Machine Learning Analysis using Functional Connectivity Data Notebook:

This Jupyter notebook demonstrates how to use machine learning techniques to predict age from resting-state fMRI (rs-fMRI) data. Below are the key points and steps covered in the notebook:

Overview

The notebook focuses on:

1. **Data Loading**: Loading rs-fMRI data and confounds.
2. **Feature Extraction**: Extracting brain connectivity features using an atlas and calculating correlation matrices.
3. **Machine Learning**: Building a machine learning model to predict age from extracted features using Support Vector Regression (SVR).
4. **Model Evaluation**: Evaluating model performance using cross-validation, hyperparameter tuning, and feature selection.
5. **Visualization**: Visualizing feature importance and brain connectivity.

1. Data Loading

- **rs-fMRI Data**: The dataset consists of 155 subjects' rs-fMRI images, preprocessed using fmriprep.

python

```
data = sorted(glob(os.path.join(wdir,'*.gz')))

confounds = sorted(glob(os.path.join(wdir,'*regressors.tsv')))

len(data)  # 155 subjects
```

2. Feature Extraction

Atlas-Based Feature Extraction

- The MIST atlas with 64 regions of interest (ROIs) is used to extract brain connectivity features from the rs-fMRI data. The NiftiLabelsMasker is applied to extract time series from the ROIs.

python

```
from nilearn.input_data import NiftiLabelsMasker
```

masker = NiftiLabelsMasker(labels_img=atlas_filename, standardize=True, memory='nilearn_cache', verbose=1)

time_series = masker.fit_transform(fmri_filenames, confounds=conf)

- A correlation matrix is computed for each subject's time series to represent functional connectivity between brain regions.

python

from nilearn.connectome **import** ConnectivityMeasure

correlation_measure = ConnectivityMeasure(kind='correlation')

correlation_matrix = correlation_measure.fit_transform([time_series])[0]

Looping Over All Subjects

- A loop iterates over all subjects to extract connectivity features for each subject.

python

all_features = []

for i, sub **in** enumerate(data):

  time_series = masker.fit_transform(sub, confounds=confounds[i])

  correlation_matrix = correlation_measure.fit_transform([time_series])[0]

  all_features.append(correlation_matrix)

3. Machine Learning Model

Support Vector Regression (SVR)

- A Support Vector Regressor (SVR) is used to predict age based on the extracted connectivity features.

python

from sklearn.svm **import** SVR

l_svr = SVR(kernel='linear')

l_svr.fit(X_train, y_train)

Model Evaluation

- The model is evaluated using cross-validation and metrics like R^2 (coefficient of determination) and Mean Absolute Error (MAE).

python

```
from sklearn.metrics import mean_absolute_error

y_pred = cross_val_predict(l_svr, X_train, y_train, cv=10)

acc = r2_score(y_train, y_pred)

mae = mean_absolute_error(y_train, y_pred)
```

Cross-Validation Results

- The model is evaluated across multiple folds of cross-validation to assess its generalization performance.

python

```
for i in range(10):
  print(f'Fold {i} -- Acc = {acc[i]}, MAE = {-mae[i]}')
```

4. Hyperparameter Tuning and Feature Selection

Hyperparameter Tuning

- A grid search is performed to find the optimal hyperparameters (C and epsilon) for the SVR model.

python

```
from sklearn.model_selection import GridSearchCV

param_grid = dict(epsilon=epsilon_range, C=C_range)

grid = GridSearchCV(l_svr, param_grid=param_grid, cv=10)

grid.fit(X_train, y_train_log)
```

Feature Selection

- Feature selection is performed using SelectPercentile, which selects the top X% of features based on univariate tests.

python

```
from sklearn.feature_selection import SelectPercentile, f_regression
```

model = Pipeline([('feature_selection', SelectPercentile(f_regression, percentile=20)), ('prediction', l_svr)])

5. Visualization

Feature Importance Visualization

- The weights (feature importances) learned by the SVR model are visualized as a matrix and plotted on a connectome map.

python

feat_exp_matrix = correlation_measure.inverse_transform(l_svr.coef_)[0]

plotting.plot_connectome(feat_exp_matrix, coords, colorbar=True)

Interactive Connectome Visualization

- Nilearn's interactive visualization feature allows users to explore the connectome interactively.

python

plotting.view_connectome(feat_exp_matrix, coords, edge_threshold='98%')

6. Model Testing on Unseen Data

- After training the model on the training set, it is tested on a left-out validation set to assess its performance on unseen data.

python

l_svr.fit(X_train, y_train_log)

y_pred = l_svr.predict(X_val)

acc = l_svr.score(X_val, y_val_log)

mae = mean_absolute_error(y_val_log, y_pred)

Conclusion

This notebook demonstrates how to:

1. Load and preprocess rs-fMRI data.
2. Extract brain connectivity features using an atlas-based approach.

3. Build and evaluate a machine learning model (SVR) for predicting age from rs-fMRI data.
4. Tune hyperparameters and perform feature selection to improve model performance.
5. Visualize feature importance and brain connectivity using Nilearn's plotting tools.