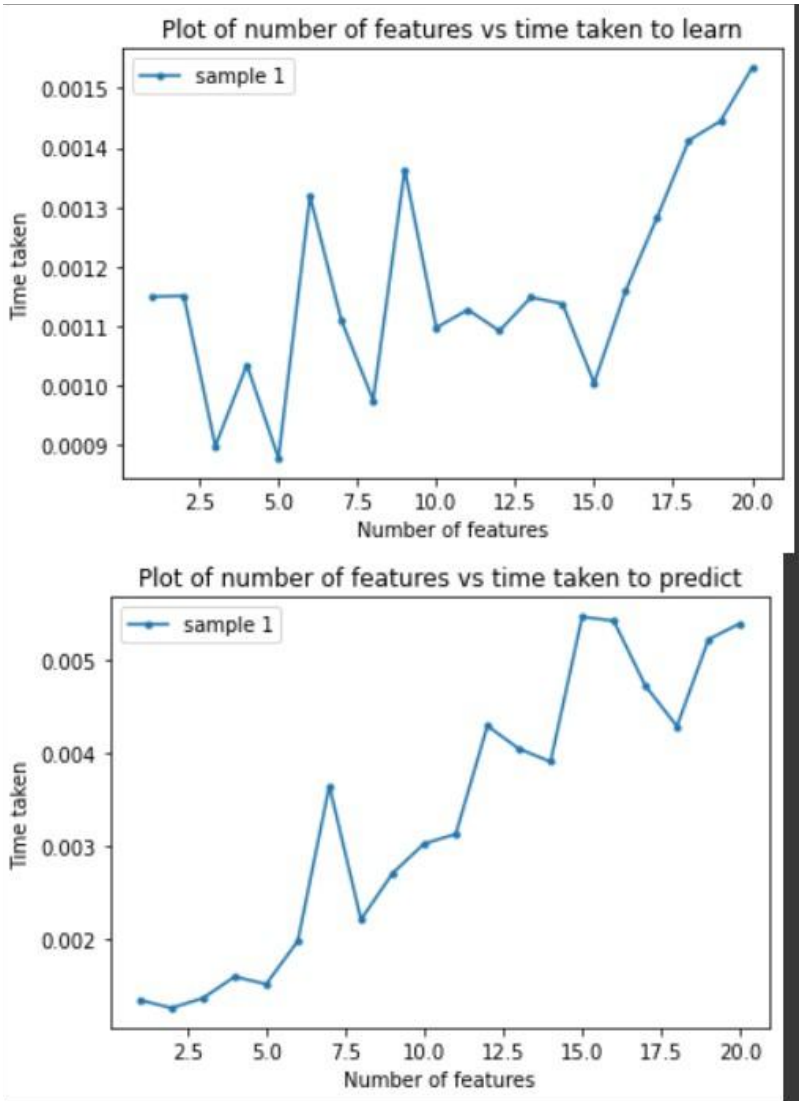
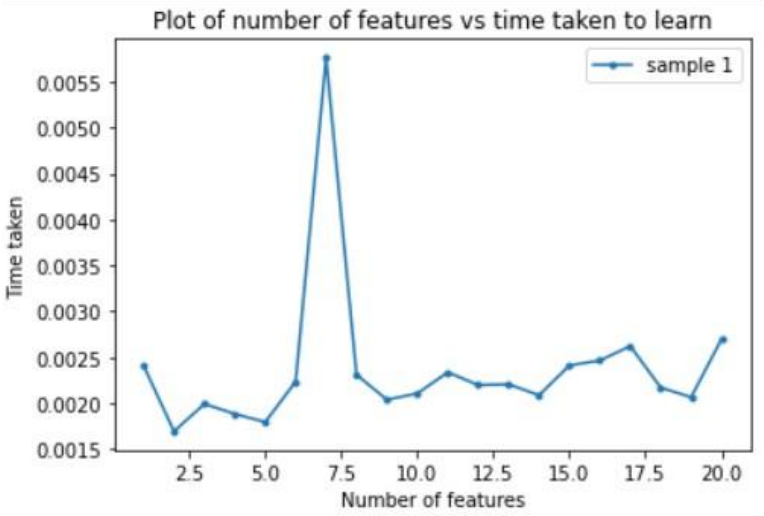
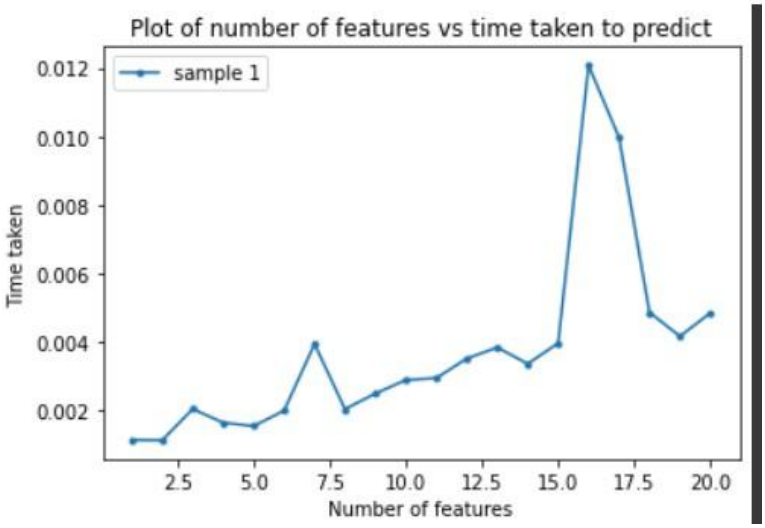
Learning and Predicting Time for Discrete Input Discrete Output

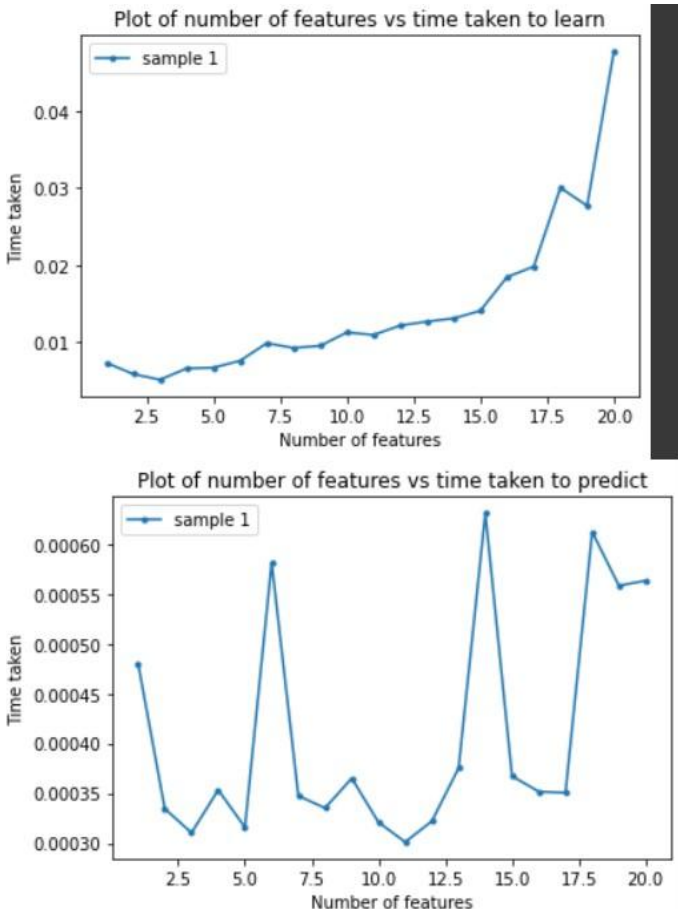


Learning and Predicting Time for Discrete Input Real Output

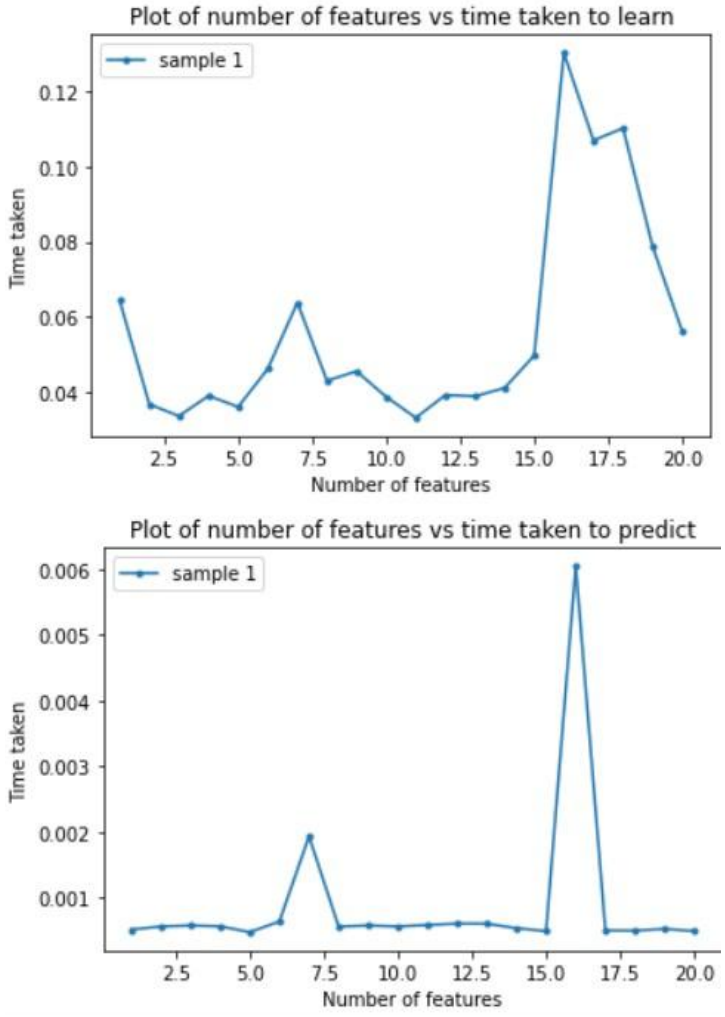




Learning and Predicting Time for Real Input Discrete Output



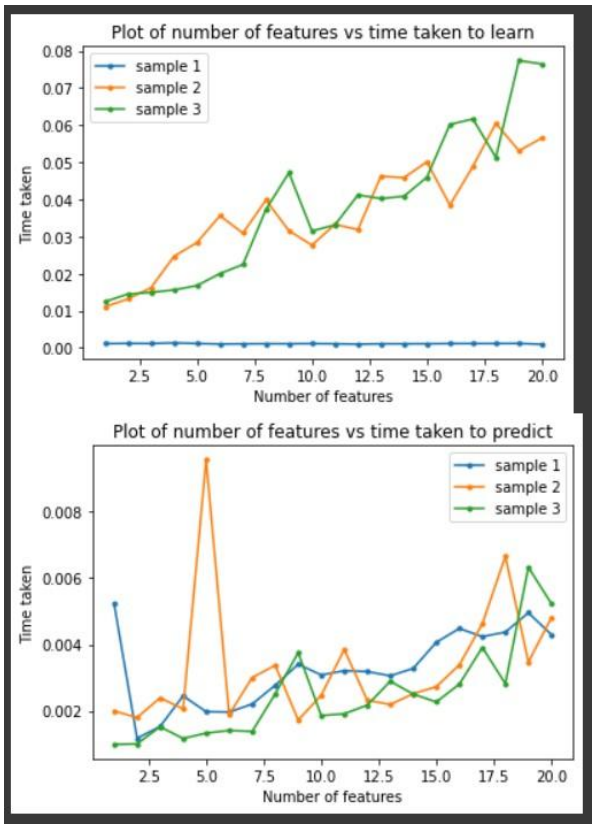
Learning and Predicting Time for Real Input Real Output

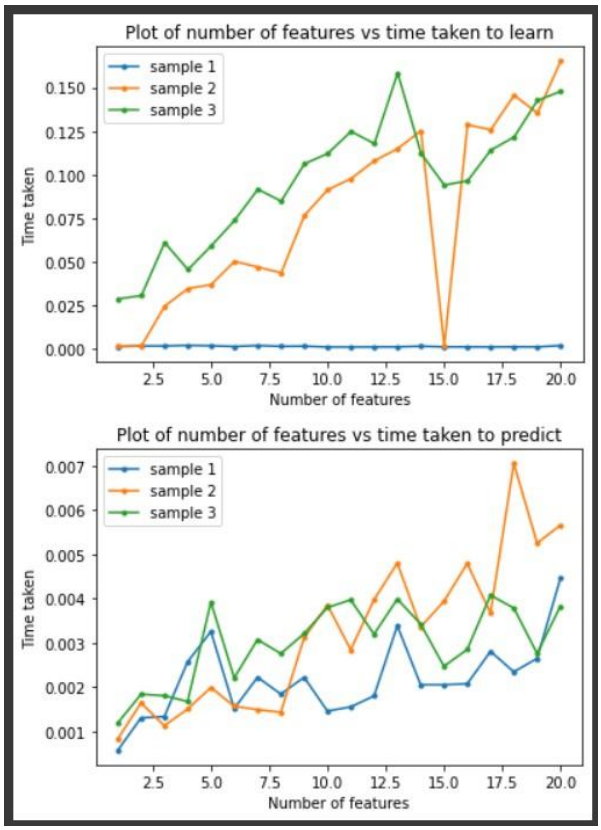


In our code, we have varied N and M simultaneously for N=3 and M=20 (images in this doc for N=1)

Images for N=3, M=20 :

DIDO and DIRO respectively





For learning time:

We can observe that the graphs overall are increasing in nature, excluding some features (the trend for sample 1 in the above graphs seems constant, but is increasing if we check it individually). Still, the overall trendline is a logarithmic increase in varying N values for each case. We observe that the logarithmic function depends on N; thus, it is a multiple of log(N), and as initially the varying components are N and M; thus, the time complexity of fitting or learning should be O(M*N*log(N))

This can be backed by theoretical time complexity for fitting a decision tree is also O(mn*logn).

For predicting time:

The prediction time complexity is O(NlogN*depth) since we need to traverse the deepest leaf in the worst case.

But as we maintain the depth as a constant in our code, the variation is logarithmic with N, which can be seen in increasing the sample size. We don't observe much change with changes in M, mostly because we are using them as criteria for splitting and not iterating over them. Thus the time complexity for predicting according to our model, barring occasional arbitrary deviations is (NlogN*depth)