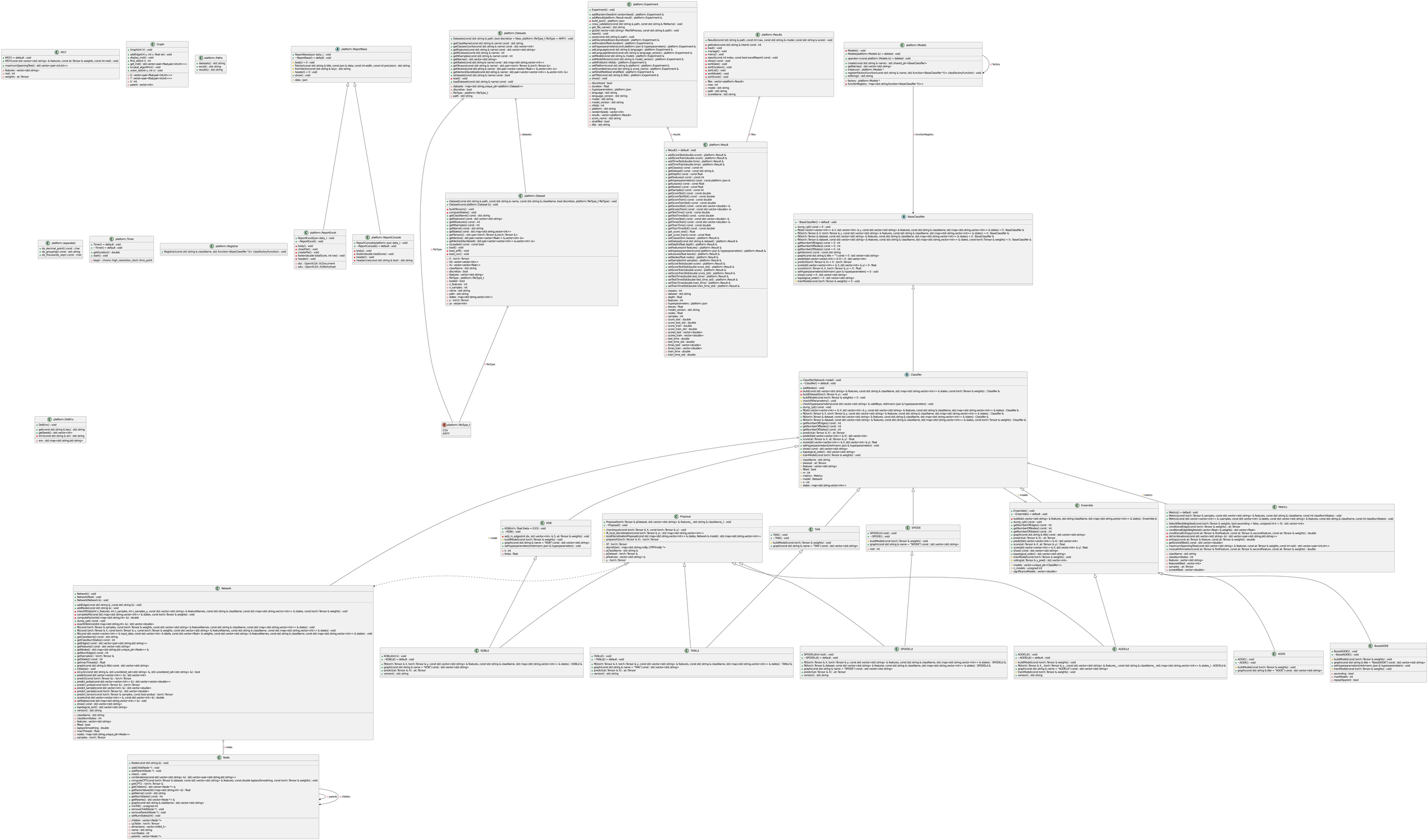
MST
+ MST() = default : void
+ MST(const std::vector<std::string> & features, const at::Tensor & weights, const int root) : void
+ maximumSpanningTree() : std::vector<std::vector<int,int>>
- features : vector<std::string>
- root : int
- weights : at::Tensor
Graph
+ Graph(int V) : void
+ addEdge(int u, int v, float wt) : void
+ display_mst() : void
- find_set(int i) : int
+ get_mst() : std::vector<pair<float,pair<int,int>>>
+ kruskal_algorithm() : void
+ union_set(int u, int v) : void
- G : vector<pair<float,pair<int,int>>>
- T : vector<pair<float,pair<int,int>>>
- V : int
- parent : vector<int>
platform::Paths
+ dataset(std::string) : void
+ excel() : std::string
+ results() : std::string
platform::ReportBase
+ ~ReportBase(int=0 data_) : void
+ ~~ReportBase() = default : void
+ body() = 0 : void
+ chart(int, const std::string & title, const json & data, const int width, const int precision) : std::string
+ formatSection(const std::string & key) : std::string
+ header() = 0 : void
+ show() : void
- data : json
platform::Datasets
+ Datasets(const std::string & path, bool discretize = false, platform::FileType_t fileType = ARFF) : void
+ getClassNames(const std::string & name) const : std::string
+ getClassNumStates(const std::string & name) const : std::vector<int>
+ getFeatures(const std::string & name) const : std::vector<std::string>
+ getCSCases(const std::string & name) : int
+ getNSamples(const std::string & name) const : int
+ getNames() : std::vector<std::string>
+ getStates(const std::string & name) const : std::map<std::string,vector<int>>
+ getTensors(const std::string & name, const torch::Device & device) : Tensor &,Tensor &
+ getVectors(const std::string & name) : std::pair<vector<float> = &,vector<int> = &>
+ getVectorsDiscretized(const std::string & name) : std::pair<vector<vector<int>> &,vector<int> &>
+ isDataset(const std::string & name) const : bool
+ load() : void
+ loadDataset(const std::string & name) const : void
- datasets : map<std::string,unique_ptr<platform::Dataset>>
- discretize : bool
- fileType : platform::FileType_t
- path : std::string
platform::Experiment
+ Experiment() : void
+ addRandomSeed(int randomSeed) : platform::Experiment &
+ addResult(platform::Result result) : platform::Experiment &
+ build_json() : platform::json
+ cross_validation(const std::string & path, const std::string & fileName) : void
+ fit() : platform::Result
+ go(std::vector<std::string> fileNames, const std::string & path) : void
+ report() : void
+ save(const std::string & path) : void
+ setDiscretized(bool discretized) : platform::Experiment &
+ setDuration(float duration) : platform::Experiment &
+ setHyperparameters(const std::string & hyperparameters) : platform::Experiment &
+ setLanguage(const std::string & language) : platform::Experiment &
+ setLanguageVersion(const std::string & language_version) : platform::Experiment &
+ setModel(const std::string & model) : platform::Experiment &
+ setModelVersion(const std::string & model_version) : platform::Experiment &
+ setNfolds(int nfolds) : platform::Experiment &
+ setNormalized(bool normalized) : platform::Experiment &
+ setScoreName(const std::string & score_name) : platform::Experiment &
+ setStratified(bool stratified) : platform::Experiment &
+ setTitle(const std::string & title) : platform::Experiment &
+ show() : void
- discretized : bool
- duration : float
- hyperparameters : platform::json
- language : std::string
- language_version : std::string
- model : std::string
- model_version : std::string
- nfolds : int
- platform : std::string
- randomSeeds : vector<int>
- results : vector<platform::Result>
- score_name : std::string
- stratified : bool
- title : std::string
platform::Results
+ Results(const std::string & path, const int max, const std::string & model, const std::string & score) : void
+ getIndex(const std::string & key, const int* const) const : int
+ load() : void
+ manage() : void
+ menuLine() : void
+ report(const int index, const bool excelReport) const : void
+ show() const : void
+ sortDate() : void
+ sortDuration() : void
+ sortLoad() : void
+ sortModel() : void
+ sortScore() : void
- files : vector<platform::Result>
- max : int
- model : std::string
- path : std::string
- scoreName : std::string
platform::Models
+ Models() : void
+ Models(platform::Models &) = deleted : void
+ operator=(const platform::Models &) = deleted : void
+ create(const std::string & name) : std::shared_ptr<BaseClassifier>
+ getClassifiers() : std::vector<std::string>
+ instance() : platform::Models*
+ registerFactoryFunction(const std::string & name, std::function<BaseClassifier *()> classfactoryFunction) : void
+ toString() : std::string
- factory : platform::Models *
- functionRegistry : map<std::string,function<BaseClassifier *()>>
platform::separated
+ dig_decimal_point(int) const : char
+ dig_thousands_sep() const : std::string
+ do_thousands_sep() const : char
platform::Timer
+ Timer() = default : void
+ ~Timer() = default : void
+ getDuration() : double
+ start() : void
- begin : chrono::high_resolution_clock::time_point
platform::Registrar
+ Registrar(const std::string & className, std::function<BaseClassifier *()> classfactoryFunction) : void
platform::ReportExcel
+ ReportExcel(int=0 data_) : void
+ ~ReportExcel() : void
+ body() : void
+ closeFile() : void
+ createFile() : void
+ footer(double totalScore) : void
+ header() : void
+ headerLine(int=0 data_) : void
- file : OpenXLSX::XLDocument
- wks : OpenXLSX::XLWorksheet
platform::ReportConsole
+ ReportConsole(platform::json data_) : void
+ ~~ReportConsole() = default : void
+ body() : void
+ footer(double totalScore) : void
+ header() : void
+ headerLine(int=0 data_) : void
platform::Dataset
+ Dataset(const std::string & path, const std::string & name, const std::string & className, bool discretize, platform::FileType_t fileType) : void
+ Dataset(const platform::Dataset &) : void
+ buildTensors() : void
+ computeStates() : void
+ getClassName() const : std::string
+ getFeatures() const : std::vector<std::string>
+ getNfeatures() const : int
+ getNSamples() const : int
+ getName() const : std::string
+ getStates() const : std::map<std::string,vector<int>>
+ getTensors() : std::pair<torch::Tensor &,torch::Tensor &>
+ getVectors() : std::pair<vector<float> = &,vector<int> &>
+ getVectorsDiscretized() : std::pair<vector<vector<int>> &,vector<int> &>
+ isDataset() const : bool
+ load() : void
+ load_arff() : void
+ load_csv() : void
- X : torch::Tensor
- Xd : vector<vector<int>>
- Xv : vector<vector<float>>
- className : std::string
- discretize : bool
- features : vector<std::string>
- fileType : platform::FileType_t
- loaded : bool
- n_features : int
- n_samples : int
- name : std::string
- path : std::string
- states : map<std::string,vector<int>>
- y : torch::Tensor
- yv : vector<int>
platform::FileType_t
CSV
ARFF
platform::DotEnv
+ DotEnv() : void
+ get(const std::string & key) : std::string
+ getSeed() : std::vector<int>
+ trim(const std::string & str) : std::string
- env : std::map<std::string,std::string>
BaseClassifier
+ ~BaseClassifier() = default : void
+ dump_cpt() const = 0 : void
+ fit(std::vector<vector<int>> & X, std::vector<int> & y, const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states) = 0 : BaseClassifier &
+ fit(torch::Tensor & X, torch::Tensor & y, const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states) = 0 : BaseClassifier &
+ fit(torch::Tensor & dataset, const int [, const std::string & name = nullptr], const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states, const torch::Tensor & weights) = 0 : BaseClassifier &
+ fit(torch::Tensor & dataset, const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states, const torch::Tensor & weights) = 0 : BaseClassifier &
+ getNumberOfEdges() const = 0 : int
+ getNumberOfNodes() const = 0 : int
+ getNumberOfStates() const = 0 : int
+ getClassifiers() const : std::vector<std::string>
+ graph(const std::string & title = "") const = 0 : std::vector<std::string>
+ predict(std::vector<vector<int>> & X) = 0 : std::vector<int>
+ predict(torch::Tensor & X) = 0 : torch::Tensor
+ scores(std::vector<vector<int>> & X, std::vector<int> & y) = 0 : float
+ score(torch::Tensor & X, torch::Tensor & y) = 0 : float
+ setHyperparameters(nlohmann::json & hyperparameters) = 0 : void
+ show() const = 0 : std::vector<std::string>
+ topological_order() = 0 : std::vector<std::string>
+ trainModel(const torch::Tensor & weights) = 0 : void
Classifier
+ ClassifierNetwork model) : void
+ ~Classifier() = default : void
+ addNodes() : void
+ build(const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states, const torch::Tensor & weights) : Classifier &
+ buildDataset(torch::Tensor & y) : void
+ buildModel(const torch::Tensor & weights) = 0 : void
+ checkFitParameters() : void
+ checkHyperparameters(const std::vector<string> & validKeys, nlohmann::json & hyperparameters) : void
+ dump_cpt() const : void
+ fit(std::vector<vector<int>> & X, std::vector<int> & y, const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states) : Classifier &
+ fit(torch::Tensor & X, torch::Tensor & y, const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states) : Classifier &
+ fit(torch::Tensor & dataset, const int [, const std::string & name = nullptr], const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states) : Classifier &
+ fit(torch::Tensor & dataset, const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states, const torch::Tensor & weights) : Classifier &
+ getNumberOfEdges() const : int
+ getNumberOfNodes() const : int
+ getNumberOfStates() const : int
+ getVertices() const : std::vector<int>
+ predict(std::vector<vector<int>> & X) : std::vector<int>
+ predict(torch::Tensor & X) : torch::Tensor
+ scores(std::vector<vector<int>> & X, std::vector<int> & y) : float
+ score(torch::Tensor & X, torch::Tensor & y) : float
+ setHyperparameters(nlohmann::json & hyperparameters) : void
+ show() const : std::vector<std::string>
+ topological_order() : std::vector<std::string>
+ trainModel(const torch::Tensor & weights) : void
- className : std::string
- classNumStates : int
- features : vector<std::string>
- fitted : bool
- m : int
- metrics : Metrics
- model : Network
- n : int
- states : map<std::string,vector<int>>
KDB
+ KDB(int k, float theta = 0.03) : void
+ ~KDB() : void
+ add_m_edges(int idx, std::vector<int> & S, at::Tensor & weights) : void
+ buildModel(const torch::Tensor & weights) : void
+ graph(const std::string & name = "KDB") const : std::vector<std::string>
+ setHyperparameters(nlohmann::json & hyperparameters) : void
- k : int
- theta : float
Proposal
+ Proposal(torch::Tensor & pDataset, std::vector<std::string> & features_, std::string & clasName_) : void
+ ~~Proposal() : void
+ checkInputs(const torch::Tensor & X, const torch::Tensor & y) : void
+ fit_local_discretization(const torch::Tensor & y) : std::map<int,vector<int>>
+ localDiscretizationProposal(const std::map<int,vector<int>> & states, Network & model) : std::map<std::string,vector<int>>
+ prepareX(torch::Tensor & X) : torch::Tensor
- Xf : torch::Tensor
- discretizers : map<std::string,mdlp::CPPFimdlp *>
- pClassName : std::string &
- pDataset : torch::Tensor &
- pFeatures : vector<std::string> &
- y : torch::Tensor
TAN
+ TAN() : void
+ ~~TAN() : void
+ buildModel(const torch::Tensor & weights) : void
+ graph(const std::string & name = "TAN") const : std::vector<std::string>
SPODE
+ SPODE(int root) : void
+ ~~SPODE() : void
+ buildModel(const torch::Tensor & weights) : void
+ graph(const std::string & name = "SPODE") const : std::vector<std::string>
- root : int
Ensemble
+ Ensemble() : void
+ ~~Ensemble() = default : void
+ buildFit(std::vector<std::string> & features, std::string clasName, std::map<std::string,vector<int>> & states) : Ensemble &
+ dump_cpt() const : void
+ getNumberOfEdges() const : int
+ getNumberOfNodes() const : int
+ getNumberOfStates() const : int
+ graph(const std::string & title) const : std::vector<std::string>
+ predict(at::Tensor & X) : at::Tensor
+ predict(std::vector<vector<int>> & X) : std::vector<int>
+ scores(at::Tensor & X, at::Tensor & y) : float
+ scores(std::vector<vector<int>> & X, std::vector<int> & y) : float
+ trainModel(const torch::Tensor & weights) : void
+ voting(at::Tensor & y_pred) : std::vector<int>
- models : vector<unique_ptr<Classifier>>
- n_models : unsigned int
- significanceModels : vector<double>
Metrics
+ Metrics() = default : void
+ Metrics(const torch::Tensor & samples, const int& vector<std::string> & features, const std::string & clasName, const int clasNumStates) : void
+ Metrics(const int& vector<vector<int>> & vsamples, const int& vector<int> & labels, const int& vector<std::string> & features, const std::string & clasName, const int clasNumStates) : void
+ SelectBestWeighted(const torch::Tensor & weights, bool ascending = false, unsigned int k = 0) : std::vector<int>
+ conditionalEdge(const torch::Tensor & weights) : at::Tensor
+ conditionalEdgeWeights(int) const : vector<float>
+ doCombinations(const int& vector<int> & source, const int& secondFeature, const at::Tensor & weights) : double
+ doCombinations(const int& vector<int> & s) : std::vector<pair<int,int>>
+ entropy(const at::Tensor & feature, const at::Tensor & weights) : double
+ getConfusionMatrix() const : std::vector<double>
+ maximumSpanningTree(const int& vector<int> & features, const at::Tensor & weights, const int root) : std::vector<pair<int,int>>
+ mutualInformation(const at::Tensor & firstFeature, const at::Tensor & secondFeature, const at::Tensor & weights) : double
- clasName : std::string
- clasNumStates : int
- features : vector<std::string>
- numFeatures : unsigned int
- samples : at::Tensor
- scoresBest : vector<double>
KDBLd
+ KDBLd(int k) : void
+ ~~KDBLd() = default : void
+ fit(torch::Tensor & X, torch::Tensor & y, const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states) : KDBLd &
+ graph(const std::string & name = "KDB") const : std::vector<std::string>
+ predict(at::Tensor & X) : at::Tensor
+ version() : std::string
TANLd
+ TANLd() : void
+ ~~TANLd() = default : void
+ fit(torch::Tensor & X, torch::Tensor & y, const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states) : TANLd &
+ graph(const std::string & name = "TAN") const : std::vector<std::string>
+ predict(at::Tensor & X) : at::Tensor
+ version() : std::string
SPODELd
+ SPODELd(int root) : void
+ ~~SPODELd() = default : void
+ fit(torch::Tensor & X, torch::Tensor & y, const std::vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states) : SPODELd &
+ fit(torch::Tensor & dataset, const int& vector<std::string> & features, const std::string & clasName, std::map<std::string,vector<int>> & states) : SPODELd &
+ graph(const std::string & name = "SPODE") const : std::vector<std::string>
+ predict(at::Tensor & X) : at::Tensor
+ version() : std::string
AODELd
+ AODELd() : void
+ ~~AODELd() = default : void
+ buildModel(const torch::Tensor & weights) : void
+ fit(torch::Tensor & X_, torch::Tensor & y_, const std::vector<std::string> & features_, const std::string & clasName_, std::map<std::string,vector<int>> & states_) : AODELd &
+ graph(const std::string & name = "AODELd") const : std::vector<std::string>
+ predict(at::Tensor & X, torch::Tensor & weights) : void
+ version() : std::string
AODE
+ AODE() : void
+ ~~AODE() : void
+ buildModel(const torch::Tensor & weights) : void
+ graph(const std::string & title = "AODE") const : std::vector<std::string>
BoostAODE
+ BoostAODE() : void
+ ~~BoostAODE() : void
+ buildModel(const torch::Tensor & weights) : void
+ graph(const std::string & title = "BoostAODE") const : std::vector<std::string>
+ setHyperparameters(nlohmann::json & hyperparameters) : void
+ trainModel(const torch::Tensor & weights) : void
- ascending : bool
- maxModels : int
- repeatSparent : bool
Network
+ Network() : void
+ Network(float) : void
+ Network(Network &) : void
+ addEdge(const std::string &, const std::string &) : void
+ addNode(const std::string &) : void
+ checkFitData(int n_features, int n_samples, int n_samples_y, const std::vector<std::string> & featureNames, const std::string & clasName, const std::map<std::string,vector<int>> & states, const torch::Tensor & weights) : void
+ completePosterior(int, const std::vector<int>, const std::vector<int>) : void
+ computeFactor(int, const std::vector<int>&) : double
+ dump_cpt() const : void
+ exactInference(std::map<std::string,int> &) : std::vector<double>
+ fit(const torch::Tensor & samples, const torch::Tensor & weights, const std::vector<std::string> & featureNames, const std::string & clasName, const std::map<std::string,vector<int>> & states) : void
+ fit(const torch::Tensor & X, const torch::Tensor & y, const torch::Tensor & weights, const std::vector<std::string> & featureNames, const std::string & clasName, const std::map<std::string,vector<int>> & states) : void
+ fit(std::vector<std::vector<int>> & X_train, std::vector<int> & y_train, std::vector<std::vector<int>> & X_test, std::vector<int> & y_test, const std::vector<std::string> & features, const std::string & clasName, const std::map<std::string,vector<int>> & states) : void
+ getClassName() const : std::string
+ getClassNumStates() const : int
+ getEdges() const : std::vector<pair<std::string,std::string>>
+ getFeatures() const : std::vector<std::string>
+ getInstance(int, std::vector<int>&) const : std::pair<std::string,std::unique_ptr<Node>> &
+ getNumEdges() const : int
+ getSamples() : torch::Tensor &
+ getStates() const : int
+ getmaxThreads() : float
+ graph(const std::string & title) const : std::vector<std::string>
+ initialize() : void
+ isCyclic(int, std::string &, std::unordered_set<std::string> &, std::unordered_set<std::string> &) : bool
+ predict(const std::vector<vector<int>> &) : std::vector<int>
+ predict(const torch::Tensor &) : torch::Tensor
+ predict_proba(const std::vector<vector<int>> &) : std::vector<vector<double>>
+ predict_proba(const torch::Tensor &) : torch::Tensor
+ predict_sample(const std::vector<int> &) : std::vector<double>
+ predict_sample(const torch::Tensor &) : std::vector<double>
+ predict_tensor(const torch::Tensor & samples, const bool proba) : torch::Tensor
+ score(const std::vector<vector<int>> &, const std::vector<int> &) : double
+ setStates(const std::map<std::string,vector<int>> &) : void
+ show() const : std::vector<std::string>
+ topological_sort() : std::vector<std::string>
+ trainModel() : void
- className : std::string
- classNumStates : int
- features : vector<std::string>
- fitted : bool
- laplaceSmoothing : double
- maxThreads : float
- nodes : unordered_map<std::string,unique_ptr<Node>>
- samples : torch::Tensor
Node
+ Node(const std::string &) : void
+ addChildNode *) : void
+ addParentNode *) : void
+ clear() : void
+ computeCPT(const std::vector<std::string> &) : std::vector<pair<std::string,std::string>>
+ computeCPT(const torch::Tensor & dataset, const std::vector<std::string> & features, const double laplaceSmoothing, const torch::Tensor & weights) : void
+ getCPT() : torch::Tensor &
+ getChildren() : std::vector<Node *> &
+ getFactorValue(std::map<std::string,int> &) : float
+ getName() const : std::string
+ getNumStates() : int
+ getParents() : std::vector<Node *> &
+ graph(const std::string & clasName) : std::vector<std::string>
+ minFill() : unsigned int
+ removeChildNode *) : void
+ removeParentNode *) : void
+ setNumStates(int) : void
- children : vector<Node *>
- cpTable : torch::Tensor
- dimensions : vector<int64_t>
- name : std::string
- numStates : int
- parents : vector<Node *>
parents
children
models
metrics
factory
functionRegistry
results
files
datasets
fileType
node
nodes