tree_ensemble_regressor.predict

```

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

```

Tree Ensemble regressor. Returns the regressed values for each input in N.

Args

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

Returns

- Regressed values for each input in N
-

Type Constraints

TreeEnsembleRegressor 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::tree_ensemble::tree_ensemble_regressor::{
TreeEnsembleRegressor,POST_TRANSFORM,TreeEnsembleRegressorTrait,AGGREGATE_FUNCTION };
useorion::operators::matrix::{MutMatrix,MutMatrixImpl};

fntree_ensemble_regressor_helper( agg:AGGREGATE_FUNCTION )->(TreeEnsembleRegressor,Tensor) {
letn_targets:usize=1; letaggregate_function=agg; letnodes_falsenodeids:Span=array![4,3,0,0,0,2,0,4,0,0].span();
letnodes_featureids:Span=array![0,2,0,0,0,0,0,2,0,0].span(); letnodes_missing_value_tracks_true:Span=array!
[0,0,0,0,0,0,0,0,0,0].span(); letnodes_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(); letnodes_nodeids:Span=array![0,1,2,3,4,0,1,2,3,4].span(); letnodes_treeids:Span=array!
[0,0,0,0,0,1,1,1,1,1].span(); letnodes_truenodeids:Span=array![1,2,0,0,0,1,0,3,0,0].span(); letnodes_values:Span=array![
FP16x16{ mag:17462, sign:false}, FP16x16{ mag:40726, sign:false}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:0,
sign:false}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:47240, sign:true}, FP16x16{ mag:0, sign:false}, FP16x16{
mag:36652, sign:true}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:0, sign:false} ] .span(); lettarget_ids:Span=array!
[0,0,0,0,0,0].span(); lettarget_nodeids:Span=array![2,3,4,1,3,4].span(); lettarget_treeids:Span=array![0,0,0,1,1,1].span();
lettarget_weights:Span=array![ FP16x16{ mag:5041, sign:false}, FP16x16{ mag:32768, sign:false}, FP16x16{ mag:32768,
sign:false}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:18724, sign:false}, FP16x16{ mag:32768, sign:false} ] .span();

letbase_values:Option>=Option::None; letpost_transform=POST_TRANSFORM::NONE;

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(208998628034825342117067982148086513282306647093844609550582231725359408128 4,0); node_index
.insert(200114008253061923966172980908457829829922381020209762276163238456111239097 9,1); node_index
.insert(259267024108419221235402744004908585279250651878195489614429631613179040390 0,2); node_index
.insert(296059127137682937835656780361854867203486734512372717862886942654845383342 0,3); node_index
.insert(458933264452572171106695256465341160654132084710250671055261382009315664425,4); node_index
.insert(108954991580026454962153690976769977874592651755558633277275928070239600910 8,5); node_index
.insert(132114200402299484568137729980140356737850353025046761034338159090983217118 0,6); node_index
.insert(259298785177596574254345931950834845729096625324145551422612763910045784477 4,7); node_index
.insert(249275562301908610903224721861596438972636853216065349703900581448439341934 8,8); node_index
.insert(132361602384570425811353834800004714947045008630773120072803960771031662591 6,9);
```

```
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 };

letmutregressor:TreeEnsembleRegressor=TreeEnsembleRegressor{ ensemble, target_ids, target_nodeids, target_treeids, target_weights, base_values, n_targets, aggregate_function, post_transform };

letmutX:Tensor=TensorTrait::new( array![3,3].span(), array![ FP16x16{ mag:32768, sign:true}, FP16x16{ mag:26214, sign:true}, FP16x16{ mag:19660, sign:true}, FP16x16{ mag:13107, sign:true}, FP16x16{ mag:6553, sign:true}, FP16x16{ mag:0, sign:false}, FP16x16{ mag:6553, sign:false}, FP16x16{ mag:13107, sign:false}, FP16x16{ mag:19660, sign:false}, ] .span() );

(regressor,X) }

fntest_tree_ensemble_regressor_SUM()->MutMatrix:: {
let(mutregressor,X)=tree_ensemble_regressor_helper(AGGREGATE_FUNCTION::SUM);
letmutres=TreeEnsembleRegressorTrait::predict(refregressor,X); res }

[0.5769,0.5769,0.5769]

```

Previous Tree Ensemble Regressor Next Linear Classifier

Last updated2 months ago