Overview

In this notebook we load the data and view different images to get a better idea about the challenge we are facing. This is always a very helpful first step. It is also important that you can see and try to make some of your own predictions about the data. If you cannot see differences between the groups it is going to be difficult for a biomarker to capture that (but not necessarily impossible)

In [ ]:
```
# special functions for using pyradiomics
from SimpleITK import GetImageFromArray
import radiomics
from radiomics.featureextractor import RadiomicsFeatureExtractor # T
his module is used for interaction with pyradiomic
import logging
logging.getLogger('radiomics').setLevel(logging.CRITICAL + 1)  # thi
s tool makes a whole TON of log noise
```

Setup the PyRadiomics Code

In [ ]:
```
# Instantiate the extractor
texture_extractor = RadiomicsFeatureExtractor(verbose=False)
texture_extractor.disableAllFeatures()
_text_feat = {ckey: [] for ckey in texture_extractor.featureClassNam
es}
texture_extractor.enableFeaturesByName(**_text_feat)

print('Extraction parameters:\n\t', texture_extractor.settings)
print('Enabled filters:\n\t', texture_extractor.enabledImagetypes)
print('Enabled features:\n\t', texture_extractor.enabledFeatures)
```

In [ ]:
```
import numpy as np # for manipulating 3d images
import pandas as pd # for reading and writing tables
import h5py # for reading the image files
import skimage # for image processing and visualizations
import sklearn # for machine learning and statistical models
import os # help us load files and deal with paths
from pathlib import Path # help manage files
```

Plot Setup Code

Here we setup the defaults to make the plots look a bit nicer for the notebook

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
plt.rcParams["figure.figsize"] = (8, 8)
plt.rcParams["figure.dpi"] = 125
plt.rcParams["font.size"] = 14
plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['DejaVu Sans']
plt.style.use('ggplot')
sns.set_style("whitegrid", {'axes.grid': False})
```

Load all of the images

We start with the training data since we have labels for them and can look in more detail

In [ ]:
```
data_root = Path('..') / 'data_MRI'
all_df = pd.DataFrame({'h5_path': list(data_root.glob('*/*.h5'))})
all_df['file_id'] = all_df['h5_path'].map(lambda x: x.stem)
all_df['training_group'] = all_df['h5_path'].map(lambda x: x.parent.
stem)
all_df['scan_id'] = all_df['file_id'].map(lambda x: int(x.split('_
')[-1]))
all_df.head(5) # show the first 5 lines
```

Read Image

In [ ]:
```
def read_scan(full_scan_path):
    # load the image using hdf5
    with h5py.File(full_scan_path, 'r') as h:
        # [::2, ::4, ::4, 0] downsampling makes it go much faster
        return h['image'][:][:, :, :, 0] # we read the data from the
file
```

Load a Scan

- the data on kaggle are located in a parent folder called input.
- Since the files have been organized into train and test we use the train folder

In [ ]:
```
sample_scan = all_df.iloc[0] # just take the first row
print(sample_scan)
# turn the h5_path into the full path
image_data = read_scan(sample_scan['h5_path'])
print('Image Shape:', image_data.shape)
```

Calculate Radiomic Features

Calculate the radiomic features for the test scan

In [ ]:
```
# we take a mask by just keeping the part of the image greater than
0
plt.imshow(np.sum((image_data>0).astype(float), 0))
```

In [ ]:
```
%%time
results = texture_extractor.execute(GetImageFromArray(image_data),
                                    GetImageFromArray((image_data>0).astype
(np.uint8)))
```

In [ ]:
```
pd.DataFrame([results]).T
```

In [ ]:
```
def calc_radiomics(in_image_data):
    return texture_extractor.execute(GetImageFromArray(in_image_dat
a),
                                     GetImageFromArray((in_image_data>0).asty
pe(np.uint8)))
```

Run over all scans

We use the `.map` function from pandas to calculate the brightness for all the scans

In [ ]:
```
%%time
all_df['radiomics'] = all_df['h5_path'].map(lambda c_filename: calc_
radiomics(read_scan(c_filename)))
```

In [ ]:
```
full_df = pd.DataFrame([dict(**c_row.pop('radiomics'), **c_row) for
_, c_row in all_df.iterrows()])
print(full_df.shape, 'data prepared')
first_cols = all_df.columns[:-1].tolist()
full_df = full_df[first_cols + [c_col for c_col in full_df.columns
                                if c_col not in first_cols]]
# export the whole table
full_df.to_csv('all_radiomics_table.csv', index=False)
full_df.sample(3)
```

Focusing on Interesting Radiomics

In [ ]:
```
# leave out anything that doesn't start with original (just junk fro
m the input)
# also remove shape since it is not very informative
value_feature_names = [c_col for c_col in full_df.columns if (c_col.
startswith('original') and '_shape_' not in c_col)]
print(np.random.choice(value_feature_names, 3), 'of', len(value_feat
ure_names))
```

In [ ]:
```
# make a cleaner vefirst_colson
clean_df = full_df[first_cols + value_feature_names].copy()
clean_df.columns = first_cols + [
    ' '.join(c_col.split('original_')[-1].split('_'))
    for c_col in value_feature_names
]
clean_col_names = clean_df.columns[len(first_cols):]
# fix some of the artifacts from the radiomics tool
obj_cols = clean_df[clean_col_names].select_dtypes(['object']).colum
ns.tolist()
for c_col in obj_cols:
    clean_df[c_col] = clean_df[c_col].map(lambda x: float(x))
clean_df.to_csv('clean_radiomics_table.csv', index=False)
clean_df.sample(3)
```

Show the distribution of different variables

In [ ]:
```
fig, m_axs = plt.subplots(12, 8, figsize=(60, 70))
for c_ax, c_var in zip(m_axs.flatten(), clean_col_names):
    c_ax.hist(clean_df[c_var].values, bins=10)
    c_ax.set_title('\n'.join(c_var.split(' ')))
fig.savefig('all_metrics.png')
```

In [ ]:
```

```