

tree_ensemble_classifier.predict

...

Copy fnpredict(refself:TreeEnsembleClassifier,X:Tensor)->(Span, MutMatrix::);

...

Tree Ensemble classifier. Returns the top class for each of N inputs.

Args

- self
- : TreeEnsembleClassifier - A TreeEnsembleClassifier object.
- X
- : Input 2D tensor.
-

Returns

- N Top class for each point
- The class score Matrix for each class, for each point.
-

Type Constraints

TreeEnsembleClassifier andX must be fixed points

Examples

...

```
Copy useorion::numbers::FP16x16; useorion::operators::tensor::{Tensor, TensorTrait, FP16x16Tensor, U32Tensor};
useorion::operators::ml::{NODE_MODES, TreeEnsembleAttributes, TreeEnsemble}; useorion::operators::ml::{
TreeEnsembleClassifier, POST_TRANSFORM, TreeEnsembleClassifierTrait }; useorion::operators::matrix::
{MutMatrix, MutMatrixImpl};
```

```
fn tree_ensemble_classifier_helper( post_transform: POST_TRANSFORM )->(TreeEnsembleClassifier, Tensor) {
let class_ids: Span = array![0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2] .span();
```

```
let class_node_ids: Span = array![2, 2, 2, 3, 3, 3, 4, 4, 4, 1, 1, 1, 3, 3, 3, 4, 4, 4] .span();
```

```
let class_tree_ids: Span = array![0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1] .span();
```

```
let class_weights: Span = array![ FP16x16{ mag:30583, sign:false}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:2185,
sign:false}, FP16x16{ mag:13107, sign:false}, FP16x16{ mag:15729, sign:false}, FP16x16{ mag:3932, sign:false}, FP16x16{
mag:0, sign:false}, FP16x16{ mag:32768, sign:false}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:32768, sign:false},
FP16x16{ mag:0, sign:false}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:29491, sign:false}, FP16x16{ mag:0, sign:false},
FP16x16{ mag:3277, sign:false}, FP16x16{ mag:6746, sign:false}, FP16x16{ mag:12529, sign:false}, FP16x16{ mag:13493,
sign:false}, ] .span();
```

```
let class_labels: Span = array![0, 1, 2] .span();
```

```
let nodes_false_node_ids: Span = array![4, 3, 0, 0, 0, 2, 0, 4, 0, 0] .span();
```

```
let nodes_feature_ids: Span = array![1, 0, 0, 0, 0, 1, 0, 0, 0, 0] .span();
```

```
let nodes_missing_value_tracks_true: Span = array![0, 0, 0, 0, 0, 0, 0, 0, 0, 0] .span();
```

```
let nodes_modes: Span = array![ NODE_MODES::BRANCH_LEQ, NODE_MODES::BRANCH_LEQ, NODE_MODES::LEAF,
NODE_MODES::LEAF, NODE_MODES::LEAF, NODE_MODES::BRANCH_LEQ, NODE_MODES::LEAF,
NODE_MODES::BRANCH_LEQ, NODE_MODES::LEAF, NODE_MODES::LEAF, ] .span();
```

```
let nodes_node_ids: Span = array![0, 1, 2, 3, 4, 0, 1, 2, 3, 4] .span();
```

```
let nodes_tree_ids: Span = array![0, 0, 0, 0, 0, 1, 1, 1, 1, 1] .span();
```

```
let nodes_true_node_ids: Span = array![1, 2, 0, 0, 0, 1, 0, 3, 0, 0] .span();
```

```
let nodes_values: Span = array![ FP16x16{ mag:81892, sign:false}, FP16x16{ mag:19992, sign:true}, FP16x16{ mag:0,
sign:false}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:110300, sign:true}, FP16x16{ mag:0,
```

```

sign:false}, FP16x16{ mag:44245, sign:true}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:0, sign:false}, ] .span();

lettree_ids:Span=array![0,1].span();

letmutroot_index:Felt252Dict=Default::default(); root_index.insert(0,0); root_index.insert(1,5);

letmutnode_index:Felt252Dict=Default::default(); node_index
.insert(2089986280348253421170679821480865132823066470938446095505822317253594081284,0); node_index
.insert(2001140082530619239661729809084578298299223810202097622761632384561112390979,1); node_index
.insert(2592670241084192212354027440049085852792506518781954896144296316131790403900,2); node_index
.insert(2960591271376829378356567803618548672034867345123727178628869426548453833420,3); node_index
.insert(458933264452572171106695256465341160654132084710250671055261382009315664425,4); node_index
.insert(1089549915800264549621536909767699778745926517555586332772759280702396009108,5); node_index
.insert(1321142004022994845681377299801403567378503530250467610343381590909832171180,6); node_index
.insert(2592987851775965742543459319508348457290966253241455514226127639100457844774,7); node_index
.insert(2492755623019086109032247218615964389726368532160653497039005814484393419348,8); node_index
.insert(1323616023845704258113538348000047149470450086307731200728039607710316625916,9);

letatts=TreeEnsembleAttributes{ nodes_falsenodeids, nodes_featureids, nodes_missing_value_tracks_true, nodes_modes,
nodes_nodeids, nodes_treeids, nodes_truenodeids, nodes_values };

letmutensemble:TreeEnsemble=TreeEnsemble{ atts, tree_ids, root_index, node_index };

letbase_values:Option>=Option::None;

letmutclassifier:TreeEnsembleClassifier=TreeEnsembleClassifier{ ensemble, class_ids, class_nodeids, class_treeids,
class_weights, classlabels, base_values, post_transform };

letmutX:Tensor=TensorTrait::new( array![3,3].span(), array![ FP16x16{ mag:65536, sign:true}, FP16x16{ mag:52429,
sign:true}, FP16x16{ mag:39322, sign:true}, FP16x16{ mag:26214, sign:true}, FP16x16{ mag:13107, sign:true}, FP16x16{
mag:0, sign:false}, FP16x16{ mag:13107, sign:false}, FP16x16{ mag:26214, sign:false}, FP16x16{ mag:39322, sign:false}, ]
.span() );

(classifier,X) }

fntest_tree_ensemble_classifier_multi_pt_softmax()->(Span, MutMatrix::) {
let(mutclassifier,X)=tree_ensemble_classifier_helper(POST_TRANSFORM::SOFTMAX);

let(labels, scores)=TreeEnsembleClassifierTrait::predict(refclassifier,X); (labels, scores) }

([0,0,1], [ [0.545123,0.217967,0.23691], [0.416047,0.284965,0.298988],
[0.322535,0.366664,0.310801], ])

...

```

[Previous Tree Ensemble Classifier](#) [Next Tree Ensemble Regressor](#)

Last updated 3 months ago