Memory Performance Prediction

221AIG08 한지원

Introduction

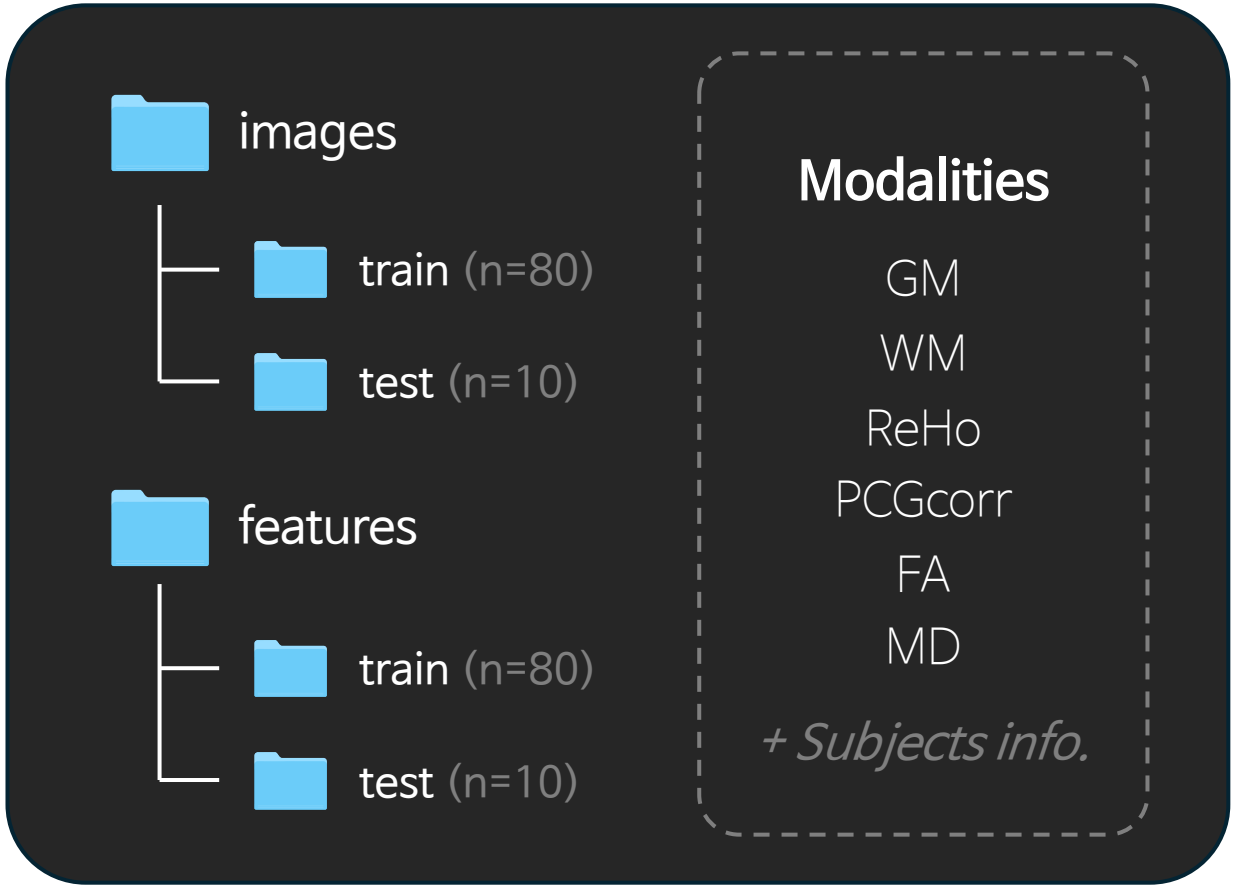
- Goal
  - Designing a machine learning model to predict memory performance scores, using feature data extracted from MRI scans and demographic information.

Data

- Structure
  - Images

    Structural, resting state functional, and diffusion-weighted MRI.
  - Features

    Extracted features, demographic information and memory performance scores.



Data

- Pre-processing
  - Merge the feature data frames considering given modalities.
  - Add the demographic information on the merged data frame.
  - Split the features into predicator variables and response variable.

```python
# Selecting Specific Modalities
modalities = ['GM', 'ReHo', 'MD', 'WM', 'FA', 'PCGcorr']
selected_modalities = {modality: predictors_paths[modality] for modality in modalities if modality in predictors_paths}

# Reading and Renaming Data (except 'ID' column)
def read_and_rename(modality, path):
    df = pd.read_csv(path)
    df = df.rename(columns={label: f"{label}_{modality}" for label in df.columns if label != 'ID'})
    # print('[ '+modality+' ]')
    # print(df.columns)
    # print('\n')
    return df
dfs = [read_and_rename(modality, path) for modality, path in selected_modalities.items()]

# Merging All Data Frames
predictors_df = reduce(lambda left, right: pd.merge(left, right, on='ID'), dfs)
```

Python

```python
#  Response and Confounding Variables
additional_variables = ['Memory', 'Age', 'Sex', 'EducationYear']
df = pd.read_csv(additional_variables_path)
print(df.head())
selected_variables = [variable for variable in additional_variables if variable in df.columns]
additional_variables_df = df[["ID"] + selected_variables]

# Merging Dataframes and Saving to CSV
df = pd.merge(additional_variables_df, predictors_df, on='ID')
# print(df.head())
df.to_csv(train_data_path+'Edited_df_'+'+'.join(modalities) + '.csv', index=False)

# Prepare X and y
X = df.drop(columns=['ID', 'Memory'])   # All variables except 'ID' and  'Memory'
y = df['Memory']    # A response variable, 'Memory'
```

Python

Model

- Feature Selection Method

Filter Method	Wrapper Method	Embedded Method
• Ranking features based on statistical measures • Weak to capture feature interactions	• Iteratively adding or removing features • Computationally expensive	• Assigning weights to features • Part of the model construction process

Model

```python
feature_selection_method = 'Lasso' # 'Lasso' or 'MI'
feature_selection(feature_selection_method)
```

Python

```
Regularization strength hyperparameter for Lasso: 0.061
Selected features by Lasso: 14 out of 327 total features
```

- Feature Selection Method

Filter Method	Wrapper Method	Embedded Method
• Ranking features based on statistical measures • Weak to capture feature interactions	• Iteratively adding or removing features • Computationally expensive	• Assigning weights to features • Part of the model construction process

LASSO (Least Absolute Shrinkage and Selection Operator)

A linear regression technique that prevents overfitting and enhances interpretability. Act as variable selector by setting some weights to zero.

Model

Random Forest	Extremely Randomized Trees
Samples subsets through bootstrapping	Samples the entire dataset
Nodes are split looking at the best split	Randomized node split
Medium Variance	Low Variance
It takes time to find the best node to split on	Faster since node splits are random

- Find the Best Parameter

  Both are types of ensemble learning methods that build multiple decision trees and aggregate their predictions.

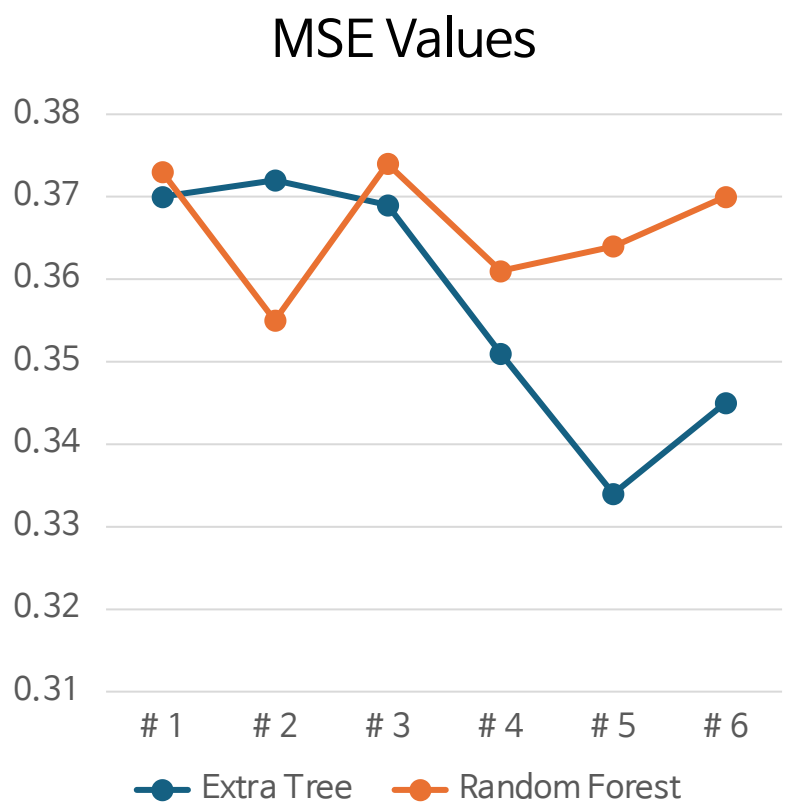
  (Due to time constraints, the XGB method was not utilized in this analysis.)

  - **Extra Tree Regression (ET)**
    - Uses the entire original sample
    - Increases the model's randomness
    - Combines the predictions of all trees
  - **Random Forest Regression (RF)**
    - Uses bootstrap samples of the data
    - Improves the model's robustness
    - Averages the predictions of all trees or by majority voting

Result

- Model Comparison: MSE Values

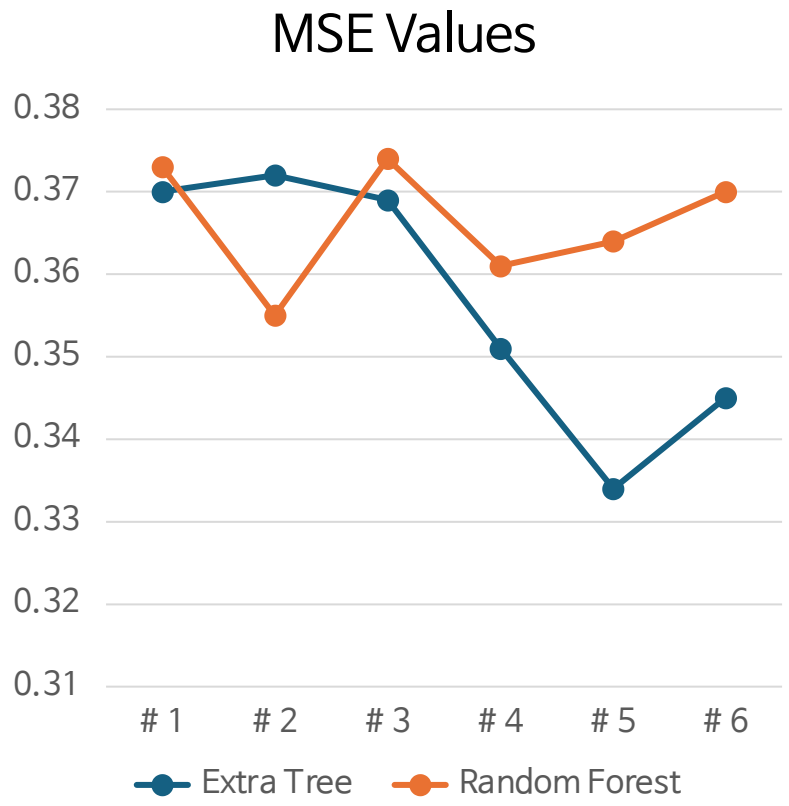
Modalities	Extra Tree (ET)	Random Forest (RF)
GM	0.370	0.373
ReHo+GM	0.372	0.355
MD+ReHo+GM	0.369	0.374
WM+MD+ReHo+GM	0.351	0.361
FA+WM+MD+ReHo+GM	0.334	0.364
PCGcorr+FA+WM+MD+ReHo+GM	0.345	0.370



Result

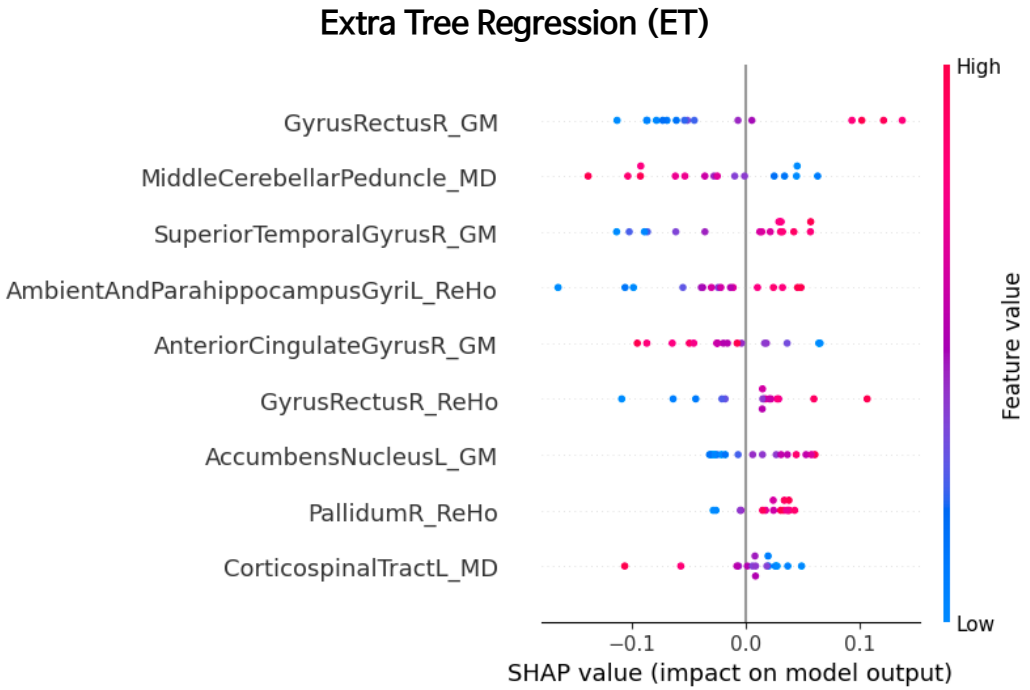
- Model Comparison: MSE Values

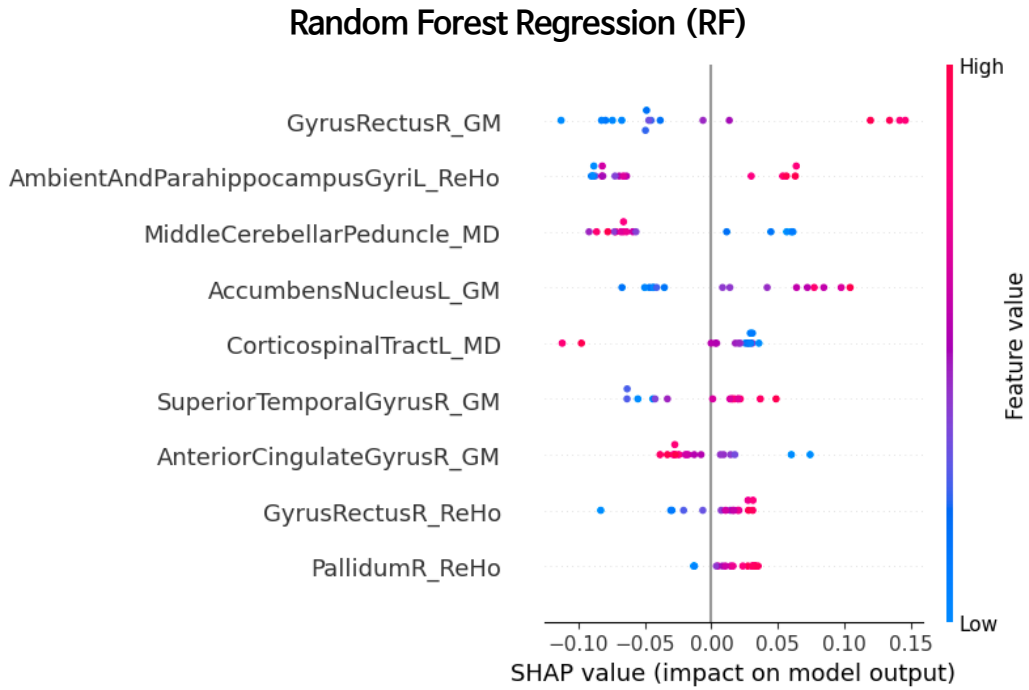
Modalities	Extra Tree (ET)	Random Forest (RF)
GM	0.370	0.373
ReHo+GM	0.372	0.355
MD+ReHo+GM	0.369	0.374
WM+MD+ReHo+GM	0.351	0.361
FA+WM+MD+ReHo+GM	0.334	0.364
PCGcorr+FA+WM+MD+ReHo+GM	0.345	0.370



Result

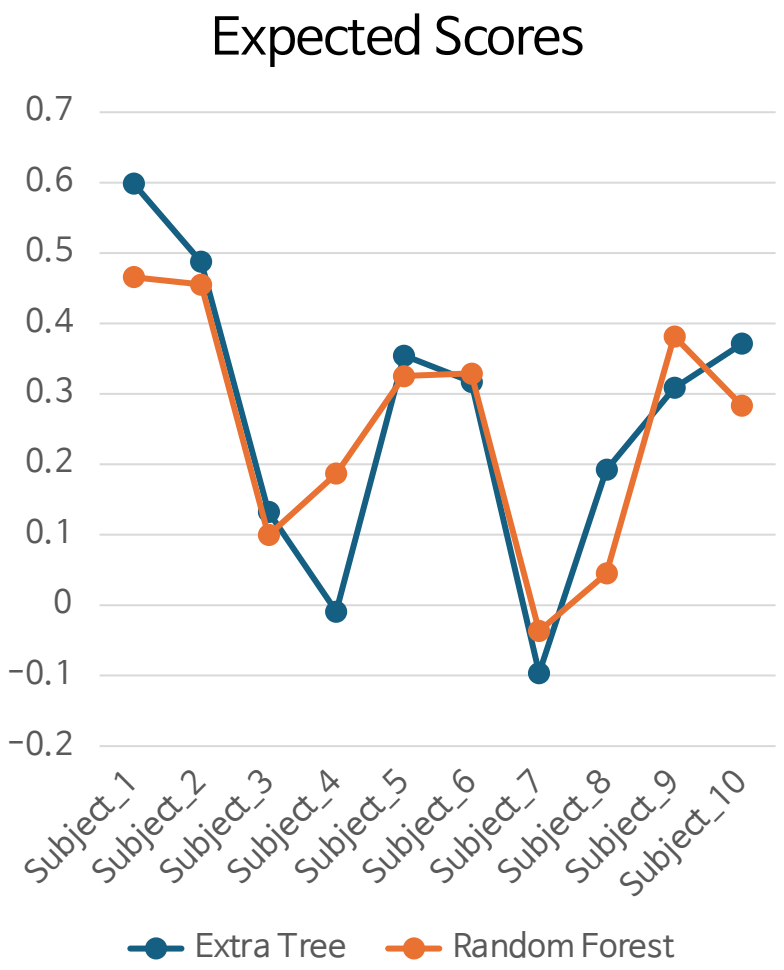
- Important Features
  - In both methods, *GytusRectusR_GM* was identified as the most important feature.





Memory Prediction

- Expected Memory Score
  - Predicted 10 memory scores for the test data, employing two ML methods.
  - Selected modalities are as follows; GM, ReHo, and MD.
  - While there are differences in values, the overall trends are generally similar.



Extra Tree (ET)	Random Forest (RF)
0.5990	0.4657
0.4875	0.4552
0.1328	0.0997
-0.0092	0.1866
0.3546	0.3252
0.3171	0.3291
-0.0962	-0.0367
0.1920	0.0449
0.3087	0.3817
0.3710	0.2837