+double getOutRange(); +double getOutBase();

+void train(vec<trainingExample> trainingSet);

-double activationFunction(double);
-double getHiddenErrorGradient(int, int);

-void backpropagate(double);

-void updateWeights();

baseModel struct trainingExample{ vec<double> in; vec<double> out} + ~baseModel() +double process(vector<double>) +void train(vector<trainingExample>) +int getNumInputs() +vec<int> getWhichInputs() +void getJSONDescription(Json::Value &model) #Json::Value vector2json(T vec) neuralNetwork knnClassification -int numInputs; -int numInputs; -vec<int> whichInputs; -vec<int> whichInputs; -int numHiddenLayers; -vec<trainingExample> neighbours; -int numHiddenNodes: -int numNeighbours; -vec-<vec<double>>> weights: -pair<int, double> * nearestNeighbours; -vec<double> wHiddenOutput; +int getNumHiddenLavers(): -vec<double> inRanges: +int getNumHiddenNodes(): -vec<double> inBases: -double outRange; +vec<double> getWeights(); +vec<double> getWHiddenOutput(); -double outBase; +vec<double> getInRanges(): -double learningRate: +vector<double> getInBases(); -double momentum; -int numEpochs; +double getOutRange(); -vec<vec<double>>> deltaWeights: +double getOutBase(); -vec<double> deltaHiddenOutput; -vec<double> hiddenErrorGradients; -double outputErrorGradient; +int getNumHiddenLayers(); +int getNumHiddenNodes(); +vec<double> getWeights(); +vec<double> getWHiddenOutput(); +vec<double> getInRanges(); +vector<double> getInBases();

modelSet #vector
baseModel*> myModelSet; #int numInputs; #vec<string> inputNames; #int numOutputs; #bool created; +modelSet(); +~modelSet(); +bool train(vec<trainingExample> trainingSet); +vec<double> process(vec<double> inputVector); +string getJSON(); +void writeJSON(string filepath); +bool putJSON(string jsonMessage); +bool readJSON(string filepath); -Json::Value parse2json(); -void json2modelSet(Json::Value); classification +classification(); +classification(vec<trainingExample> trainingSet); +classification(int numInputs, int numOutputs); +bool train(vec<trainingExample> trainingSet); regression +regression(); +regression(vec<trainingExample> trainingSet); +regression(int numInputs, int numOutputs); +bool train(vec<trainingExample> trainingSet);

^{*} vec and string are the stdlib types std::vector and std::string