Documentation for png prototype.ipynb

Code Summary Description

This code implements a pipeline for preprocessing and classifying radio burst spectrogram images using both a custom CNN model and transfer learning with pre-trained models. The data, organized into folders by label, is loaded and preprocessed to remove background noise. A custom CNN is defined, trained, and evaluated using train-test splits, with performance visualized through accuracy and loss plots. Additionally, the code performs K-Fold cross-validation on multiple pre-trained models (e.g., VGG16, ResNet50, EfficientNet) using frozen base layers with custom dense layers on top. Model evaluation includes accuracy, confusion matrix visualization, and saving the best-performing model during training.

Code Cell 1

import pandas as pd → This line imports a library or module.

from tensorflow.keras import layers → Imports specific parts (functions, classes) of a library.

import numpy as np → This line imports a library or module.

import matplotlib.pyplot as plt → This line imports a library or module.

import sys → This line imports a library or module.

from tqdm import tqdm → Imports specific parts (functions, classes) of a library.

import os → This line imports a library or module.

import cv2 → This line imports a library or module.

from concurrent.futures import ThreadPoolExecutor → Imports specific parts (functions, classes) of a library.

sys.path.append(r"D:\My Laptop\Me\Programming\Machine Learning\Internships\Egyptian Space Agency\2012") → Adds a custom path to the system path for importing modules.

→ Performs a specific operation in the code.

Now you can import your custom library → This is a comment explaining the following code.

from read_Data import read_fits → Imports specific parts (functions, classes) of a library.

Code Cell 2

os.listdir('radio bursts data') → Lists the contents of a directory.

Code Cell 3

print(len(os.listdir(os.path.join("radio bursts data",'noise and empty')))/2) → Lists the contents of a directory.

print(len(os.listdir(os.path.join("radio bursts data",'rbtype2')))/2) → Lists the contents of a directory.

print(len(os.listdir(os.path.join("radio bursts data",'rbtype3')))/2) → Lists the contents of a directory.

Code Cell 4

def read_image(full_img_path): → Performs a specific operation in the code.

try: → Performs a specific operation in the code.

img = cv2.imread(full_img_path) → Uses OpenCV library for computer vision tasks.

return img → Returns a value or result from a function.

except Exception as e: → Performs a specific operation in the code.

print(f"Warning: Error reading image '{full_img_path}': {e}") → Prints values or output to the console.

return None → Returns a value or result from a function.

Code Cell 5

→ Performs a specific operation in the code.

def read_data(path, verbose=False,shape=(128,128)): → Performs a specific operation in the code.

images = [] → Performs a specific operation in the code.

labels = [] → Performs a specific operation in the code.

names =[] → Performs a specific operation in the code.

for img_type in tqdm(os.listdir(path)): → Lists the contents of a directory.

type_path = os.path.join(path, img_type) → Creates file or directory paths.

img_paths = [os.path.join(type_path, img) for img in os.listdir(type_path) if img.endswith('png')] → Lists the contents of a directory.

→ Performs a specific operation in the code.

Use ThreadPoolExecutor for parallel image reading → Iterates over a sequence or collection.

with ThreadPoolExecutor() as executor: → Performs a specific operation in the code.

results = list(executor.map(read_image, img_paths)) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Filter out None results and extend images and labels lists → This is a comment explaining the following code.

for img, img_path in zip(results, img_paths): → Iterates over a sequence or collection.

if img is not None: → Performs a specific operation in the code.

img = cv2.resize(img,shape) → Uses OpenCV library for computer vision tasks.

images.append(img) → Performs a specific operation in the code.

labels.append(img_type) → Performs a specific operation in the code.

names.append(img_path) → Performs a specific operation in the code.

if verbose: → Performs a specific operation in the code.

print(f"Read file: {img_path}") → Prints values or output to the console.

→ Performs a specific operation in the code.

print(f'Finished reading {len(img_paths)} images of type "{img_type}".') → Prints values or output to the console.

return np.array(images),np.array(names),np.array(labels) → Returns a value or result from a function.

→ Performs a specific operation in the code.

Adjust the verbosity as needed → This is a comment explaining the following code.

→ Performs a specific operation in the code.

Code Cell 6

images,pathes, labels = read_data('radio bursts data', verbose=False) → Performs a specific operation in the code.

Code Cell 7

images.shape → Performs a specific operation in the code.

Code Cell 8

len(pathes) → Performs a specific operation in the code.

Code Cell 9

labels.shape → Performs a specific operation in the code.

Code Cell 10

images.shape → Performs a specific operation in the code.

Code Cell 11

from collections import Counter → Imports specific parts (functions, classes) of a library.

Counter(labels) → Performs a specific operation in the code.

Code Cell 12

def preprocess_spectrogram(data): → Performs a specific operation in the code.

""" → Performs a specific operation in the code.

Calculate the median over time for each frequency channel → Performs a specific operation in the code.

and subtract median values from each time column to remove background noise. → Performs a specific operation in the code.

""" → Performs a specific operation in the code.

median_values = np.median(data, axis=0) → Performs a specific operation in the code.

processed_data = data - median_values → Performs a specific operation in the code.

return processed_data → Returns a value or result from a function.

→ Performs a specific operation in the code.

Code Cell 13

preprocessed_spectrogram_data = [preprocess_spectrogram(img) for img in images] → Iterates over a sequence or collection.

preprocessed_spectrogram_data = np.array(preprocessed_spectrogram_data) → Performs a specific operation in the code.

Code Cell 14

print(f"Shape of preprocessed spectrogram data: {preprocessed_spectrogram_data.shape}") → Prints values or output to the console.

Code Cell 15

plt.imshow(images[1], aspect='auto', origin='lower') → Performs a specific operation in the code.

plt.title('Noise') → Performs a specific operation in the code.

plt.xlabel('Time') → Performs a specific operation in the code.

plt.ylabel('Frequency') → Performs a specific operation in the code.

Code Cell 16

plt.imshow(images[-1], aspect='auto', origin='lower') → Performs a specific operation in the code.

plt.title('Type 3') → Performs a specific operation in the code.

plt.xlabel('Time') → Performs a specific operation in the code.

plt.ylabel('Frequency') → Performs a specific operation in the code.

Code Cell 17

plt.imshow(images[700], aspect='auto', origin='lower') → Performs a specific operation in the code.

plt.title('Type 2') → Performs a specific operation in the code.

plt.xlabel('Time') → Performs a specific operation in the code.

plt.ylabel('Frequency') → Performs a specific operation in the code.

Code Cell 18

difference = images - preprocessed_spectrogram_data → Performs a specific operation in the code.

plt.imshow(difference[600], aspect='auto', origin='lower', cmap='viridis') → Performs a specific operation in the code.

plt.title('Difference Between Original and Preprocessed') → Performs a specific operation in the code.

plt.xlabel('Time') → Performs a specific operation in the code.

plt.ylabel('Frequency') → Performs a specific operation in the code.

plt.colorbar() → Performs a specific operation in the code.

plt.show() → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Code Cell 19

difference.shape → Performs a specific operation in the code.

Code Cell 20

preprocessed_spectrogram_data.shape → Performs a specific operation in the code.

Code Cell 21

→ Performs a specific operation in the code.

import tensorflow as tf → This line imports a library or module.

from tensorflow.keras.models import Sequential,Model,load_model → Imports specific parts (functions, classes) of a library.

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout , Add,Input → Imports specific parts (functions, classes) of a library.

from tensorflow.keras.optimizers import Adam → Imports specific parts (functions, classes) of a library.

from sklearn.model_selection import train_test_split → Imports specific parts (functions, classes) of a library.

from sklearn.preprocessing import LabelEncoder → Imports specific parts (functions, classes) of a library.

from tensorflow.keras.callbacks import ModelCheckpoint → Imports specific parts (functions, classes) of a library.

from tensorflow.keras.regularizers import l2 → Imports specific parts (functions, classes) of a library.

Code Cell 22

preprocessed_spectrogram_data = np.expand_dims(preprocessed_spectrogram_data, axis=-1) # Shape will be (num_samples, 128, 128, 1) → This is a comment explaining the following code.

Code Cell 23

difference = np.expand_dims(difference, axis=-1) # Shape will be (num_samples, 128, 128, 1) → This is a comment explaining the following code.

difference.shape → This is a comment explaining the following code.

Code Cell 24

images_fits = np.expand_dims(images, axis=-1) → This is a comment explaining the following code.

images_fits.shape → This is a comment explaining the following code.

Code Cell 25

X_train, X_test, y_train, y_test = train_test_split(→ Performs a specific operation in the code.

preprocessed_spectrogram_data, labels, test_size=0.2, random_state=42,shuffle=True → Performs a specific operation in the code.

) → Performs a specific operation in the code.

Code Cell 26

X_train.shape → Performs a specific operation in the code.

Code Cell 27

X_test.shape → Performs a specific operation in the code.

Code Cell 28

label_encoder = LabelEncoder() → Performs a specific operation in the code.

y_train_encoded = label_encoder.fit_transform(y_train) → Iterates over a sequence or collection.

Code Cell 29

y_test_encoded = label_encoder.transform(y_test) → Performs a specific operation in the code.

Code Cell 30

y_train_encoded → Performs a specific operation in the code.

Code Cell 31

X_train = X_train / 255.0 → Performs a specific operation in the code.

X_test = X_test / 255.0 → Performs a specific operation in the code.

Code Cell 32

def create_cnn_model(input_shape, num_classes): → Performs a specific operation in the code.

model = Sequential() → Performs a specific operation in the code.

→ Performs a specific operation in the code.

First Convolutional Layer → This is a comment explaining the following code.

model.add(Conv2D(32, (3, 3), activation='relu', kernel_regularizer=l2(0.001), input_shape=input_shape)) → Performs a specific operation in the code.

model.add(Dropout(0.2)) → Performs a specific operation in the code.

model.add(MaxPooling2D((2, 2))) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Second Convolutional Layer → This is a comment explaining the following code.

model.add(Conv2D(64, (3, 3), activation='relu')) → Performs a specific operation in the code.

model.add(Dropout(0.3)) → Performs a specific operation in the code.

model.add(MaxPooling2D((2, 2))) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Third Convolutional Layer → This is a comment explaining the following code.

model.add(Conv2D(128, (3, 3), activation='relu')) → Performs a specific operation in the code.

model.add(Dropout(0.5)) → Performs a specific operation in the code.

model.add(MaxPooling2D((2, 2))) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Flattening the output from the convolutional layers → This is a comment explaining the following code.

model.add(Flatten()) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Fully Connected Layer → This is a comment explaining the following code.

model.add(Dense(128, activation='relu')) → Performs a specific operation in the code.

model.add(Dropout(0.5)) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Output Layer → This is a comment explaining the following code.

model.add(Dense(3, activation='softmax')) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

return model → Returns a value or result from a function.

Code Cell 33

input_shape = X_train.shape[1:] # e.g., (128, 128, 1) → Performs a specific operation in the code.

num_classes = len(label_encoder.classes_) → Performs a specific operation in the code.

Code Cell 34

model = create_cnn_model(input_shape, num_classes) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Compile the model → This is a comment explaining the following code.

model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy']) → Performs a specific operation in the code.

Code Cell 35

model.summary() → Performs a specific operation in the code.

Code Cell 36

print(f'y train encoded dim ={y_train_encoded.shape}') → Prints values or output to the console.

print(f'x train dim ={X_train.shape}') → Prints values or output to the console.

→ Performs a specific operation in the code.

Code Cell 37

num_classes = len(label_encoder.classes_) → Performs a specific operation in the code.

print(f"Number of classes: {num_classes}") → Prints values or output to the console.

→ Performs a specific operation in the code.

Code Cell 38

y_train_encoded → Performs a specific operation in the code.

Code Cell 39

y_train → Performs a specific operation in the code.

Code Cell 40

checkpoint = ModelCheckpoint('best_model2.keras', monitor='val_accuracy', save_best_only=True) → Defines or uses machine learning libraries like Keras or TensorFlow.

history = model.fit(→ Performs a specific operation in the code.

X_train, y_train_encoded, → Performs a specific operation in the code.

epochs=20, → Performs a specific operation in the code.

batch_size=32, → Performs a specific operation in the code.

validation_split=0.2, → Performs a specific operation in the code.

callbacks=[checkpoint] → Performs a specific operation in the code.

) → Performs a specific operation in the code.

Code Cell 41

plt.figure(figsize=(10, 5)) → Performs a specific operation in the code.

plt.plot(history.history['accuracy'], label='Train Accuracy') → Performs a specific operation in the code.

plt.plot(history.history['val_accuracy'], label='Validation Accuracy') → Performs a specific operation in the code.

plt.title('Model Accuracy') → Performs a specific operation in the code.

plt.xlabel('Epoch') → Performs a specific operation in the code.

plt.ylabel('Accuracy') → Performs a specific operation in the code.

plt.legend(loc='upper left') → Performs a specific operation in the code.

plt.show() → Performs a specific operation in the code.

Code Cell 42

Plot training & validation loss values → This is a comment explaining the following code.

plt.figure(figsize=(10, 5)) → Performs a specific operation in the code.

plt.plot(history.history['loss'], label='Train Loss') → Performs a specific operation in the code.

plt.plot(history.history['val_loss'], label='Validation Loss') → Performs a specific operation in the code.

plt.title('Model Loss') → Performs a specific operation in the code.

plt.xlabel('Epoch') → Performs a specific operation in the code.

plt.ylabel('Loss') → Performs a specific operation in the code.

plt.legend(loc='upper left') → Performs a specific operation in the code.

plt.show() → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Code Cell 43

y_test_encoded.shape → Performs a specific operation in the code.

Code Cell 44

model = load_model('best_model2.keras') → Defines or uses machine learning libraries like Keras or TensorFlow.

Code Cell 45

test_loss, test_accuracy = model.evaluate(X_test, y_test_encoded) # Use y_test_encoded → Performs a specific operation in the code.

train_loss, train_accuracy = model.evaluate(X_train, y_train_encoded) # Use y_test_encoded → Performs a specific operation in the code.

print(f'Train Accuracy: {train_accuracy * 100:.2f}%') → Prints values or output to the console.

print(f'Test Accuracy: {test_accuracy * 100:.2f}%') → Prints values or output to the console.

Code Cell 46

train_accuracies =[] → Performs a specific operation in the code.

train_losses =[] → Performs a specific operation in the code.

test_accuracies =[] → Performs a specific operation in the code.

test_losses =[] → Performs a specific operation in the code.

Code Cell 47

train_accuracies.append(train_accuracy) → Performs a specific operation in the code.

train_losses.append(train_loss) → Performs a specific operation in the code.

test_accuracies.append(test_accuracy) → Performs a specific operation in the code.

test_losses.append(test_loss) → Performs a specific operation in the code.

Code Cell 48

from sklearn.metrics import confusion_matrix → Imports specific parts (functions, classes) of a library.

from seaborn import heatmap → Imports specific parts (functions, classes) of a library.

→ Performs a specific operation in the code.

heatmap(confusion_matrix(np.argmax(model.predict(X_test),axis=1),y_test_encoded),annot=True) → Performs a specific operation in the code.

Code Cell 49

model.save_weights('models/processed_spetro_97.weights.h5') → This is a comment explaining the following code.

Code Cell 50

import numpy as np → This line imports a library or module.

from sklearn.model_selection import KFold → Imports specific parts (functions, classes) of a library.

from tensorflow.keras.callbacks import ModelCheckpoint → Imports specific parts (functions, classes) of a library.

from tensorflow.keras.optimizers import Adam → Imports specific parts (functions, classes) of a library.

from tensorflow.keras.models import load_model → Imports specific parts (functions, classes) of a library.

from tensorflow.keras.applications import (→ Imports specific parts (functions, classes) of a library.

VGG16, VGG19, → Performs a specific operation in the code.

ResNet50, ResNet101, ResNet152, → Performs a specific operation in the code.

EfficientNetB0, EfficientNetB1, EfficientNetB2, → Performs a specific operation in the code.

DenseNet121, DenseNet169, DenseNet201, → Performs a specific operation in the code.

InceptionV3, InceptionResNetV2, → Performs a specific operation in the code.

MobileNet, MobileNetV2, → Performs a specific operation in the code.

Xception → Performs a specific operation in the code.

) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Dictionary to store models and their names for easy iteration (16 models) → This is a comment explaining the following code.

pretrained_models = { → Performs a specific operation in the code.

'VGG16': VGG16, → Performs a specific operation in the code.

'VGG19': VGG19, → Performs a specific operation in the code.

'ResNet50': ResNet50, → Performs a specific operation in the code.

'ResNet101': ResNet101, → Performs a specific operation in the code.

'ResNet152': ResNet152, → Performs a specific operation in the code.

'EfficientNetB0': EfficientNetB0, → Performs a specific operation in the code.

'EfficientNetB1': EfficientNetB1, → Performs a specific operation in the code.

'EfficientNetB2': EfficientNetB2, → Performs a specific operation in the code.

'DenseNet121': DenseNet121, → Performs a specific operation in the code.

'DenseNet169': DenseNet169, → Performs a specific operation in the code.

'DenseNet201': DenseNet201, → Performs a specific operation in the code.

'InceptionV3': InceptionV3, → Performs a specific operation in the code.

'InceptionResNetV2': InceptionResNetV2, → Performs a specific operation in the code.

'MobileNet': MobileNet, → Performs a specific operation in the code.

'MobileNetV2': MobileNetV2, → Performs a specific operation in the code.

'Xception': Xception → Performs a specific operation in the code.

} → Performs a specific operation in the code.

Code Cell 51

def create_pretrained_model(model_class, input_shape, num_classes): → Performs a specific operation in the code.

base_model = model_class(weights='imagenet', include_top=False, input_shape=input_shape) → Performs a specific operation in the code.

base_model.trainable = False # Freeze the base model layers → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Add custom layers on top of the base model → This is a comment explaining the following code.

model = Sequential([→ Performs a specific operation in the code.

base_model, → Performs a specific operation in the code.

layers.Flatten(), → Performs a specific operation in the code.

layers.Dense(128, activation='relu'), → Performs a specific operation in the code.

layers.Dropout(0.5), → Performs a specific operation in the code.

layers.Dense(num_classes, activation='softmax') → Performs a specific operation in the code.

]) → Performs a specific operation in the code.

return model → Returns a value or result from a function.

Code Cell 52

Set up K-fold cross-validation → This is a comment explaining the following code.

k_folds = 5 # Number of folds for cross-validation → Performs a specific operation in the code.

kf = KFold(n_splits=k_folds, shuffle=True, random_state=42) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Dictionary to store cross-validation results for each model → This is a comment explaining the following code.

cv_results = {model_name: [] for model_name in pretrained_models.keys()} → Iterates over a sequence or collection.

→ Performs a specific operation in the code.

models = {} → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Run K-fold CV for each model → This is a comment explaining the following code.

for model_name, model_class in pretrained_models.items(): → Iterates over a sequence or collection.

print(f"Running {k_folds}-fold CV for {model_name}...") → Prints values or output to the console.

→ Performs a specific operation in the code.

fold_no = 1 → Performs a specific operation in the code.

for train_index, test_index in kf.split(X_train): → Iterates over a sequence or collection.

Split data → This is a comment explaining the following code.

X_train_fold, X_val_fold = X_train[train_index], X_train[test_index] → Performs a specific operation in the code.

y_train_fold, y_val_fold = y_train_encoded[train_index], y_train_encoded[test_index] → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Create a new instance of the model → This is a comment explaining the following code.

model = create_pretrained_model(model_class, input_shape, num_classes) → Performs a specific operation in the code.

model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy']) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Train the model on this fold → This is a comment explaining the following code.

checkpoint = ModelCheckpoint(f'{model_name}_fold_{fold_no}.keras', monitor='val_accuracy', save_best_only=True) → Defines or uses machine learning libraries like Keras or TensorFlow.

history = model.fit(→ Performs a specific operation in the code.

X_train_fold, y_train_fold, → Performs a specific operation in the code.

epochs=10, # You can increase this for better results → Iterates over a sequence or collection.

batch_size=32, → Performs a specific operation in the code.

validation_data=(X_val_fold, y_val_fold), → Performs a specific operation in the code.

callbacks=[checkpoint], → Performs a specific operation in the code.

verbose=1 → Performs a specific operation in the code.

) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

models[model_name] = model → Performs a specific operation in the code.

Evaluate the best model for this fold on the validation set → This is a comment explaining the following code.

best_model = load_model(f'{model_name}_fold_{fold_no}.keras') → Defines or uses machine learning libraries like Keras or TensorFlow.

_, accuracy = best_model.evaluate(X_val_fold, y_val_fold, verbose=0) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

print(f"{model_name} - Fold {fold_no} Validation Accuracy: {accuracy * 100:.2f}%") → Prints values or output to the console.

cv_results[model_name].append(accuracy) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

fold_no += 1 → Performs a specific operation in the code.

Code Cell 53

Calculate mean and standard deviation of accuracies for each model → This is a comment explaining the following code.

for model_name, accuracies in cv_results.items(): → Iterates over a sequence or collection.

mean_accuracy = np.mean(accuracies) → Performs a specific operation in the code.

std_accuracy = np.std(accuracies) → Performs a specific operation in the code.

print(f"\n{model_name} - Mean CV Accuracy: {mean_accuracy * 100:.2f}%") → Prints values or output to the console.

print(f"{model_name} - Std CV Accuracy: {std_accuracy * 100:.2f}%") → Prints values or output to the console.

Code Cell 54

import numpy as np → This line imports a library or module.

→ Performs a specific operation in the code.

Dictionary to store evaluation results → This is a comment explaining the following code.

test_results = {} → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Open a file to write the output → This is a comment explaining the following code.

with open("test_results_output.txt", "w") as file: → Performs a specific operation in the code.

Loop over each saved model, load it, and evaluate on the test set → This is a comment explaining the following code.

for model_name, model in models.items(): → Iterates over a sequence or collection.

print(f"Evaluating model: {model_name}") → Prints values or output to the console.

→ Performs a specific operation in the code.

List to store test accuracies across folds → This is a comment explaining the following code.

fold_accuracies = [] → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Evaluate on the test set → This is a comment explaining the following code.

_, test_accuracy = model.evaluate(X_test, y_test_encoded, verbose=1) → Performs a specific operation in the code.

fold_accuracies.append(test_accuracy) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Print to console and save to file → This is a comment explaining the following code.

print(f"{model_name} Test Accuracy: {test_accuracy * 100:.2f}%") → Prints values or output to the console.

file.write(f"{model_name} Test Accuracy: {test_accuracy * 100:.2f}%\n") → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Calculate mean and standard deviation of test accuracies → This is a comment explaining the following code.

mean_accuracy = np.mean(fold_accuracies) → Performs a specific operation in the code.

std_accuracy = np.std(fold_accuracies) → Performs a specific operation in the code.

test_results[model_name] = { → Performs a specific operation in the code.

"mean_accuracy": mean_accuracy, → Performs a specific operation in the code.

"std_accuracy": std_accuracy → Performs a specific operation in the code.

} → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Print and write mean and std accuracies → This is a comment explaining the following code.

print(f"\n{model_name} - Test Accuracy: {mean_accuracy * 100:.2f}%") → Prints values or output to the console.

print(f"{model_name} - Std Test Accuracy: {std_accuracy * 100:.2f}%\n") → Prints values or output to the console.

→ Performs a specific operation in the code.

file.write(f"{model_name} - Test Accuracy: {mean_accuracy * 100:.2f}%\n") → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Code Cell 55

for model_name, accuracies in cv_results.items(): → Iterates over a sequence or collection.

print(f"Max accuracy for {model_name}: {np.max(accuracies)}") → Prints values or output to the console.

print(f"The index for max accuracy for {model_name}: {np.argmax(accuracies)}") → Prints values or output to the console.

print() → Prints values or output to the console.

Code Cell 56

import numpy as np → This line imports a library or module.

→ Performs a specific operation in the code.

Open a file to write the output → This is a comment explaining the following code.

with open("cv_results_output.txt", "w") as file: → Performs a specific operation in the code.

for model_name, accuracies in cv_results.items(): → Iterates over a sequence or collection.

max_accuracy = np.max(accuracies) → Performs a specific operation in the code.

max_index = np.argmax(accuracies) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Write the results to the file → This is a comment explaining the following code.

file.write(f"Max accuracy for {model_name}: {max_accuracy}\n") → Performs a specific operation in the code.

file.write(f"The index for max accuracy for {model_name}: {max_index}\n") → Iterates over a sequence or collection.

file.write("\n") → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Code Cell 57

cv_results → Performs a specific operation in the code.

Code Cell 58

import numpy as np → This line imports a library or module.

from tensorflow.keras.models import load_model → Imports specific parts (functions, classes) of a library.

from sklearn.linear_model import LogisticRegression → Imports specific parts (functions, classes) of a library.

from sklearn.metrics import accuracy_score → Imports specific parts (functions, classes) of a library.

from sklearn.ensemble import VotingClassifier → Imports specific parts (functions, classes) of a library.

from sklearn.model_selection import train_test_split → Imports specific parts (functions, classes) of a library.

→ Performs a specific operation in the code.

Code Cell 59

Load the pretrained models → This is a comment explaining the following code.

models = { → Performs a specific operation in the code.

'VGG16': load_model(r'models weights\VGG16_fold_3.keras'), → Defines or uses machine learning libraries like Keras or TensorFlow.

'DenseNet121': load_model(r'models weights\DenseNet121_fold_1.keras'), → Defines or uses machine learning libraries like Keras or TensorFlow.

'MobileNet':load_model(r'models weights\MobileNet_fold_3.keras'), → Defines or uses machine learning libraries like Keras or TensorFlow.

'MobileNetV2':load_model(r'models weights\MobileNetV2_fold_3.keras'), → Defines or uses machine learning libraries like Keras or TensorFlow.

'DenseNet169':load_model(r'models weights\DenseNet169_fold_3.keras'), → Defines or uses machine learning libraries like Keras or TensorFlow.

'DenseNet201':load_model(r'models weights\DenseNet201_fold_3.keras') → Defines or uses machine learning libraries like Keras or TensorFlow.

} → Performs a specific operation in the code.

→ Performs a specific operation in the code.

Code Cell 60

def soft_voting_ensemble(models, X): → Performs a specific operation in the code.

""" → Performs a specific operation in the code.

Apply soft voting by averaging probabilities from all models. → Performs a specific operation in the code.

""" → Performs a specific operation in the code.

Gather predictions → This is a comment explaining the following code.

predictions = [model.predict(X) for model in models.values()] → Iterates over a sequence or collection.

avg_predictions = np.mean(predictions, axis=0) → Performs a specific operation in the code.

final_predictions = np.argmax(avg_predictions, axis=1) → Performs a specific operation in the code.

return final_predictions → Returns a value or result from a function.

→ Performs a specific operation in the code.

Predict with soft voting ensemble → This is a comment explaining the following code.

soft_voting_predictions = soft_voting_ensemble(models, X_test) → Performs a specific operation in the code.

soft_voting_accuracy = accuracy_score(y_test_encoded, soft_voting_predictions) → Performs a specific operation in the code.

print(f"Soft Voting Test Accuracy: {soft_voting_accuracy * 100:.2f}%") → Prints values or output to the console.

→ Performs a specific operation in the code.

Code Cell 61

from sklearn.metrics import confusion_matrix → Imports specific parts (functions, classes) of a library.

from seaborn import heatmap → Imports specific parts (functions, classes) of a library.

→ Performs a specific operation in the code.

heatmap(confusion_matrix(soft_voting_predictions,y_test_encoded),annot=True) → Performs a specific operation in the code.

Code Cell 62

Get predictions from base models on the training data for stacking → Iterates over a sequence or collection.

def get_stacking_data(models, X): → Performs a specific operation in the code.

""" → Performs a specific operation in the code.

Get base model predictions to use as new features for stacking. → Iterates over a sequence or collection.

""" → Performs a specific operation in the code.

stacking_data = [] → Performs a specific operation in the code.

for model in models.values(): → Iterates over a sequence or collection.

stacking_data.append(model.predict(X)) → Performs a specific operation in the code.

Stack along columns → This is a comment explaining the following code.

return np.concatenate(stacking_data, axis=1) → Returns a value or result from a function.

→ Performs a specific operation in the code.

Generate stacking data → This is a comment explaining the following code.

stacking_train_data = get_stacking_data(models, X_train) → Performs a specific operation in the code.

stacking_test_data = get_stacking_data(models, X_test) → Performs a specific operation in the code.

→ Performs a specific operation in the code.

→ Performs a specific operation in the code.

Code Cell 63

Define and train a meta-model (e.g., Logistic Regression) → This is a comment explaining the following code.

meta_model = LogisticRegression(max_iter=1000) → Performs a specific operation in the code.

meta_model.fit(stacking_train_data, y_train_encoded) → Performs a specific operation in the code.

Predict with stacking ensemble → This is a comment explaining the following code.

stacking_predictions = meta_model.predict(stacking_test_data) → Performs a specific operation in the code.

Code Cell 64

→ Performs a specific operation in the code.

stacking_accuracy = accuracy_score(y_test_encoded, stacking_predictions) → Performs a specific operation in the code.

print(f"Stacking Test Accuracy: {stacking_accuracy * 100:.2f}%") → Prints values or output to the console.

Code Cell 65

from sklearn.metrics import confusion_matrix → Imports specific parts (functions, classes) of a library.

from seaborn import heatmap → Imports specific parts (functions, classes) of a library.

→ Performs a specific operation in the code.

heatmap(confusion_matrix(stacking_predictions,y_test_encoded),annot=True) → Performs a specific operation in the code.