Supplementary Material

**Classification of Autism Spectrum Disorder using Electroencephalography in Chinese Children: A Cross-sectional Retrospective Study**

**Si Yang Ke1,3†, Huiwen Wu2†, Haoqi Sun1,4, Aiqin Zhou2, Jianhua Liu6, Xiaoyun Zheng2, Kevin Liu1,3, M. Brandon Westover4,5\*, Haiqing Xu2\* , and Xue-jun Kong1,7\***

**\* Correspondence:** xkong1@mgh.harvard.edu

**Python code for performing the LOOCV for machine learning classification.**

**1. Helper function logistic\_regression\_netsed\_cv() that is used in forward selection.** It performs a 10 × 10 nested cross validation (CV) using logistic regression where the where the inner CV selects the best alpha (regularization strength) based on ROC-AUC highest. The mean ROC AUC of the outer CV is returned.

from sklearn.model\_selection import GridSearchCV, RepeatedStratifiedKFold

from sklearn.impute import SimpleImputer, KNNImputer

from sklearn.metrics import recall\_score, precision\_score, roc\_auc\_score

from sklearn.linear\_model import LogisticRegression

import numpy as np

def logistic\_regression\_nested\_cv(feature\_list, X\_train,y\_train):

“”“ This function is used by forward selection, it takes in a

feature subset and performs 10 x 10 nested CV using logistic regression.

The mean CV ROC-AUC is returned”””

    X\_train = X\_train[feature\_list]

    numeric\_cols = [col for col in feature\_list if col not in [ 'sex']]

    categorical\_cols = ['sex']

 # Initialize the array to store the nested scores

 nested\_roc\_auc = np.zeros(10)

    # Define the indices for the outer loop splits

    outer\_cv = StratifiedKFold(n\_splits=10, random\_state=7, shuffle=True)

    outer\_cv\_splits = outer\_cv.split(X\_train, y\_train)

    # Loop over the outer loop splits

    for j, (train\_val\_idx, train\_test\_idx) in enumerate(outer\_cv\_splits):

        # Split the data into training/validation and testing sets

        X\_train\_val, y\_train\_val = X\_train\_val.loc[train\_val\_idx,:], y\_val[train\_val\_idx]

        X\_train\_test, y\_train\_test = X\_train\_val.loc[train\_test\_idx,:], y\_val[train\_test\_idx]

        steps = [('impute', SimpleImputer(strategy='mean'))]

        pipeline = Pipeline(steps=steps)

        # Apply the pipeline on the training/validation set

        X\_train\_val = pipeline.fit\_transform(X\_train\_val)

        # Apply the pipeline on the testing set

        X\_test\_val = pipeline.transform(X\_test\_val)

        # Define the indices for the inner loop splits

        inner\_cv = RepeatedStratifiedKFold(n\_splits=10, random\_state=42, n\_repeats=1)

        # Define the estimator and the parameter grid for GridSearchCV

        lr = LogisticRegression(class\_weight='balanced',max\_iter=1000)

        param\_grid = {'C': [0.0001,0.001, 0.01, 0.1, 1, 10, 100]}

        clf = GridSearchCV(estimator=lr, param\_grid=param\_grid, cv=inner\_cv,n\_jobs=-1,scoring='roc\_auc')

        # Fit the estimator on the training/validation set and calibrate the predictions

        clf.fit(X\_train\_val, y\_train\_val)

        # find the best cutoff

        roc\_auc = roc\_auc\_score(y\_train\_test, clf.predict\_proba(X\_train\_test)[:, 1])

        # Store the scores in the array

        nested\_roc\_auc[j] = roc\_auc

    return nested\_roc\_auc.mean()

**2. Function forward\_selection() that performs forward selection given the top 15 features from the nested inference step. It iteratively adds features from the top 15 feature list to the current set of features and evaluates the performance using logistic regression with nested cross-validation. The best set of features is then determined based on the maximum mean ROC AUC.**

def forward\_selection(X\_train, y\_train,top\_15\_features):

    all\_mean\_roc\_auc = []

    for i in tqdm(range(len(top\_15\_features))):

        features = top\_15\_features[:i+1]

        features = features + ['age\_months','sex']

        all\_mean\_roc\_auc.append(logistic\_regression\_nested\_cv(features,X\_train,y\_train))

    best\_index = np.argmax(all\_mean\_roc\_auc)

    best\_features = top\_15\_features[:best\_index+1] + ['age\_months', 'sex']

    return best\_features, all\_mean\_roc\_auc

**3. Main for loop for performing LOOCV that uses the two functions defined previously in 1 and 2.**

from sklearn.model\_selection import LeaveOneOut

from sklearn.preprocessing import LabelEncoder

y\_true\_list = []

y\_pred\_proba\_list = []

y\_pred\_list = []

y = df['ASD']

# store features as X

X = df.copy(deep=True)[features\_names]

# label encode the sex column

le = LabelEncoder()

X['sex'] = le.fit\_transform(X['sex'])

loo = LeaveOneOut()

# Loop over the outer loop splits

for j, (train\_idx, test\_idx) in enumerate(loo.split(df)):

    # Split the data into training/validation and testing sets

    train = df.loc[train\_idx,:]

    X\_train, y\_train = X.loc[train\_idx,:], y[train\_idx]

    X\_test, y\_test = X.loc[test\_idx,:], y[test\_idx]

    features = X\_train.columns

    # save X\_train\_val to a csv file

    train\_val.to\_csv('train.csv',index=False)

    # Run the IPW or Optimal Full Match to get list of top 15 features

    print("Generating Ranked P-value List From R")

    args = ["Rscript", "get\_significance\_list\_optimized.R", "full\_match"]

    res = subprocess.run(args, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

    try:

        significance\_ranking = pd.read\_csv('ranking\_list.csv')

        top\_15\_features = significance\_ranking.outcome.tolist()

    except FileNotFoundError:

        assert False, "File not found!"

 # Forward selection

    best\_features, all\_mean\_roc\_auc = forward\_selection(X\_train\_val.reset\_index(drop=True),y\_train\_val.reset\_index(drop=True),top\_15\_features)

    X\_train = X\_train[best\_features]

    X\_test = X\_test[best\_features]

    steps = [('mean\_impute', SimpleImputer(strategy='mean'))]

    pipeline = Pipeline(steps=steps)

    # Apply the pipeline on the trainin

    X\_train = pipeline.fit\_transform(X\_train)

    # Apply the pipeline on the testing set

    X\_test = pipeline.transform(X\_test)

    # define inner cv

    inner\_cv =  RepeatedStratifiedKFold(n\_splits=10, random\_state=42, n\_repeats=3)

    # Define the estimator and the parameter grid for GridSearchCV

    lr = LogisticRegression(class\_weight='balanced',max\_iter=1000)

    param\_grid = {'C': [0.0001,0.001, 0.01, 0.1, 1, 10, 100]}

    clf = GridSearchCV(estimator=lr, param\_grid=param\_grid, cv=inner\_cv,n\_jobs=-1,scoring='roc\_auc')

    clf.fit(X\_train, y\_train)

    nested\_C = clf.best\_params\_['C']

    y\_pred = clf.predict(X\_test)

    y\_pred\_proba = clf.predict\_proba(X\_test)[:,1]

    y\_true\_list.extend(y\_test)

    y\_pred\_proba\_list.extend(y\_pred\_proba)

    y\_pred\_list.extend(y\_pred)

    old\_significance\_ranking = significance\_ranking

    print("Finished LOOCV Iteration ",j)