Deep Classification Trees (Experimental)

`mod` classifier

Deep-forest classification estimator built on Keras.

This module provides :class:`~tree_machine.deep_trees.classifier.DFClassifier`, an sklearn-compatible classifier that builds a differentiable forest model using Keras.

The estimator follows the familiar `fit` / `predict` / `predict_proba` / `score` API and supports both built-in and user-provided Keras losses/metrics.

`class` DFClassifier

Bases: `BaseDeep`, `ClassifierMixin`

A deep-forest classifier with an sklearn-compatible API.

Parameters:

Name	Type	Description	Default
`metric`	`AcceptableMetric`	Name of the built-in metric/loss key. Currently only `"cross_entropy"` is supported.	`'cross_entropy'`
`n_estimators`	`int`	Number of trees/estimators to build.	`100`
`internal_size`	`int`	Internal representation size used by the differentiable tree layers.	`12`
`max_depth`	`int`	Maximum depth of each differentiable tree.	`6`

`feature_fraction`	`float`	Fraction of features sampled per estimator.	`1.0`
`loss`	`LossLike | None`	Optional custom Keras loss (callable/instance/class). If omitted (and `metrics` is also omitted), the built-in loss derived from `metric` is used.	`None`
`metrics`	`Sequence[MetricLike] | None`	Optional sequence of custom Keras metrics (callables/instances/classes).	`None`
`compile_kwargs`	`dict[str, Any] | None`	Extra keyword arguments forwarded to `Model.compile`. If `compile_kwargs` does not define an optimizer, `"adam"` is used.	`None`
`decision_l1/decision_l2`		L1/L2 regularization strength applied to routing Dense weights.	*required*
`leaf_l1/leaf_l2`		L1/L2 regularization strength applied to leaf values.	*required*
`feature_dropout`	`float`	Dropout rate applied to inputs during training.	`0.0`
`routing_dropout`	`float`	Dropout rate applied to routing probabilities during training.	`0.0`

Notes

- Each call to :meth: `fit` builds and compiles a new Keras model.
- :meth: `score` returns the negative loss value on `(X, y)`.

Source code in `src/tree_machine/deep_trees/classifier.py`

▼ Details

```
149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
```

▼ Details

```
215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241
```

▼ Details

```
281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
```

▼ Details

▼ Details

▼ Details

```
 98
 99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
```

▼ Details

```
152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178
```

▼ Details

```
218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244
```

▼ Details

```
284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
```