**NAME: MAHESH GAYA**

**PROJECT 2 – MACHINE LEARNING**

1. **(10 points) In class, an example was shown for how to use the decision tree, using only 3 columns for prediction. Update it to use all of the categorical columns.**

**Discuss what you did and record how well the algorithm performs in your write-up (be specific - what was the actual accuracy, not a vague opinion of how well you thought it did or "about" what the accuracy was).**

1. **(15 points) Enhance the decision tree code so that the best attribute is selected based on the attribute with the highest information gain (rather than randomly as it is in the starter code). Discuss what you did and record the performance in your write-up.**
2. **(10 points) Do at least one of the following. Discuss what you did and record the performance in your write-up.**

**If you do more than one, I will consider awarding a small number of extra credit points, but make sure to point out that you're doing extra in your write-up.**

1. **Bin up the numerical columns and use them for training and prediction.**
2. **Change the choose\_attribute() function so that it can deal with numerical columns in the middle of training - split at a good place that gives high information gain (if you do it this way, and you do it well, it could be worth some extra credit by itself)**
3. **Implement some kind of early stopping that improves performance.**
4. **Implement some kind of pruning that improves performance (if you do it this way, and you do it well, it could be worth some extra credit by itself).**
5. The accuracy for the original algorithm with attributes (“Status of existing checking account”, “Credit history”, “Purpose”) was 0.7. When we added the rest of the categorical attributes, the accuracy dropped a little bit. The accuracy for the original algorithm with all the categorical attributes was found to be 0.67. When there are less attributes to work with, a decision tree will reach an answer in less time and will also have a higher accuracy. However, it does not always say a lot about the full data, as several other attributes can have more impact the final decision.
6. There were a lot of things I tried. At first, I struggled with different errors because I had little knowledge about the Pandas library, but I learned a lot while debugging those errors. Then, to make things simpler, I used the whole training data instead of the one that is different for each node (I removed that chunk of code for clarity in the source code). This helped me understand what was actually going on behind the scenes. I was able to implement the code for choosing the attribute with the highest information gain. I will explain what I did exactly.

First, I looped through the whole list of the training data by column. For each attribute, I found their unique values and found the number of good or bad (in terms of creditability) rows. Next, I looped through each of the unique values and found out how many have good and bad creditability.

Then I calculated the entropy:

*entropy = - p\_good \* log2(p\_good) - p\_bad \* log2(p\_bad)*

I also calculate the expected entropy with the information gain:

*information\_gain = parent\_entropy - (total\_count\_per\_unique\_value[i] / total\_row\_count) \* entropy[i]*

Next, I used the max entropy for the next parent (I know that this is not the best way to find the entropy for parent, but I did not know any other way). Then I selected the attribute based on the highest information gain.

After, I did this using the full training data for test, I re-implemented the whole algorithm for the predictor data. Basically, this list gets trimmed down every time the tree gets a new node, and the algorithm bases its prediction only on that data instead of the whole training set.

I got an accuracy of about 0.59 on average (each time I ran it, it kept fluctuating from 0.57 to 0.61).

Things that could be done to improve accuracy in the future:

* 1. Improve on the algorithm to find the parent entropy
  2. Cut the tree if there is no useful information about the data
  3. Have a post pruning and check the accuracy for each node
  4. Improve the algorithm to find the best attribute without reusing the attributes that have already been used.

1. I binned up the numerical data into four categories. I tried to preserve the style that the author had. For instance, I used "A21", "A22", "A23", "A24" for “Duration in month”. I did the same for the rest of the numerical columns but I based the numbering on the column index. Then, I added those attributes to the tree, and I got an accuracy of about 0.63. This is a significant improvement on the previous averaged accuracy (0.59).

Things that could be improved:

* 1. Add more or less categories, depending on the resulting accuracy, for some of the numerical data. Some of them have a greater range than others.