clinical

April 29, 2023

[1]:
```
import pandas as pd
import os
import numpy as np
import ut
import gc
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
```

[2]:
```
def get_metadata(data_path):
    # Clean metadata dataframe
    md = pd.read_csv(os.path.join(data_path, 'ADNI1_Complete_2Yr_3T_4_18_2023.csv'))

    md = md.rename(columns={'Image Data ID': 'Img_ID'})

    md = md.drop(columns=['Modality', 'Description', 'Type', 'Format', 'Downloaded', 'Acq Date'])

    md['Group'] = md['Group'].map({'CN':0, 'MCI':1, 'AD':2})
    md['Sex'] = md['Sex'].map({'F':0, 'M':1})
    md = md.rename(columns = {'Visit':'VISCODE'})

    return md
```

[3]:
```
def clean_data(md):
    cl = pd.read_csv("/scratch/users/neuroimage/conda/data/clinical/ADNIMERGE.csv", low_memory=False, na_values=[-4])
    cl = cl[cl.COLPROT=='ADNI1']

    # could add EXAMDATE for time series
    filter = ['PTID', 'PTEDUCAT', 'PTETHCAT', 'PTRACCAT', 'PTMARRY', 'MMSE', 'AGE', 'VISCODE']
    cl = cl[filter]
    cl = cl.rename(columns={'PTID':'Subject', 'PTEDUCAT':'Educat', 'PTRACCAT':'Race', 'PTETHCAT':'Ethn', 'PTMARRY':'Marry'})

    comb = md.merge(cl, how='left', on=['Subject', 'VISCODE'])
```

```
    comb['MMSE'] = comb['MMSE'].astype(np.int32)
    comb = comb.drop(columns=['VISCODE'])

    # one-hot encoding
    one_hot = pd.get_dummies(comb, columns=['Ethn', 'Race', 'Marry'])

    return one_hot
```

[4]:
```
def full_pipeline(X_matrix):

    df = ut.matrix_to_df(X_matrix)
    X, y = df.pipe(ut.clean_data, md=clean)

    X_train_pca, y_train, X_test_pca, y_test = ut.perform_pca(X,y)

    return X_train_pca, y_train, X_test_pca, y_test
```

[5]:
```
data_path = '/scratch/users/neuroimage/conda/data'
img_path = os.path.join(data_path, 'preprocessed/imgsss')
```

[6]:
```
smt_files_gm, smt_files_wm = ut.get_ordered_files(img_path, "smt")
md = get_metadata(data_path)
clean = clean_data(md)

X_wm, X_gm = ut.imgs_to_matrix(img_path, smt_files_gm, smt_files_wm,␣
 ↪combine=False)
X_comb = ut.imgs_to_matrix(img_path, smt_files_gm, smt_files_wm, combine=True)
```

[7]:
```
X_train_wm, y_train_wm, X_test_wm, y_test_wm = full_pipeline(X_wm)
gc.collect()
X_train_gm, y_train_gm, X_test_gm, y_test_gm = full_pipeline(X_gm)
gc.collect()
X_train_cb, y_train_cb, X_test_cb, y_test_cb = full_pipeline(X_comb)
gc.collect()
```

```
n components:147
n components:160
n components:175
```

[7]:
```
0
```

[8]:
```
base_pca_dir = os.path.join(data_path, 'full_pca_data')
os.makedirs(base_pca_dir, exist_ok=True)

arrs = {'X_train_wm': X_train_wm, 'y_train_wm': y_train_wm,
        'X_test_wm': X_test_wm, 'y_test_wm': y_test_wm,
        'X_train_gm': X_train_gm, 'y_train_gm': y_train_gm,
```

```
        'X_test_gm': X_test_gm, 'y_test_gm': y_test_gm,
        'X_train_cb': X_train_cb, 'y_train_cb': y_train_cb,
        'X_test_cb': X_test_cb, 'y_test_cb': y_test_cb,
       }

# loop through the dictionary and save each dataframe to CSV file
for name, arr in arrs.items():
    np.savetxt(os.path.join(base_pca_dir,f'{name}.csv'), arr, delimiter=',')
```

[9]:
```
logreg_wm_acc = ut.perform_logreg(X_train_wm, y_train_wm, X_test_wm, y_test_wm)
logreg_gm_acc = ut.perform_logreg(X_train_gm, y_train_gm, X_test_gm, y_test_gm)
logreg_cb_acc = ut.perform_logreg(X_train_cb, y_train_cb, X_test_cb, y_test_cb)
```

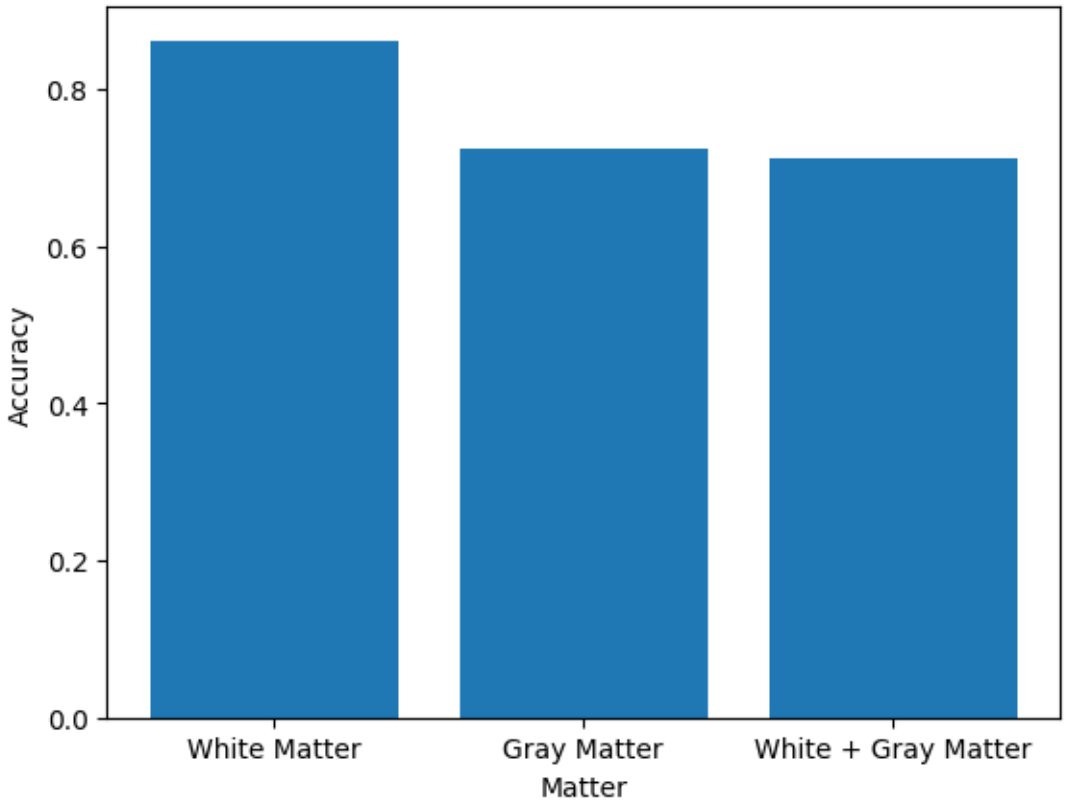
[10]:
```
print(f"Classification accuracy under WM data: {logreg_wm_acc}")
print(f"Classification accuracy under GM data: {logreg_gm_acc}")
print(f"Classification accuracy under Combined data: {logreg_cb_acc}")
```

```
Classification accuracy under WM data: 0.8620689655172413
Classification accuracy under GM data: 0.7241379310344828
Classification accuracy under Combined data: 0.7126436781609196
```

[11]:
```
import matplotlib.pyplot as plt

# Create a bar plot
accuracies = [logreg_wm_acc, logreg_gm_acc, logreg_cb_acc]
labels = ['White Matter', 'Gray Matter', 'White + Gray Matter']
plt.bar(labels, accuracies)

# Add labels and title
plt.xlabel('Matter')
plt.ylabel('Accuracy')
# Show the plot
plt.show()
```



[12]:
```
# define logistic regression model
lr = LogisticRegression(multi_class='multinomial', random_state=10,␣
 ↪penalty=None, max_iter=1000)

# define solvers to test
solvers = ['lbfgs', 'newton-cg', 'sag', 'saga']

# define grid search parameters
param_grid = {'solver': solvers}

# perform grid search
grid_search = GridSearchCV(lr, param_grid=param_grid, cv=5)
grid_search.fit(X_train_wm, y_train_wm)

# get the best model
best_model = grid_search.best_estimator_
print(best_model)

# use the best model to make predictions on the test set
y_preds = best_model.predict(X_test_wm)
```

```python
acc = sum(y_preds == y_test_wm) / len(y_test_wm)
print(f"{best_model} has accuracy: {acc}")
```

```
/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge

/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge
```

```
LogisticRegression(max_iter=1000, multi_class='multinomial', penalty=None,
                   random_state=10)
LogisticRegression(max_iter=1000, multi_class='multinomial', penalty=None,
                   random_state=10) has accuracy: 0.8620689655172413

/scratch/users/neuroimage/conda/venv/lib/python3.11/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning:

The max_iter was reached which means the coef_ did not converge
```