01_data_ingestion_03_data_transformation

April 8, 2024

[1]:
```
import os
```

[2]:
```
%pwd
```

[2]:
```
'D:\\Desktop\\Deep Learning\\Lab 7\\Deep-Learning-Model-Customization-and-
Performance-Evaluation\\Research\\Cifar10'
```

[3]:
```
os.chdir("../")
os.chdir("../")
```

[4]:
```
%pwd
```

[4]:
```
'D:\\Desktop\\Deep Learning\\Lab 7\\Deep-Learning-Model-Customization-and-
Performance-Evaluation'
```

[5]:
```
import logging
from pathlib import Path
logging.basicConfig(
    # filename='extract_data.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)
```

[6]:
```
print(Path(os.getcwd()))
```

```
D:\Desktop\Deep Learning\Lab 7\Deep-Learning-Model-Customization-and-
Performance-Evaluation
```

[7]:
```
import logging
import os
from dataclasses import dataclass
from pathlib import Path
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
```

```
@dataclass
class DataTransformationConfig:
    root_dir: Path
    X_train_file: Path
    y_train_file: Path
    X_test_file: Path
    y_test_file: Path


class ConfigurationManager:
    def __init__(self):
        self.root_dir = Path(os.getcwd())
        self.X_train_file = self.root_dir / "Dataset/Modeltraining/Cifar10/
 ↪X_train.npy"
        self.y_train_file = self.root_dir / "Dataset/Modeltraining/Cifar10/
 ↪y_train.npy"
        self.X_test_file = self.root_dir / "Dataset/Modeltraining/Cifar10/
 ↪X_test.npy"
        self.y_test_file = self.root_dir / "Dataset/Modeltraining/Cifar10/
 ↪y_test.npy"

    def get_data_transformation_config(self) -> DataTransformationConfig:
        return DataTransformationConfig(
            root_dir=self.root_dir,
            X_train_file=self.X_train_file,
            y_train_file=self.y_train_file,
            X_test_file=self.X_test_file,
            y_test_file=self.y_test_file
        )

class DataTransformation:
    def __init__(self, config: DataTransformationConfig):
        self.config = config

    def ensure_directories_exist(self):
        """Ensure that directories for all file paths exist."""
        for path in [self.config.X_train_file, self.config.y_train_file, self.
 ↪config.X_test_file, self.config.y_test_file]:
            path.parent.mkdir(parents=True, exist_ok=True)

    def ensure_uniform_class_distribution(self, X, y):
        unique, counts = np.unique(y, return_counts=True)
        min_count = np.min(counts)
        avg_count = np.mean(counts)
        # Logging the details about class distributions
        logging.info(f"Number of unique classes in y: {len(unique)}")
        logging.info(f"Unique classes in y: {unique}")
        logging.info(f"Counts of each class in y: {counts}")
        logging.info(f"Average frequency of all classes: {avg_count}")
        logging.info(f"Minimum frequency among classes: {min_count}")

        X_list = []
        y_list = []
        for class_value in unique:
            class_indices = np.where(y == class_value)[0]
            np.random.shuffle(class_indices)
            selected_indices = class_indices[:min_count]
            X_list.append(X[selected_indices, :])
            y_list.append(y[selected_indices])

            # Logging details about data removal for balancing
            logging.info(f"Class {class_value} reduced to {min_count} instances␣
↪for balancing.")

        X_balanced = np.concatenate(X_list, axis=0)
        y_balanced = np.concatenate(y_list, axis=0)

        # Shuffling the data to mix the classes
        indices = np.arange(X_balanced.shape[0])
        np.random.shuffle(indices)
        X_balanced = X_balanced[indices]
        y_balanced = y_balanced[indices]

        # Logging final dataset size after balancing
        logging.info(f"Number of rows in the balanced dataset: {X_balanced.
↪shape[0]}")

        return X_balanced, y_balanced

    def train_test_splitting(self):

        self.ensure_directories_exist()

        # Ensure the directory for the status file exists
        self.config.root_dir.mkdir(parents=True, exist_ok=True)  # This line␣
↪ensures that the root directory exists

        # Load and preprocess CIFAR-10 dataset
        logging.info("Loading and preprocessing CIFAR-10 dataset...")
        (X_train, y_train), (X_test, y_test) = cifar10.load_data()
        X_train, X_test = X_train / 255.0, X_test / 255.0
        y_train, y_test = to_categorical(y_train, 10), to_categorical(y_test, 10)

        # Combine X_train and X_test to create X
        X = np.concatenate((X_train, X_test), axis=0)

        # Combine y_train and y_test to create y
        y = np.concatenate((y_train, y_test), axis=0)

        # Ensure uniform class distribution
        X_balanced, y_balanced = self.ensure_uniform_class_distribution(X, y)

        # Split the dataset into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X_balanced, y_balanced, test_size=0.2, random_state=42)

        # Save the split data as numpy files
        np.save(self.config.X_train_file, X_train)
        logging.info(f"X_train data saved to {self.config.X_train_file}")
        np.save(self.config.y_train_file, y_train)
        logging.info(f"y_train data saved to {self.config.y_train_file}")
        np.save(self.config.X_test_file, X_test)
        logging.info(f"X_test data saved to {self.config.X_test_file}")
        np.save(self.config.y_test_file, y_test)
        logging.info(f"y_test data saved to {self.config.y_test_file}")

def main():
    config_manager = ConfigurationManager()
    data_transformation_config = config_manager.get_data_transformation_config()
    data_transformation = DataTransformation(data_transformation_config)
    data_transformation.train_test_splitting()

if __name__ == "__main__":
    main()
```

```
2024-04-08 14:38:12 - WARNING - From D:\Desktop\Deep Learning\Lab 2\MNSIT-
MLPClassifer\venv\lib\site-packages\keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

2024-04-08 14:38:12 - INFO - Loading and preprocessing CIFAR-10 dataset…
2024-04-08 14:38:13 - INFO - Number of unique classes in y: 2
2024-04-08 14:38:13 - INFO - Unique classes in y: [0. 1.]
2024-04-08 14:38:13 - INFO - Counts of each class in y: [540000  60000]
2024-04-08 14:38:13 - INFO - Average frequency of all classes: 300000.0
2024-04-08 14:38:13 - INFO - Minimum frequency among classes: 60000
2024-04-08 14:38:13 - INFO - Class 0.0 reduced to 60000 instances for balancing.
```

```
2024-04-08 14:38:14 - INFO - Class 1.0 reduced to 60000 instances for balancing.
2024-04-08 14:38:15 - INFO - Number of rows in the balanced dataset: 120000
2024-04-08 14:38:31 - INFO - X_train data saved to D:\Desktop\Deep Learning\Lab
7\Deep-Learning-Model-Customization-and-Performance-
Evaluation\Dataset\Modeltraining\Cifar10\X_train.npy
2024-04-08 14:38:31 - INFO - y_train data saved to D:\Desktop\Deep Learning\Lab
7\Deep-Learning-Model-Customization-and-Performance-
Evaluation\Dataset\Modeltraining\Cifar10\y_train.npy
2024-04-08 14:38:35 - INFO - X_test data saved to D:\Desktop\Deep Learning\Lab
7\Deep-Learning-Model-Customization-and-Performance-
Evaluation\Dataset\Modeltraining\Cifar10\X_test.npy
2024-04-08 14:38:35 - INFO - y_test data saved to D:\Desktop\Deep Learning\Lab
7\Deep-Learning-Model-Customization-and-Performance-
Evaluation\Dataset\Modeltraining\Cifar10\y_test.npy
```

[13]:

```python
import pandas as pd
import numpy as np

# Specify the path to your numpy array file
npy_file_path = 'Dataset/Modeltraining/Cifar10/X_train.npy'

# Load the numpy array file
data = np.load(npy_file_path)

# Reshape the 4D array into a 2D array
num_images = data.shape[0]
num_pixels = data.shape[1] * data.shape[2] * data.shape[3]
data_reshaped = data.reshape(num_images, num_pixels)

# Convert the 2D array to a DataFrame
df = pd.DataFrame(data_reshaped)

# Display the DataFrame
df
```

[13]:

```
              0         1         2         3         4         5         6    \
0      0.560784  0.552941  0.533333  0.541176  0.529412  0.494118  0.478431
1      0.996078  0.996078  0.996078  0.992157  0.992157  0.992157  0.992157
2      0.423529  0.721569  0.952941  0.431373  0.717647  0.937255  0.443137
3      0.560784  0.580392  0.678431  0.572549  0.584314  0.682353  0.556863
4      0.247059  0.384314  0.701961  0.247059  0.384314  0.701961  0.250980
…           …         …         …         …         …         …         …
95995  0.286275  0.321569  0.329412  0.254902  0.270588  0.266667  0.345098
95996  0.968627  0.862745  0.835294  0.960784  0.862745  0.835294  0.956863
95997  0.717647  0.611765  0.478431  0.713725  0.611765  0.478431  0.729412
95998  0.992157  1.000000  0.996078  0.996078  1.000000  0.996078  1.000000
95999  0.521569  0.596078  0.439216  0.494118  0.580392  0.419608  0.560784
```

```
              7         8         9  …      3062      3063      3064  \
0      0.466667  0.419608  0.498039  …  0.109804  0.486275  0.309804
1      0.992157  0.992157  0.996078  …  0.086275  0.121569  0.149020
2      0.721569  0.937255  0.443137  …  0.580392  0.525490  0.580392
3      0.560784  0.662745  0.576471  …  0.470588  0.431373  0.384314
4      0.388235  0.705882  0.258824  …  0.705882  0.235294  0.345098
…           …       …       …  …       …       …       …
95995  0.329412  0.329412  0.490196  …  0.231373  0.400000  0.356863
95996  0.870588  0.839216  0.956863  …  0.752941  0.811765  0.729412
95997  0.623529  0.494118  0.729412  …  0.345098  0.662745  0.545098
95998  1.000000  1.000000  0.980392  …  1.000000  1.000000  1.000000
95999  0.631373  0.478431  0.701961  …  0.129412  0.219608  0.243137

           3065      3066      3067      3068      3069      3070      3071
0      0.117647  0.498039  0.317647  0.129412  0.498039  0.317647  0.125490
1      0.086275  0.078431  0.094118  0.043137  0.078431  0.047059  0.027451
2      0.572549  0.505882  0.580392  0.576471  0.501961  0.560784  0.560784
3      0.541176  0.415686  0.368627  0.525490  0.396078  0.349020  0.498039
4      0.701961  0.239216  0.352941  0.709804  0.239216  0.349020  0.709804
…           …       …       …       …       …       …       …
95995  0.203922  0.317647  0.298039  0.152941  0.266667  0.250980  0.109804
95996  0.749020  0.803922  0.725490  0.745098  0.807843  0.717647  0.729412
95997  0.345098  0.658824  0.537255  0.337255  0.654902  0.537255  0.337255
95998  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000
95999  0.156863  0.243137  0.262745  0.184314  0.313725  0.337255  0.243137

[96000 rows x 3072 columns]
```

[25]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Specify the path to your numpy array file
npy_file_path = 'Dataset/Modeltraining/Cifar10/X_train.npy'

# Load the numpy array file
data = np.load(npy_file_path)

# Reshape the 4D array into a 2D array
num_images = data.shape[0]
num_pixels = data.shape[1] * data.shape[2] * data.shape[3]
data_reshaped = data.reshape(num_images, num_pixels)

# Convert the 2D array to a DataFrame
df = pd.DataFrame(data_reshaped)
```

```python
# Select a random row number (change this as per your requirement)
row_number = 10

# Inverse transform the scaled row to get the original image
original_image = df.iloc[row_number].values.reshape(32, 32, 3)

# Display the image
plt.imshow(original_image)
plt.title(f"row_number: {row_number}")  # Display the row number as the label
plt.axis('off')
plt.show()
```

[20]:

```python
import pandas as pd
import numpy as np

# Specify the path to your numpy array file
npy_file_path = 'Dataset/Modeltraining/Cifar10/y_train.npy'

# Load the numpy array file
data = np.load(npy_file_path)

# Reshape the 1D array into a 2D array with one column
```

```
data_reshaped = data.reshape(-1, 1)

# Convert the 2D array to a DataFrame
df = pd.DataFrame(data_reshaped, columns=['Label'])

# Display the DataFrame
df.head()
```

[20]:

```
   Label
0    0.0
1    0.0
2    0.0
3    1.0
4    0.0
```

[ ]:

```

```