Hello Feature Class example: using the feature classes to calculate features

This example shows how to use the Radiomics package to directly instantiate the feature classes for feature extraction. Note that this is not the intended standard use. For an example on the standard use with feature extractor, see the `helloRadiomics` example.

In [ ]:
```
from __future__ import print_function
import os
import collections
import SimpleITK as sitk
import numpy
import six
import radiomics
from radiomics import firstorder, glcm, imageoperations, shape, glrl
m, glszm
```

Getting the test case

Test cases can be downloaded to temporary files. This is handled by the `radiomics.getTestCase()` function, which checks if the requested test case is available and if not, downloads it. It returns a tuple with the location of the image and mask of the requested test case, or (None, None) if it fails.

Alternatively, if the data is available somewhere locally, this directory can be passed as a second argument to `radiomics.getTestCase()` . If that directory does not exist or does not contain the testcase, functionality reverts to default and tries to download the test data.

If getting the test case fails, PyRadiomics will log an error explaining the cause.

In [ ]:
```
imageName, maskName = radiomics.getTestCase('brain1')

if imageName is None or maskName is None:  # Something went wrong, i
n this case PyRadiomics will also log an error
    raise Exception('Error getting testcase!')  # Raise exception to
prevent cells below from running in case of "run all"
```

In [ ]:
```
image = sitk.ReadImage(imageName)
mask = sitk.ReadImage(maskName)
```

Preprocess the image

Extraction Settings

In [ ]:
```
settings = {}
settings['binWidth'] = 25
settings['resampledPixelSpacing'] = None
# settings['resampledPixelSpacing'] = [3, 3, 3]  # This is an exampl
e for defining resampling (voxels with size 3x3x3mm)
settings['interpolator'] = 'sitkBSpline'
settings['verbose'] = True
```

If enabled, resample the image

In [ ]:
```
# Resample if necessary
interpolator = settings.get('interpolator')
resampledPixelSpacing = settings.get('resampledPixelSpacing')
if interpolator is not None and resampledPixelSpacing is not None:
  image, mask = imageoperations.resampleImage(image, mask, **setting
s)
```

Calculate features using original image

In [ ]:
```
# Crop the image
# bb is the bounding box, upon which the image and mask are cropped
bb, correctedMask = imageoperations.checkMask(image, mask, label=1)
if correctedMask is not None:
    mask = correctedMask
croppedImage, croppedMask = imageoperations.cropToTumorMask(image, m
ask, bb)
```

Calculate Firstorder features

In [ ]:
```
firstOrderFeatures = firstorder.RadiomicsFirstOrder(croppedImage, cr
oppedMask, **settings)

# Set the features to be calculated
firstOrderFeatures.enableFeatureByName('Mean', True)
# firstOrderFeatures.enableAllFeatures()
```

In [ ]:
```
# Print out the docstrings of the enabled features
print('Will calculate the following first order features: ')
for f in firstOrderFeatures.enabledFeatures.keys():
  print(f)
  print(getattr(firstOrderFeatures, 'get%sFeatureValue' % f).__doc_
_)
```

In [ ]:
```
# Calculate the features and print(out result)
print('Calculating first order features...',)
result = firstOrderFeatures.execute()
print('done')

print('Calculated first order features: ')
for (key, val) in six.iteritems(result):
  print('  ', key, ':', val)
```

Calculate Shape Features

In [ ]:
```
shapeFeatures = shape.RadiomicsShape(croppedImage, croppedMask, **se
ttings)

# Set the features to be calculated
# shapeFeatures.enableFeatureByName('Volume', True)
shapeFeatures.enableAllFeatures()
```

In [ ]:
```
# Print out the docstrings of the enabled features
print('Will calculate the following shape features: ')
for f in shapeFeatures.enabledFeatures.keys():
  print(f)
  print(getattr(shapeFeatures, 'get%sFeatureValue' % f).__doc__)
```

In [ ]:
```
# Calculate the features and print(out result)
print('Calculating shape features...',)
result = shapeFeatures.execute()
print('done')

print('Calculated shape features: ')
for (key, val) in six.iteritems(result):
  print('  ', key, ':', val)
```

Calculate GLCM Features

In [ ]:
```
glcmFeatures = glcm.RadiomicsGLCM(croppedImage, croppedMask, **setti
ngs)

# Set the features to be calculated
# glcmFeatures.enableFeatureByName('SumEntropy', True)
glcmFeatures.enableAllFeatures()
```

In [ ]:
```
# Print out the docstrings of the enabled features
print('Will calculate the following GLCM features: ')
for f in glcmFeatures.enabledFeatures.keys():
  print(f)
  print(getattr(glcmFeatures, 'get%sFeatureValue' % f).__doc__)
```

In [ ]:
```
# Calculate the features and print(out result)
print('Calculating GLCM features...',)
result = glcmFeatures.execute()
print('done')

print('Calculated GLCM features: ')
for (key, val) in six.iteritems(result):
  print('  ', key, ':', val)
```

Calculate GLRLM Features

In [ ]:
```
glrlmFeatures = glrlm.RadiomicsGLRLM(croppedImage, croppedMask, **se
ttings)

# Set the features to be calculated
# glrlmFeatures.enableFeatureByName('ShortRunEmphasis', True)
glrlmFeatures.enableAllFeatures()
```

In [ ]:
```
# Print out the docstrings of the enabled features
print('Will calculate the following GLRLM features: ')
for f in glrlmFeatures.enabledFeatures.keys():
  print(f)
  print(getattr(glrlmFeatures, 'get%sFeatureValue' % f).__doc__)
```

In [ ]:
```
# Calculate the features and print(out result)
print('Calculating GLRLM features...',)
result = glrlmFeatures.execute()
print('done')

print('Calculated GLRLM features: ')
for (key, val) in six.iteritems(result):
  print('  ', key, ':', val)
```

Calculate GLSZM Features

In [ ]:
```
glszmFeatures = glszm.RadiomicsGLSZM(croppedImage, croppedMask, **se
ttings)

# Set the features to be calculated
# glszmFeatures.enableFeatureByName('LargeAreaEmphasis', True)
glszmFeatures.enableAllFeatures()
```

In [ ]:
```
# Print out the docstrings of the enabled features
print('Will calculate the following GLSZM features: ')
for f in glszmFeatures.enabledFeatures.keys():
  print(f)
  print(getattr(glszmFeatures, 'get%sFeatureValue' % f).__doc__)
```

In [ ]:
```
# Calculate the features and print(out result)
print('Calculating GLSZM features...',)
result = glszmFeatures.execute()
print('done')

print('Calculated GLSZM features: ')
for (key, val) in six.iteritems(result):
  print('  ', key, ':', val)
```

Calculate Features using Laplacian of Gaussian Filter

Calculating features on filtered images is very similar to calculating features on the original image. All filters in PyRadiomics have the same input and output signature, and there is even one for applying no filter. This enables to loop over a list of requested filters and apply them in the same piece of code. It is applied like this in the execute function in feature extractor. The input for the filters is the image, with additional keywords. If no additional keywords are supplied, the filter uses default values where applicable. It returns a generator object (https://docs.python.org/2/reference/simple_stmts.html?#yield), allowing to define the generators to be applied before the filters functions are actually called.

Calculate Firstorder on LoG filtered images

In [ ]:
```
logFeatures = {}
sigmaValues = [1.0, 3.0, 5.0]
for logImage, imageTypename, inputSettings in imageoperations.getLoG
Image(image, mask, sigma=sigmaValues):
  logImage, croppedMask = imageoperations.cropToTumorMask(logImage,
mask, bb)
  logFirstorderFeatures = firstorder.RadiomicsFirstOrder(logImage, c
roppedMask, **inputSettings)
  logFirstorderFeatures.enableAllFeatures()
  logFeatures[imageTypename] = logFirstorderFeatures.execute()
```

In [ ]:
```
# Show result
for sigma, features in six.iteritems(logFeatures):
  for (key, val) in six.iteritems(features):
    laplacianFeatureName = '%s_%s' % (str(sigma), key)
    print('  ', laplacianFeatureName, ':', val)
```

Calculate Features using Wavelet filter

Calculate Firstorder on filtered images

In [ ]:
```
waveletFeatures = {}
for decompositionImage, decompositionName, inputSettings in imageope
rations.getWaveletImage(image, mask):
  decompositionImage, croppedMask = imageoperations.cropToTumorMask
(decompositionImage, mask, bb)
  waveletFirstOrderFeaturs = firstorder.RadiomicsFirstOrder(decompos
itionImage, croppedMask, **inputSettings)
  waveletFirstOrderFeaturs.enableAllFeatures()

  print('Calculate firstorder features with ', decompositionName)
  waveletFeatures[decompositionName] = waveletFirstOrderFeaturs.exec
ute()
```

In [ ]:
```
# Show result
for decompositionName, features in six.iteritems(waveletFeatures):
  for (key, val) in six.iteritems(features):
    waveletFeatureName = '%s_%s' % (str(decompositionName), key)
    print('  ', waveletFeatureName, ':', val)
```