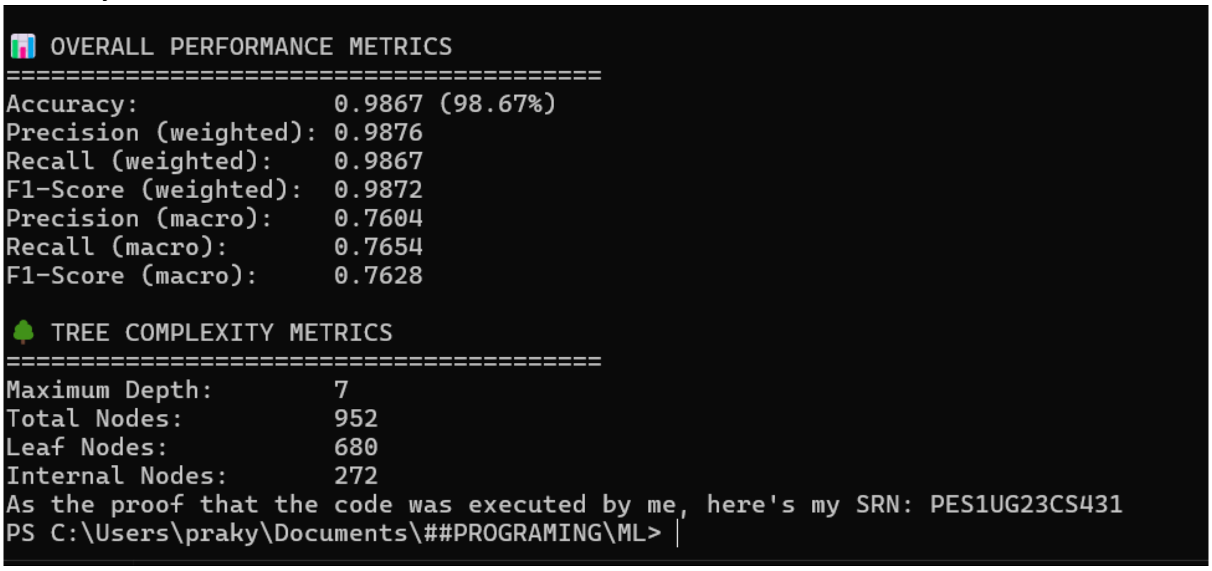
Machine Learning Lab 3 - Report

Name: Prakyath P Nayak
SRN: PES1UG23CS431
Section: G

Objective
Implement the ID3 Decision Tree algorithm and perform comparative analysis across three diverse datasets to understand algorithm performance under different data characteristics.

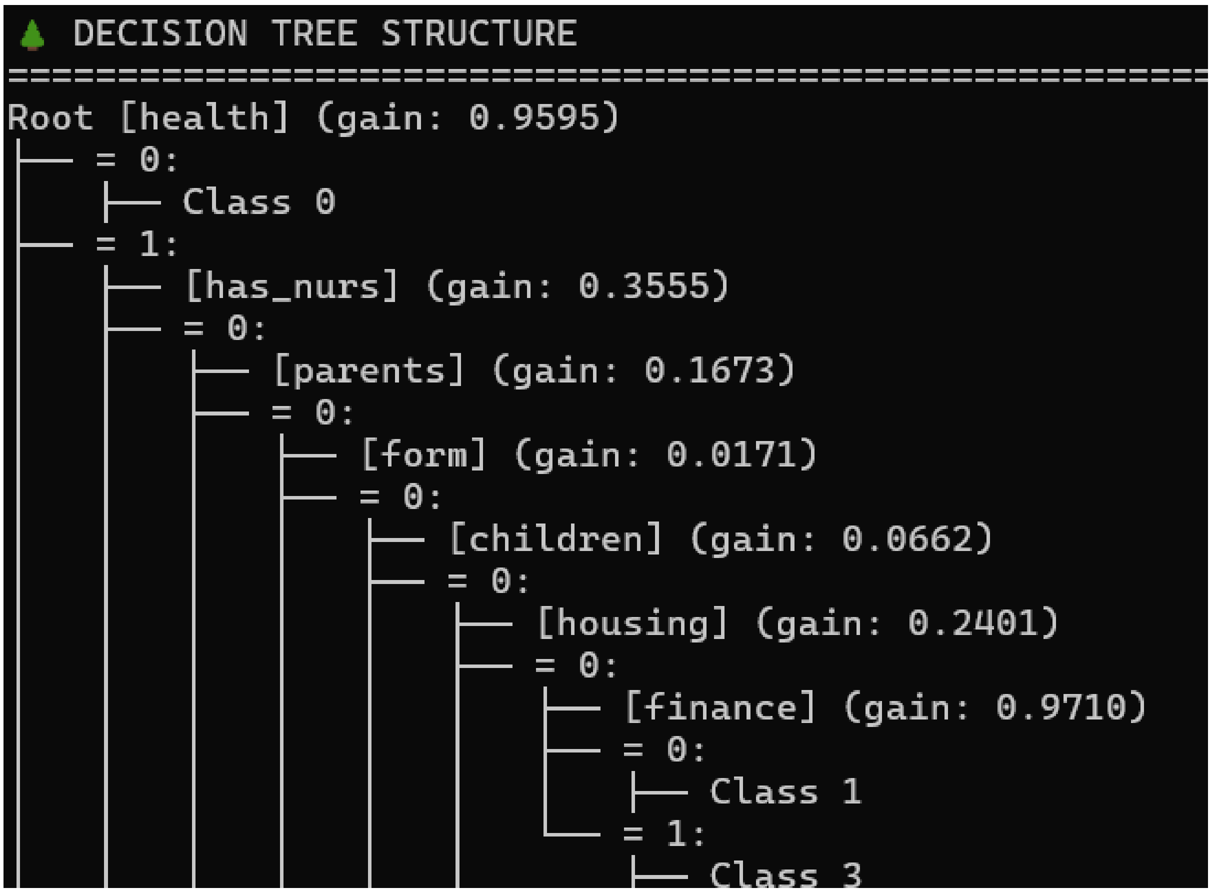
Abstract
The following report showcases the work around the provided problem statement. We were provided with 3 datasets and were told to explore decision trees using these datasets. We were also provided with two code files, one of which contained just function definitions and some instructions on how to implement the functions. The other file was a tester for the code which was to be written by us. Due to the emojis in the provided code, there were UTF-8 formatting issues. Hence, all of the code was executed on my local machine. Find below the detailed breakdown of my approach towards this.

Performance Analysis
I ended up using the boilerplate code provided by the faculty as we didn't have enough time to learn and implement the entire thing on our own. Here are the performance metrics for all the datasets that were provided.
Nursery.csv:



```
📊 OVERALL PERFORMANCE METRICS
========================================
Accuracy:             0.9867 (98.67%)
Precision (weighted): 0.9876
Recall (weighted):    0.9867
F1-Score (weighted):  0.9872
Precision (macro):    0.7604
Recall (macro):       0.7654
F1-Score (macro):     0.7628

🌳 TREE COMPLEXITY METRICS
========================================
Maximum Depth:        7
Total Nodes:          952
Leaf Nodes:           680
Internal Nodes:       272
As the proof that the code was executed by me, here's my SRN: PES1UG23CS431
PS C:\Users\praky\Documents\##PROGRAMING\ML> |
```

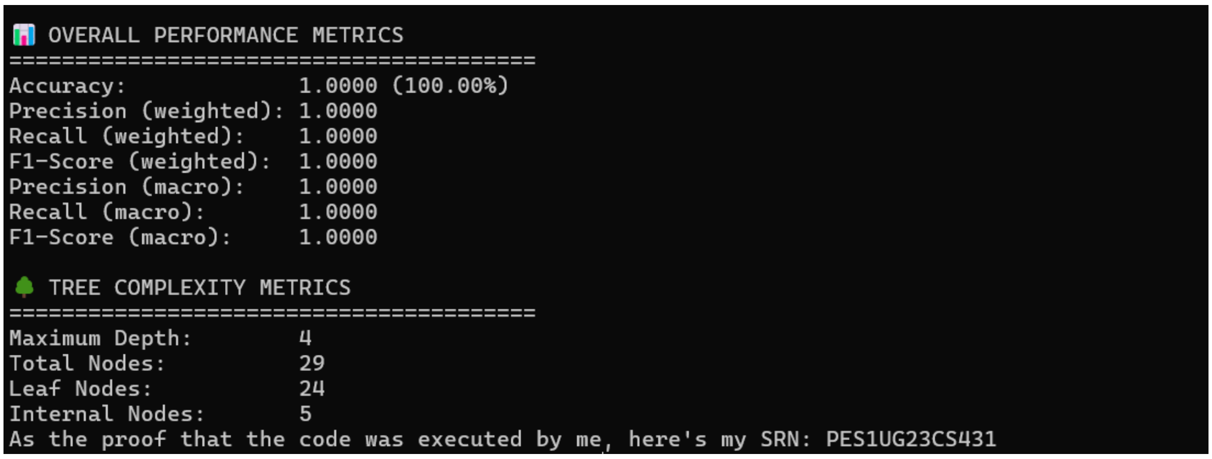
It can be seen that the model has a high accuracy over the testing data and the tree height is only 7. Keeping in mind that the dataset consisted of more than 12 thousand data points, that's very impressive. But we can see that the precision is considerably lower. That means that while the model is extremely good at generalizing over the major classes, the minority classes may require some additional balancing. The large size of the tree can be justified with the large amount of classes present per attribute in the dataset.



```
🌲 DECISION TREE STRUCTURE
==============================================================
Root [health] (gain: 0.9595)
├── = 0:
│   ├── Class 0
├── = 1:
│   ├── [has_nurs] (gain: 0.3555)
│   ├── = 0:
│   │   ├── [parents] (gain: 0.1673)
│   │   ├── = 0:
│   │   │   ├── [form] (gain: 0.0171)
│   │   │   ├── = 0:
│   │   │   │   ├── [children] (gain: 0.0662)
│   │   │   │   ├── = 0:
│   │   │   │   │   ├── [housing] (gain: 0.2401)
│   │   │   │   │   ├── = 0:
│   │   │   │   │   │   ├── [finance] (gain: 0.9710)
│   │   │   │   │   │   ├── = 0:
│   │   │   │   │   │   │   ├── Class 1
│   │   │   │   │   │   └── = 1:
│   │   │   │   │   │   │   ├── Class 3
```

By some coincidence or maybe some factor which was embedded onto the dataset by human judgement, health was the most significant attribute. It could be that health only had two possible values. Usually, when all other attributes have a high number of classes and some one or two attributes only have two classes. They often get selected to be the root nodes. This is because the fewer classes an attribute has, the more information that it gives on the subject (if the attribute was relevant). More inspection is probably needed on this. But unfortunately, I'm ill equipped with techniques to make those inspections.

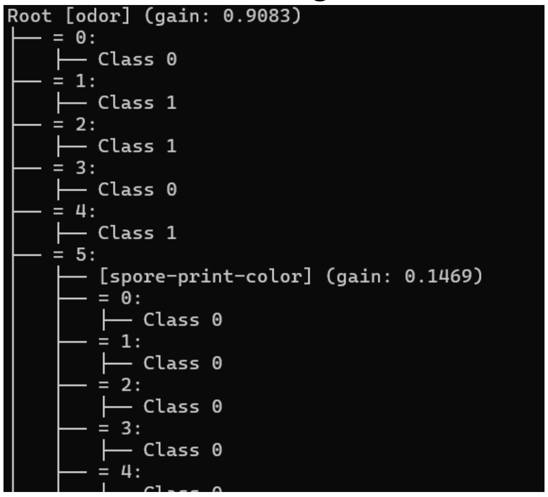
mushrooms.csv:



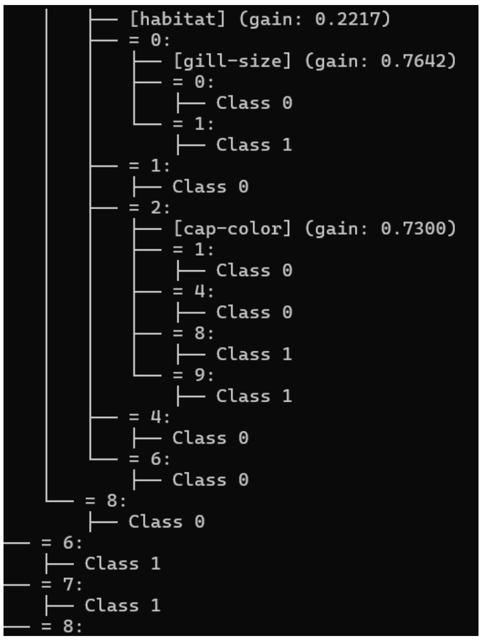
```
📊 OVERALL PERFORMANCE METRICS
========================================
Accuracy:             1.0000 (100.00%)
Precision (weighted): 1.0000
Recall (weighted):    1.0000
F1-Score (weighted):  1.0000
Precision (macro):    1.0000
Recall (macro):       1.0000
F1-Score (macro):     1.0000

🌳 TREE COMPLEXITY METRICS
========================================
Maximum Depth:        4
Total Nodes:          29
Leaf Nodes:           24
Internal Nodes:       5
As the proof that the code was executed by me, here's my SRN: PES1UG23CS431
```

As seen in the image, the model seems to have achieved 100% accuracy over the training dataset. The dataset contained more than eight thousand data points. So it could be that there were a lot of duplicate data points and the data points accurately resemble the target distribution (or a very small chance that the entire dataset is somehow biased). The duplicates being present in the dataset can also be justified by looking at the maximum depth and the total nodes in the tree, which is very small compared to the huge dataset that we had.

Now, look at the images below:



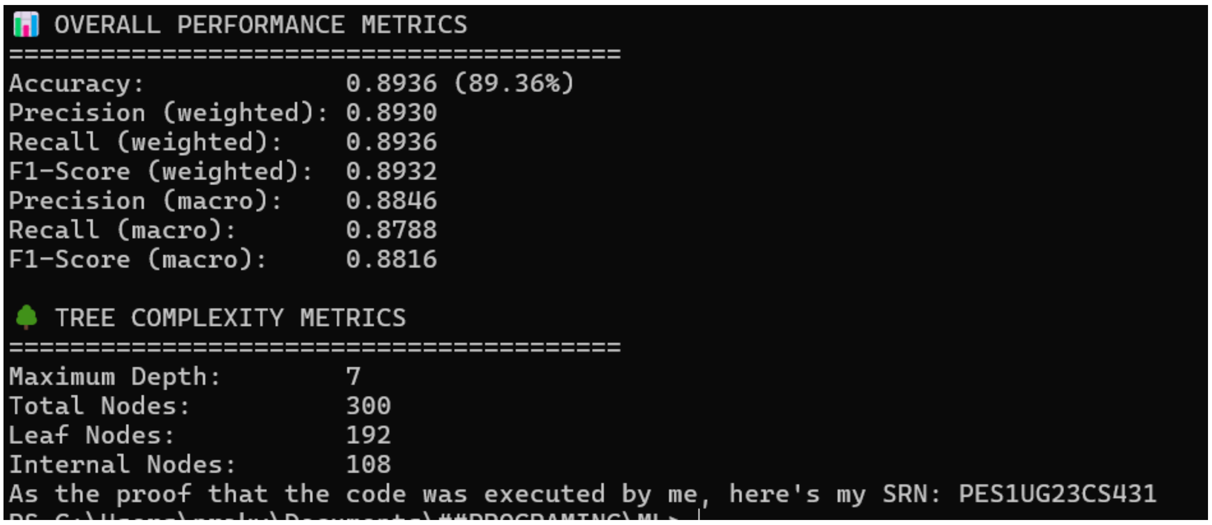
```
Root [odor] (gain: 0.9083)
├── = 0:
│   ├── Class 0
├── = 1:
│   ├── Class 1
├── = 2:
│   ├── Class 1
├── = 3:
│   ├── Class 0
├── = 4:
│   ├── Class 1
├── = 5:
│   ├── [spore-print-color] (gain: 0.1469)
│   ├── = 0:
│   │   ├── Class 0
│   ├── = 1:
│   │   ├── Class 0
│   ├── = 2:
│   │   ├── Class 0
│   ├── = 3:
│   │   ├── Class 0
│   ├── = 4:
```



```
│   │   ├── [habitat] (gain: 0.2217)
│   │   ├── = 0:
│   │   │   ├── [gill-size] (gain: 0.7642)
│   │   │   ├── = 0:
│   │   │   │   ├── Class 0
│   │   │   └── = 1:
│   │   │       ├── Class 1
│   │   ├── = 1:
│   │   │   ├── Class 0
│   │   ├── = 2:
│   │   │   ├── [cap-color] (gain: 0.7300)
│   │   │   ├── = 1:
│   │   │   │   ├── Class 0
│   │   │   ├── = 4:
│   │   │   │   ├── Class 0
│   │   │   ├── = 8:
│   │   │   │   ├── Class 1
│   │   │   └── = 9:
│   │   │       ├── Class 1
│   │   ├── = 4:
│   │   │   ├── Class 0
│   │   └── = 6:
│   │       ├── Class 0
│   └── = 8:
│       ├── Class 0
├── = 6:
│   ├── Class 1
├── = 7:
│   ├── Class 1
├── = 8:
```

The odor was the most significant attribute. But you can see that almost all branches directly lead to classification. I am inclined to believe that there was some sampling error or some bias as this isn't and most of the time shouldn't be seen in any decision tree which was made with 8k data points.

tictactoe.csv:



```
📊 OVERALL PERFORMANCE METRICS
========================================
Accuracy:             0.8936 (89.36%)
Precision (weighted): 0.8930
Recall (weighted):    0.8936
F1-Score (weighted):  0.8932
Precision (macro):    0.8846
Recall (macro):       0.8788
F1-Score (macro):     0.8816

🌳 TREE COMPLEXITY METRICS
========================================
Maximum Depth:        7
Total Nodes:          300
Leaf Nodes:           192
Internal Nodes:       108
As the proof that the code was executed by me, here's my SRN: PES1UG23CS431
```

The accuracy seems poor here. But by messing around with a few things in the test.py, I found out that this was the best that was achievable with decision trees. The dataset only had 900 or so data points. But it can be seen that the accuracy is pretty much the same over all the metrics. Thus it is evident that the training set represents the total dataset really well, but fails to have data points over some specific class. The tree seems to be more complex than the rest while considering the data set size. The number of classes is much higher than the rest. That is also evident when we see that the tree is very wide as well.

Another aspect that I found interesting here was that the first split was on middle-middle-square. That makes complete logical sense as it has the most "control" over the board (or adjacency). Hence, it was the most contributing attribute.

Comparative Analysis Report

The mushroom dataset achieved the highest accuracy, being 100%. It could be that the odor greatly contributed to the overall accuracy or there was a bias in the dataset. The lowest accuracy was achieved by the tictactoe dataset. It could be that the number of samples was way lesser compared to others or that the testing dataset was missing some information on the entire distribution which the decision tree couldn't generalize over. The Nursery dataset had the least precision. That was probably due to the skewed distribution of data points across the classes. In the end, the model couldn't generalize over the minority classes.

I noticed that the lesser number of classes an attribute has, the more accurate the model becomes. It is probably because the number of datasets needed for making assumptions on the entire distribution depends on the number of classes that attribute has. In simple, the more the classes, the higher the number of "information" that we have to have to represent that attribute.

Practical Applications

The tic tac toe dataset could probably be used to train an RL model on state values. Other than that, both the other datasets were related to real world applications which could be used to save lives. But then again, for both of those, the data attributes are dependent on human measure, which could have a high bias.

Future Improvements

As I have mentioned before, I am currently ill-equipped to make significant improvements on the methods that were provided to us. The most I would be able to do is apply pruning methods or maybe somehow precompute the importance of the attributes and incorporate that into the entropy measure.