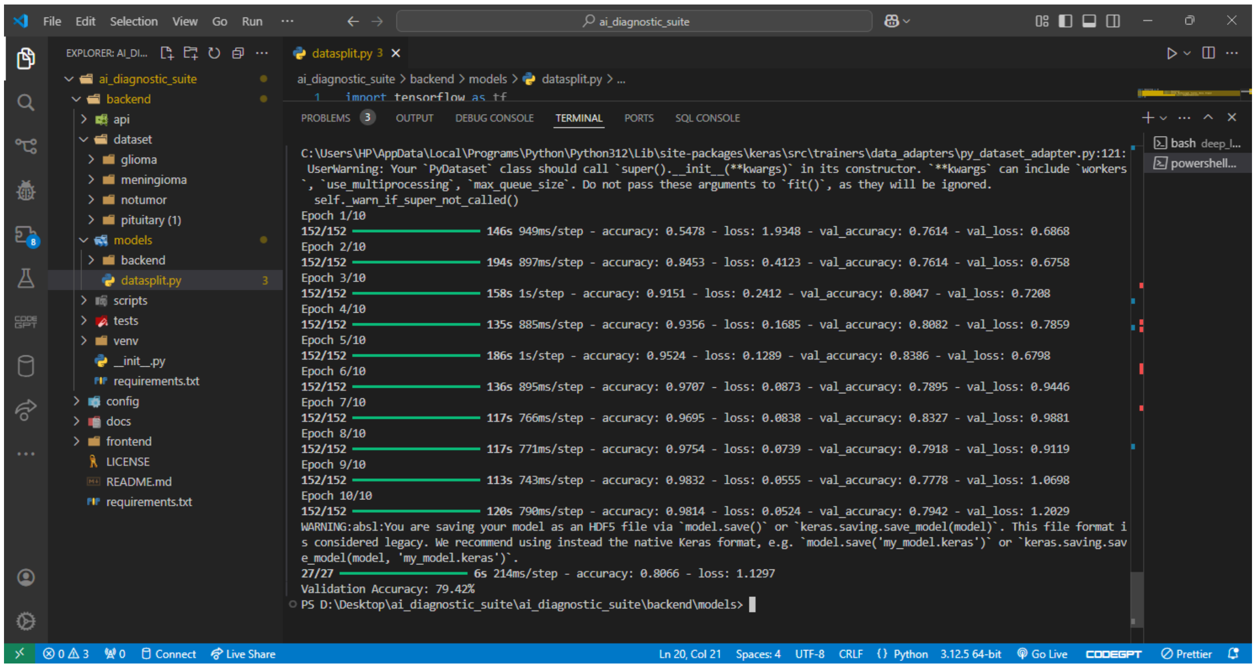
Data Spliting validation score & code implementation:



```
C:\Users\HP\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121:
 UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers
`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 146s 949ms/step - accuracy: 0.5478 - loss: 1.9348 - val_accuracy: 0.7614 - val_loss: 0.6868
Epoch 2/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 194s 897ms/step - accuracy: 0.8453 - loss: 0.4123 - val_accuracy: 0.7614 - val_loss: 0.6758
Epoch 3/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 158s 1s/step - accuracy: 0.9151 - loss: 0.2412 - val_accuracy: 0.8047 - val_loss: 0.7208
Epoch 4/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 135s 885ms/step - accuracy: 0.9356 - loss: 0.1685 - val_accuracy: 0.8082 - val_loss: 0.7859
Epoch 5/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 186s 1s/step - accuracy: 0.9524 - loss: 0.1289 - val_accuracy: 0.8386 - val_loss: 0.6798
Epoch 6/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 136s 895ms/step - accuracy: 0.9707 - loss: 0.0873 - val_accuracy: 0.7895 - val_loss: 0.9446
Epoch 7/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 117s 766ms/step - accuracy: 0.9695 - loss: 0.0838 - val_accuracy: 0.8327 - val_loss: 0.9881
Epoch 8/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 117s 771ms/step - accuracy: 0.9754 - loss: 0.0739 - val_accuracy: 0.7918 - val_loss: 0.9119
Epoch 9/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 113s 743ms/step - accuracy: 0.9832 - loss: 0.0555 - val_accuracy: 0.7778 - val_loss: 1.0698
Epoch 10/10
152/152 ━━━━━━━━━━━━━━━━━━━━ 120s 790ms/step - accuracy: 0.9814 - loss: 0.0524 - val_accuracy: 0.7942 - val_loss: 1.2029
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format i
s considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.sav
e_model(model, 'my_model.keras')`.
27/27 ━━━━━━━━━━━━━━━━━━━━ 6s 214ms/step - accuracy: 0.8066 - loss: 1.1297
Validation Accuracy: 79.42%
PS D:\Desktop\ai_diagnostic_suite\ai_diagnostic_suite\backend\models>
```

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os

# Set dataset path
data_dir = os.path.join("..", "dataset")  # Adjusted relative path
image_size = (224, 224)
batch_size = 32
print("Dataset Path:", os.path.abspath(data_dir))
print("Directory Exists:", os.path.exists(data_dir))

# Image Data Augmentation
datagen = ImageDataGenerator(
    rescale=1.0 / 255.0,
    validation_split=0.15  # 15% for validation, 85% for training
)

# Load Training Data
train_data = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

# Load Validation Data
val_data = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

# Define CNN Model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(train_data.class_indices), activation='softmax')
])

# Compile Model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Train Model
history = model.fit(train_data, validation_data=val_data, epochs=10)

# Save Model
model.save("backend/brain_tumor_baseline_model.h5")

# Evaluate Model
loss, accuracy = model.evaluate(val_data)
```

```python
print(f"Validation Accuracy: {accuracy * 100:.2f}%")
```