**Supplementary Table 2: Pseudocode for the IEL algorithm**

Inputs: IEL (evolutionary algorithm) parameters; feature set; prediction targets, generation limit

**for** target in prediction targets do:

 if desired, rebalance sampling data (we used SMOTEENN)

 set hyperparameter values and numbers of features with random values in desired ranges

 Create empty fitness\_list, feature\_importances\_list

 #run n=100 models with all permutations in initial generation of evolutionary learning

   **for** each matrix of hyperparameter settings and feature sets in initial generation do:

 sample X

 Create empty fitness\_scores, feature\_importance\_scores

 compute k for cross-validation

 **for** each fold in cross-validation do:

 train model on fold

 store model’s scores in fitness\_scores, feature\_importance\_scores

 compute and store mean(fitness\_scores) in fitness\_list

 compute and store mean(feature\_importance\_scores) in feature\_importances\_list

 rank models by performance and store in ranked\_fitness

 create queue to store top 3 performing models from each generation and initialize with first 3

 Create empty lists for best fitness function, feature sets and hyperparameter settings

 #Continue genetic algorithm for second to final learning generations

 initialize second learning generation, g, and convergence condition = False

 **while** g < total number of desired learning generations and converged == False do:

 #mating of parent models

 set parents = select top 40 models

 set children = generate 20 children by crossover at pivot point (hyperparameters, feature sets) from parents

 #mutation of parent models

 set mutations = select next 20 best-performing models

 set mutation\_children = change value of hyperparameter by specified amount

 # establish new generation of child models

 set new\_children = children + mutation\_children

 initialize 60 new random\_children from distributions

 set new population\_arrays = new\_children + random\_children

 initialize score\_lists based on various metrics

  **for** each matrix of hyperparameter settings and feature sets in population\_arrays do

 sample X

 initialize empty metric lists (fitness\_list, statistical metrics of choice)

 compute k for cross-validation

 **for** each fold in dataset do

 train model on fold

 store model’s scores in metric lists

 compute and store mean(metrics) in mean metric lists

 set ranked\_fitness = fitness\_list

 order ranked\_fitness based on fitness\_score and add top 3 with their hyperparameters to best\_lists

 add top fitness models to queue
 #determine convergence or continuance of learning

 **if** length(queue) < minimum size of queue set in IEL parameters

 increment g by 1 and continue learning

 **else**

 **if** convergence condition not met

 delete first column from queue

 increment g by 1 and continue learning

 **else** stop learning