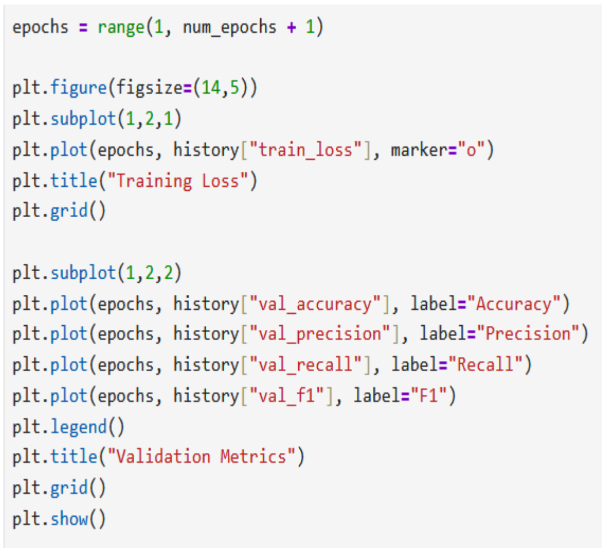
```
Original shape: (32, 32, 3)
```

```
Original shape: (32, 32, 3)
```

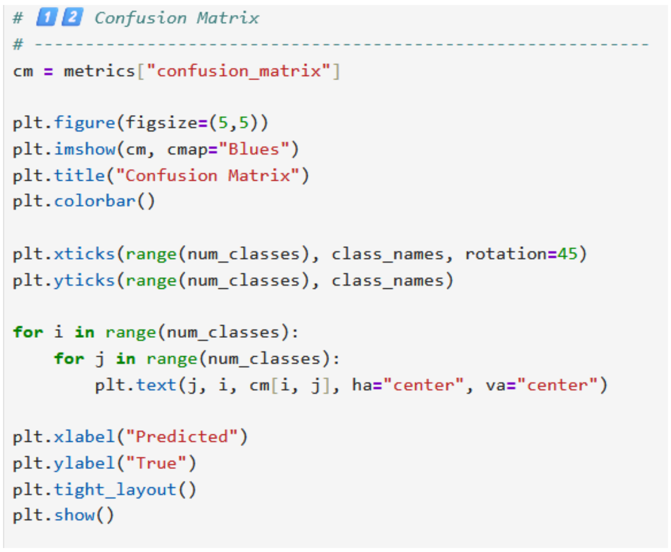
Figure 22 – Visualization of In-built Image Data in MindSpore

Step 4: Validation and Visualization of Results:

Once the model execution is done and results are obtained, the visualization of results enable us to identify if the accuracy level attained is significant enough to ensure the model is capable of distinguishing between images. Generally an accuracy of ~95% is considered significant. If not, further tuning and training needs to be carried out. Other better performing models can be executed on the data as well.

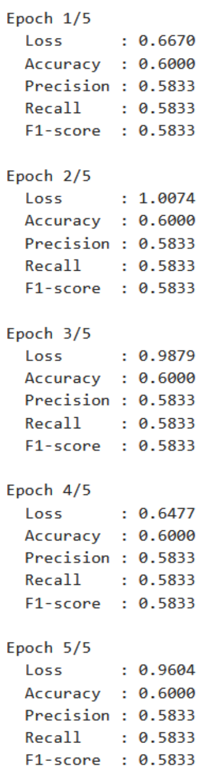


```python
epochs = range(1, num_epochs + 1)

plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
plt.plot(epochs, history["train_loss"], marker="o")
plt.title("Training Loss")
plt.grid()

plt.subplot(1,2,2)
plt.plot(epochs, history["val_accuracy"], label="Accuracy")
plt.plot(epochs, history["val_precision"], label="Precision")
plt.plot(epochs, history["val_recall"], label="Recall")
plt.plot(epochs, history["val_f1"], label="F1")
plt.legend()
plt.title("Validation Metrics")
plt.grid()
plt.show()
```

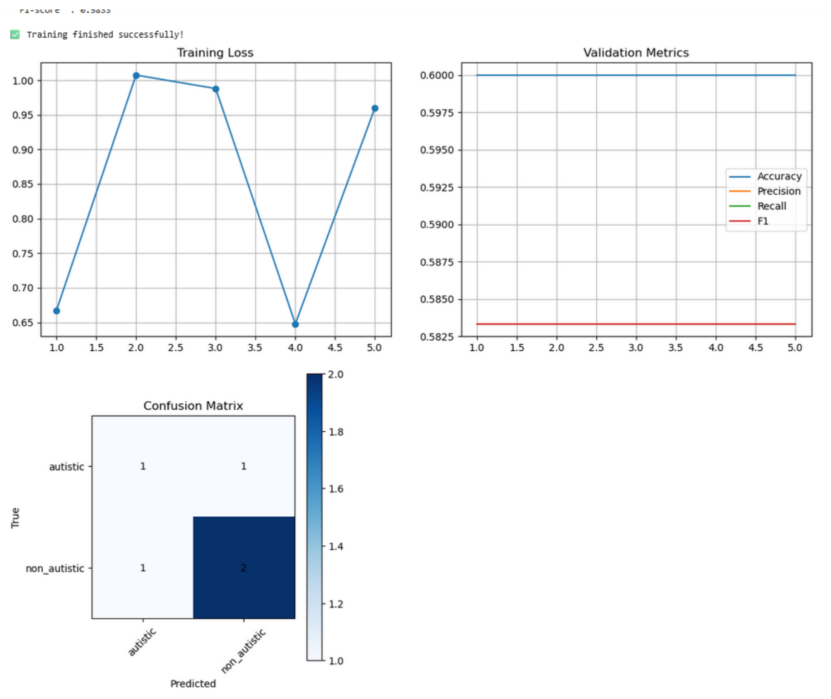


```python
# 1️⃣2️⃣ Confusion Matrix
# ------------------------------------------------------------
cm = metrics["confusion_matrix"]

plt.figure(figsize=(5,5))
plt.imshow(cm, cmap="Blues")
plt.title("Confusion Matrix")
plt.colorbar()

plt.xticks(range(num_classes), class_names, rotation=45)
plt.yticks(range(num_classes), class_names)

for i in range(num_classes):
    for j in range(num_classes):
        plt.text(j, i, cm[i, j], ha="center", va="center")

plt.xlabel("Predicted")
plt.ylabel("True")
plt.tight_layout()
plt.show()
```

Figure 23 – Calculation of Evaluation Metrics for Autistic Data Classification in MindSpore



```
Epoch 1/5
  Loss      : 0.6670
  Accuracy  : 0.6000
  Precision : 0.5833
  Recall    : 0.5833
  F1-score  : 0.5833

Epoch 2/5
  Loss      : 1.0074
  Accuracy  : 0.6000
  Precision : 0.5833
  Recall    : 0.5833
  F1-score  : 0.5833

Epoch 3/5
  Loss      : 0.9879
  Accuracy  : 0.6000
  Precision : 0.5833
  Recall    : 0.5833
  F1-score  : 0.5833

Epoch 4/5
  Loss      : 0.6477
  Accuracy  : 0.6000
  Precision : 0.5833
  Recall    : 0.5833
  F1-score  : 0.5833

Epoch 5/5
  Loss      : 0.9604
  Accuracy  : 0.6000
  Precision : 0.5833
  Recall    : 0.5833
  F1-score  : 0.5833
```



```
F1-score  : 0.5833
✅ Training finished successfully!
```

Figure 24 – Metric Visualization of Autistic Image Data Classification in MindSpore

- The entire project with all the datasets and the model execution with visualization is available at ***https://github.com/gracia2026/Autism-classificatio-with-MindSpore.git***

IV. Further Practice

The students on completion of this course would have gained the experience to use different packages in MindSpore, resolve version conflicts and execute machine learning models with hyperparameter optimization. With this knowledge, further activities would focus on:

1. Integrating with a cloud platform (ModelArts, OpenI) for larger datasets and verify the model performance.
2. Since Ascend and CANN architecture integration with MindSpore is suitable to the Linux OS, extending the implementation of the models with high computing Ascend platform would be an efficient path to expand the learning trajectory.
3. Benchmarking other machine learning/deep learning models on the MindSpore platform and rank the performance based on metrics for both balanced and unbalanced datasets.
4. Review the available data cleaning methods available in Mindspore to measure their impact on model performance.
5. Expand the application to other inter-disciplinary fields of study – corrosion detection, climate change causes, student performance prediction and so on to recognize the impact of AI in enhancing sustainability across healthcare, environment and education.

V. Summary

The following key learning objectives have been achieved through this Mindspore-Based Learning Model for Autism Classification.

1. Recognizing the need to apply machine learning to real-world problems:

Data is increasing in size every minute and the use of a comprehensive AI computing platform that can seamless work across device, edge and cloud platforms to process and learn from the humungous data, is the need of the hour. This experiment targets a real-world problem can be extended to other application areas. Moreover the Huawei course material and lab experiment clearly outlines the steps to implement the learning pipeline.

2. Learning the role of data organization, type and preparation for machine/deep learning:

Data collected, require collation, cleaning, organization and preparation for application of learning models and visualization of results. This experiment clearly demonstrates the different approach to learning depending on data size, type, and quality.

3. Mastering the setup and use of MindSpore platform and possible expansion to big data:

The steps to set up the MS environment along with the libraries/packages needed to execute data transformation and decoding, manage version conflicts, and handle deprecated versions portray the pace at which technology is advancing. The introduction to the set up and environment, Huawei tutorials to migrate and run ExeML on ModelArts with a no-code development is a game changer for big data analytics and better performance.

4. Applying the hyper-parameter tuning to optimize the models for evaluating their performance.:

Model performance evaluation methods and metrics are critical to enhance and deploy the models. Every algorithm has a unique set of parameters that need to be observed and optimized to attain maximum performance.

5. Visualizing data and results of learning models that are compliant with Ethical and Societal Practices:

Data visualization and portrayal of results is a key feature for the success of a *learning pipeline. Moreover, while using real-world datasets, the need to confirm ethical issues governing the public use of personal information (images or text) is also a crucial part of industry-institute collaboration where sensitive information is processed and learned by systems.*

6. Creating a machine/deep learning pipeline to infer knowledge from structured and image datasets:

This experiment along with the theoretical basis of machine learning helps us understand, develop, visualize and validate the complete machine learning/deep learning framework for both supervised and unsupervised learning on structured data and deep learning from image data.