Improving performance using scipy.sparse

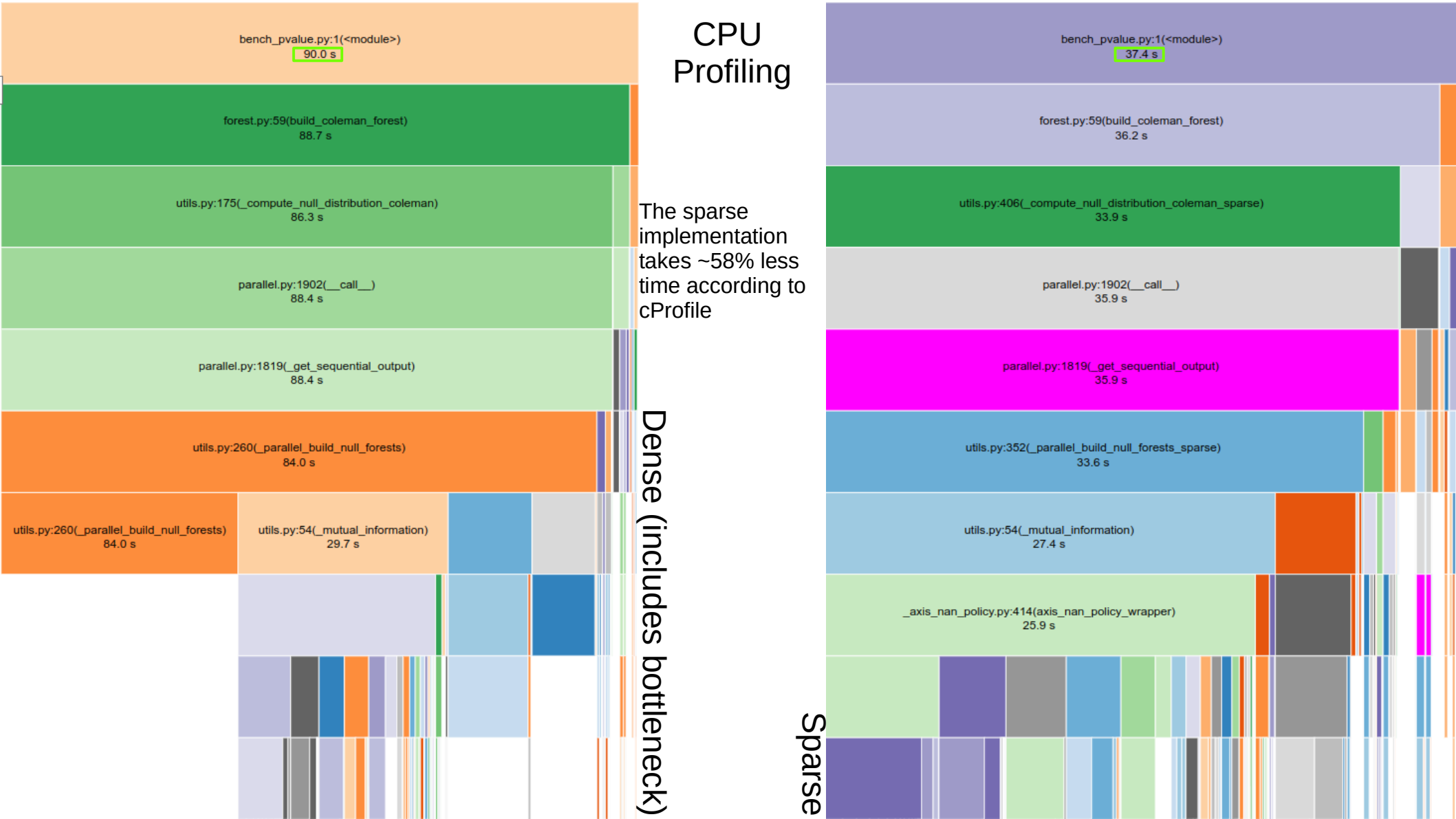
Note that this only works in the special case of binary classification or regression as scipy.sparse only supports 2d matrices.

This could be generalized using pydata.sparse but currently that uses too much memory.

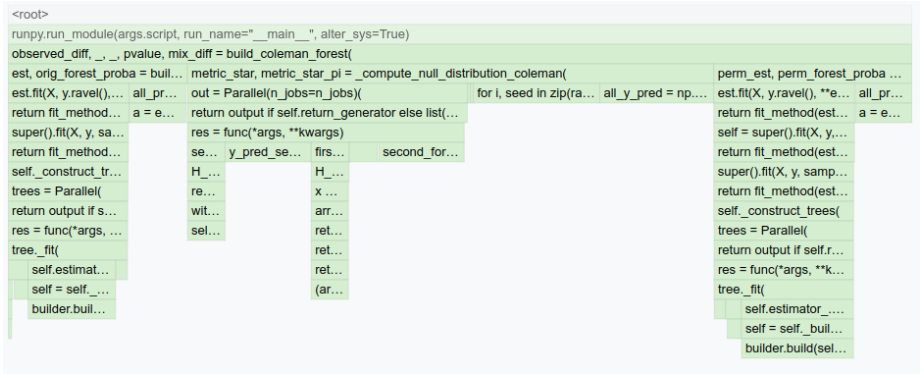
The next two slides show the CPU and Memory profiling analysis of the script to the right.

The profilers seem to indicate that using scipy.sparse cuts runtime by more than half and uses 7% less memory.

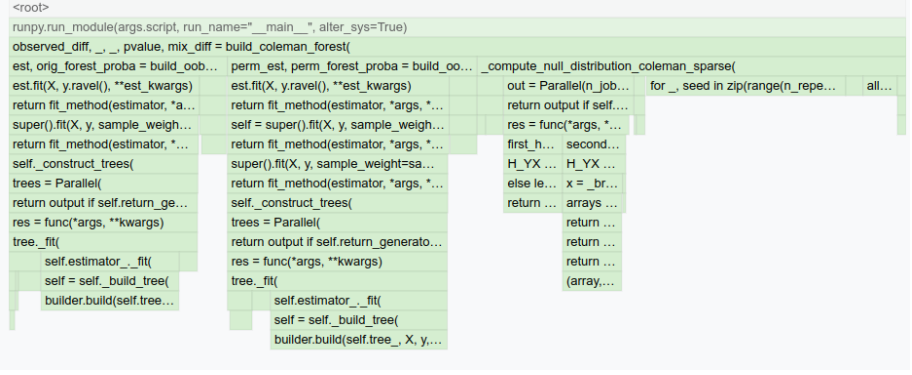
```python
# Test bench mark different approaches to the pvalue calculation to figure out
# the best way to calculate the pvalue.
# code is based on examples/treeple/treeple_tutorial_1_2_pvalue.py

import sys
from treeple.datasets import make_trunk_classification
from treeple.ensemble import HonestForestClassifier
from treeple.stats import PermutationHonestForestClassifier, build_coleman_forest

N_PERMUTATIONS = 10000

X, y = make_trunk_classification(
    n_samples=1000,
    n_dim=1,
    mu_0=0,
    mu_1=1,
    n_informative=1,
    seed=1,
)

est = HonestForestClassifier(
    n_estimators=100,
    max_samples=1.6,
    max_features=0.3,
    bootstrap=True,
    stratify=True,
    random_state=1,
)

est_null = PermutationHonestForestClassifier(
    n_estimators=100,
    max_samples=1.6,
    max_features=0.3,
    bootstrap=True,
    stratify=True,
    random_state=1,
)


observed_diff, _, _, pvalue, mix_diff = build_coleman_forest(
    est,
    est_null,
    X,
    y,
    metric="mi",
    n_repeats=N_PERMUTATIONS,
    return_posteriors=False,
    seed=1,
    use_sparse="sparse" in sys.argv,
)
```


CPU Profiling
bench_pvalue.py:1(<module>)
90.0 s
forest.py:59(build_coleman_forest)
88.7 s
utils.py:175(_compute_null_distribution_coleman)
86.3 s
parallel.py:1902(__call__)
88.4 s
parallel.py:1819(_get_sequential_output)
88.4 s
utils.py:260(_parallel_build_null_forests)
84.0 s
utils.py:260(_parallel_build_null_forests)
84.0 s
utils.py:54(_mutual_information)
29.7 s
The sparse implementation takes ~58% less time according to cProfile
Dense (includes bottleneck)
bench_pvalue.py:1(<module>)
37.4 s
forest.py:59(build_coleman_forest)
36.2 s
utils.py:406(_compute_null_distribution_coleman_sparse)
33.9 s
parallel.py:1902(__call__)
35.9 s
parallel.py:1819(_get_sequential_output)
35.9 s
utils.py:352(_parallel_build_null_forests_sparse)
33.6 s
utils.py:54(_mutual_information)
27.4 s
_axis_nan_policy.py:414(axis_nan_policy_wrapper)
25.9 s
Sparse

Memory Profiling using memray



Dense
Total for benchmarking script: ~129MB
Total for build_coleman_forest: ~24.2MB



Sparse
Total for benchmarking script: ~120.1MB
Total for build_coleman_forest: ~15.2MB