QHCC

Generated by Doxygen 1.12.0

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 demo_QHCC_XGBoost Namespace Reference

The demo file for doing classification on QHCC datasets.

Functions

- auto_feature_select (X0, y0, k=1)

  Feature selection to select the top k features with the highest mutual information values Input:
- experiment_redcap_lab (df, seed)

  Experiments on prediction with tabular data Input:
- experiment_redcap_lab_radiomics (df, df_1, df_2, ids_train=None, ids_test=None, seed=100)

  Experiments on prediction with both tabular data and radiomics features Input:
- main ()

  The main function for performing experiments and printing out cross validation results.

3.1.1 Detailed Description

The demo file for doing classification on QHCC datasets.

Version

v1.0

3.1.2 Function Documentation

3.1.2.1 auto_feature_select()

```
demo_QHCC_XGBoost.auto_feature_select (
              X0,
              y0,
              k = 1)
```

Feature selection to select the top k features with the highest mutual information values Input:

Parameters

X0	dataframe, the tabular data
y0	dataframe, the label data
k	integer, the number of feature elements to be selected Output:

Returns

selected_features: the indexes of the selected features

3.1.2.2 experiment_redcap_lab()

```
demo_QHCC_XGBoost.experiment_redcap_lab (
              df,
              seed)
```

Experiments on prediction with tabular data Input:

Parameters

df	dataframe, the tabular data
seed	random seed Output:

Returns

model: the trained model

ids_train: the patient IDs for the training data

ids_test: the patient IDs for the test data

acc: accuracy

auc: AUC (area under the curve) score

3.1.2.3 experiment_redcap_lab_radiomics()

```
demo_QHCC_XGBoost.experiment_redcap_lab_radiomics (
              df,
              df_1,
              df_2,
              ids_train = None,
              ids_test = None,
              seed = 100)
```

Experiments on prediction with both tabular data and radiomics features Input:

Parameters

df	dataframe, the tabular data
df_1	dataframe, the radiomics features extracted from MRI images
df_2	dataframe, the radiomics features extracted from CT images
ids_train	the patient IDs for the training data
ids_test	the patient IDs for the test data
seed	random seed Output:

Returns

model: the trained model

ids_train: the patient IDs for the training data

ids_test: the patient IDs for the test data

acc: accuracy

auc: AUC (area under the curve) score

3.1.2.4 main()

```
demo_QHCC_XGBoost.main ()
```

The main function for performing experiments and printing out cross validation results.

3.2 extract_QHCC_radiomics_features Namespace Reference

Extracting radiomics features from medical images with given masks.

Variables

- extractor = featureextractor.RadiomicsFeatureExtractor()

  Initialize the PyRadiomics feature extractor.
- str mri_dir = "./data/L1_files"

  File paths: MRI image and Mask file directories.
- str mask_dir = "./data/L1_files_mask"
- mri_files = sorted([f for f in os.listdir(mri_dir) if f.endswith(".nii")])

  Get lists of MRI and Mask files default file format: ".nii".
- mask_files = sorted([f for f in os.listdir(mask_dir) if f.endswith(".nii")])
- radiomics_data = pd.DataFrame()

  Initialize a DataFrame to store the features.
- mri_path = os.path.join(mri_dir, mri_file)

  Batch processing.
- mask_path = os.path.join(mask_dir, mask_file)
- image = sitk.ReadImage(mri_path)

  Load MRI image and Mask.
- mask = sitk.ReadImage(mask_path)
- temp_mri_path = os.path.join(mri_dir, f"temp_{mri_file}")

  Save the adjusted MRI image.
- features = extractor.execute(temp_mri_path, mask_path)

  Extract features.
- dict flat_features = {key: value for key, value in features.items()}

  Flatten features and convert them to a DataFrame row
  Use the filename as the ID.
- feature_row = pd.DataFrame([flat_features], index=[mri_file.split('.')[0]])
- str output_path = "radiomics_features.csv"

  Save Radiomics features to CSV.
- index

3.2.1 Detailed Description

Extracting radiomics features from medical images with given masks.

Version

v1.0

3.2.2 Variable Documentation

3.2.2.1 extractor

```
extract_QHCC_radiomics_features.extractor = featureextractor.RadiomicsFeatureExtractor()
```

Initialize the PyRadiomics feature extractor.

3.2.2.2 feature_row

```
extract_QHCC_radiomics_features.feature_row = pd.DataFrame([flat_features], index=[mri_file.↩
split('.')[0]])
```

3.2.2.3 features

```
extract_QHCC_radiomics_features.features = extractor.execute(temp_mri_path, mask_path)
```

Extract features.

3.2.2.4 flat_features

```
dict extract_QHCC_radiomics_features.flat_features = {key:  value for key, value in features.↩
items()}
```

Flatten features and convert them to a DataFrame row
Use the filename as the ID.

3.2.2.5 image

```
extract_QHCC_radiomics_features.image = sitk.ReadImage(mri_path)
```

Load MRI image and Mask.

3.2.2.6 index

```
extract_QHCC_radiomics_features.index
```

3.2.2.7 mask

```
extract_QHCC_radiomics_features.mask = sitk.ReadImage(mask_path)
```

3.2.2.8 mask_dir

```
str extract_QHCC_radiomics_features.mask_dir = "./data/L1_files_mask"
```

3.2.2.9 mask_files

```
extract_QHCC_radiomics_features.mask_files = sorted([f for f in os.listdir(mask_dir) if f.↩
endswith(".nii")])
```

3.2.2.10 mask_path

```
extract_QHCC_radiomics_features.mask_path = os.path.join(mask_dir, mask_file)
```

3.2.2.11 mri_dir

```
str extract_QHCC_radiomics_features.mri_dir = "./data/L1_files"
```

File paths: MRI image and Mask file directories.

3.2.2.12 mri_files

```
extract_QHCC_radiomics_features.mri_files = sorted([f for f in os.listdir(mri_dir) if f.↩
endswith(".nii")])
```

Get lists of MRI and Mask files default file format: ".nii".

3.2.2.13 mri_path

```
extract_QHCC_radiomics_features.mri_path = os.path.join(mri_dir, mri_file)
```

Batch processing.

3.2.2.14 output_path

```
extract_QHCC_radiomics_features.output_path = "radiomics_features.csv"
```

Save Radiomics features to CSV.

3.2.2.15 radiomics_data

```
extract_QHCC_radiomics_features.radiomics_data = pd.DataFrame()
```

Initialize a DataFrame to store the features.

Add to the main DataFrame.

3.2.2.16 temp_mri_path

```
extract_QHCC_radiomics_features.temp_mri_path = os.path.join(mri_dir, f"temp_{mri_file}")
```

Save the adjusted MRI image.

Chapter 4

File Documentation

4.1 demo_QHCC_XGBoost.py File Reference

Namespaces

- namespace demo_QHCC_XGBoost

  The demo file for doing classification on QHCC datasets.

Functions

- demo_QHCC_XGBoost.auto_feature_select (X0, y0, k=1)

  Feature selection to select the top k features with the highest mutual information values Input:
- demo_QHCC_XGBoost.experiment_redcap_lab (df, seed)

  Experiments on prediction with tabular data Input:
- demo_QHCC_XGBoost.experiment_redcap_lab_radiomics (df, df_1, df_2, ids_train=None, ids_test=None, seed=100)

  Experiments on prediction with both tabular data and radiomics features Input:
- demo_QHCC_XGBoost.main ()

  The main function for performing experiments and printing out cross validation results.

4.2 extract_QHCC_radiomics_features.py File Reference

Namespaces

- namespace extract_QHCC_radiomics_features

  Extracting radiomics features from medical images with given masks.

Variables

- extract_QHCC_radiomics_features.extractor = featureextractor.RadiomicsFeatureExtractor()

    Initialize the PyRadiomics feature extractor.
- str extract_QHCC_radiomics_features.mri_dir = "./data/L1_files"

    File paths: MRI image and Mask file directories.
- str extract_QHCC_radiomics_features.mask_dir = "./data/L1_files_mask"
- extract_QHCC_radiomics_features.mri_files = sorted([f for f in os.listdir(mri_dir) if f.endswith(".nii")])

    Get lists of MRI and Mask files default file format: ".nii".
- extract_QHCC_radiomics_features.mask_files = sorted([f for f in os.listdir(mask_dir) if f.endswith(".nii")])
- extract_QHCC_radiomics_features.radiomics_data = pd.DataFrame()

    Initialize a DataFrame to store the features.
- extract_QHCC_radiomics_features.mri_path = os.path.join(mri_dir, mri_file)

    Batch processing.
- extract_QHCC_radiomics_features.mask_path = os.path.join(mask_dir, mask_file)
- extract_QHCC_radiomics_features.image = sitk.ReadImage(mri_path)

    Load MRI image and Mask.
- extract_QHCC_radiomics_features.mask = sitk.ReadImage(mask_path)
- extract_QHCC_radiomics_features.temp_mri_path = os.path.join(mri_dir, f"temp_{mri_file}")

    Save the adjusted MRI image.
- extract_QHCC_radiomics_features.features = extractor.execute(temp_mri_path, mask_path)

    Extract features.
- dict extract_QHCC_radiomics_features.flat_features = {key: value for key, value in features.items()}

    Flatten features and convert them to a DataFrame row
    Use the filename as the ID.
- extract_QHCC_radiomics_features.feature_row = pd.DataFrame([flat_features], index=[mri_file.split('.')[0]])
- str extract_QHCC_radiomics_features.output_path = "radiomics_features.csv"

    Save Radiomics features to CSV.
- extract_QHCC_radiomics_features.index