**Machine Learning CS 529 Project 1 Report**

**Esteban Guillen**

**High-level Description of code**

I implemented the ID3 algorithm in Python (Figure 1). All code in contained in a file called ***id3.py***. The code follows the algorithm ( <https://en.wikipedia.org/wiki/ID3_algorithm> ) closely and adds the ability to switch between entropy and misclassification error for the impurity measure. Split stopping using the Chi-squared test was also added.

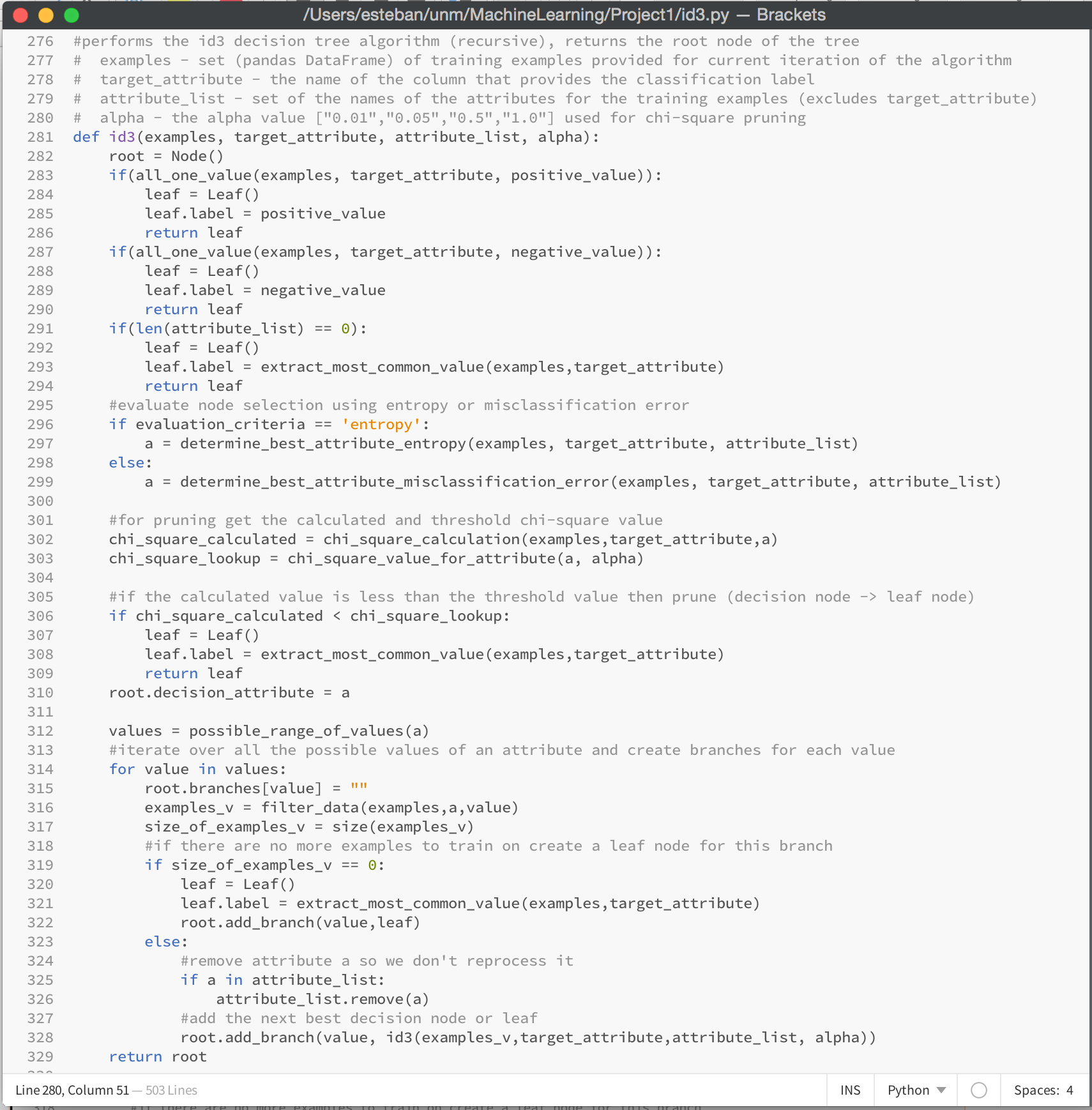


Figure 1. ID3 Algorithm

I used pandas DataFrame data structures for holding the training, testing and validation data (Figure 2 shows training and testing data being loaded). The DataFrame made it easy to index into the data and create subsets based on attribute values.

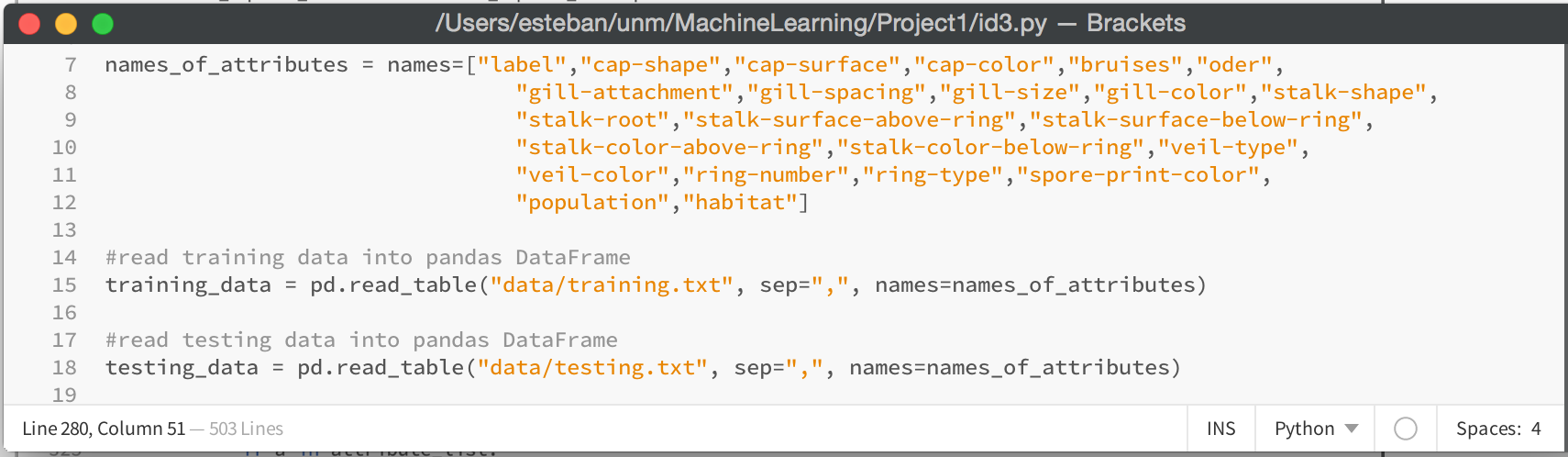


Figure 2. Using pandas DataFrame

I used two types of objects to represent my decision tree (Figure 3). ***Node*** which is a decision node that consists of a string representing the decision attribute and a dictionary to manage the branches and child nodes. ***Leaf*** which simply consists of a string that represents the classification label (e or p).

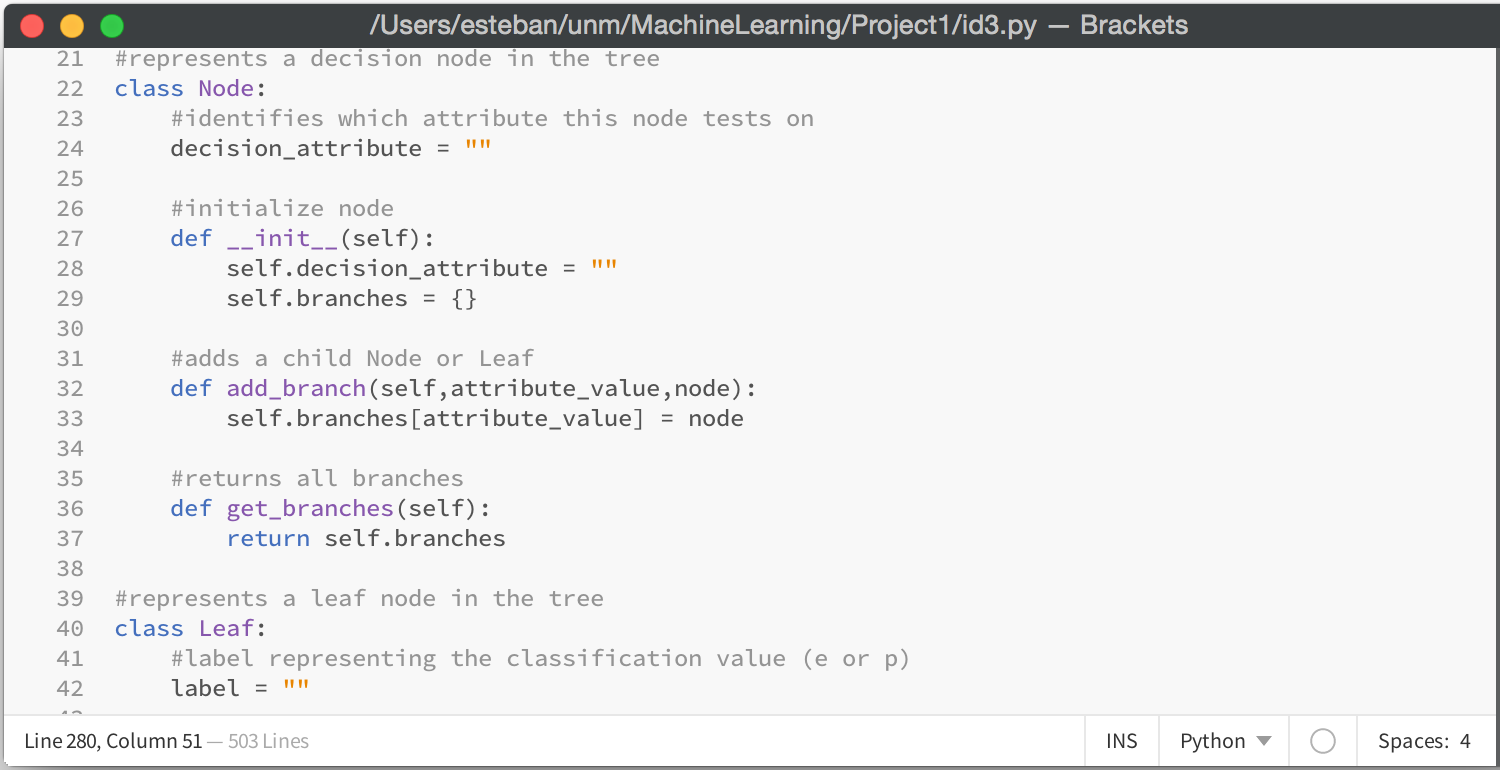


Figure 3. Objects used to represent the decision tree

The calculations for entropy and misclassification error can be seen below (Figure 4). The calculations were very straight forward (entropy used the log function from the math library).

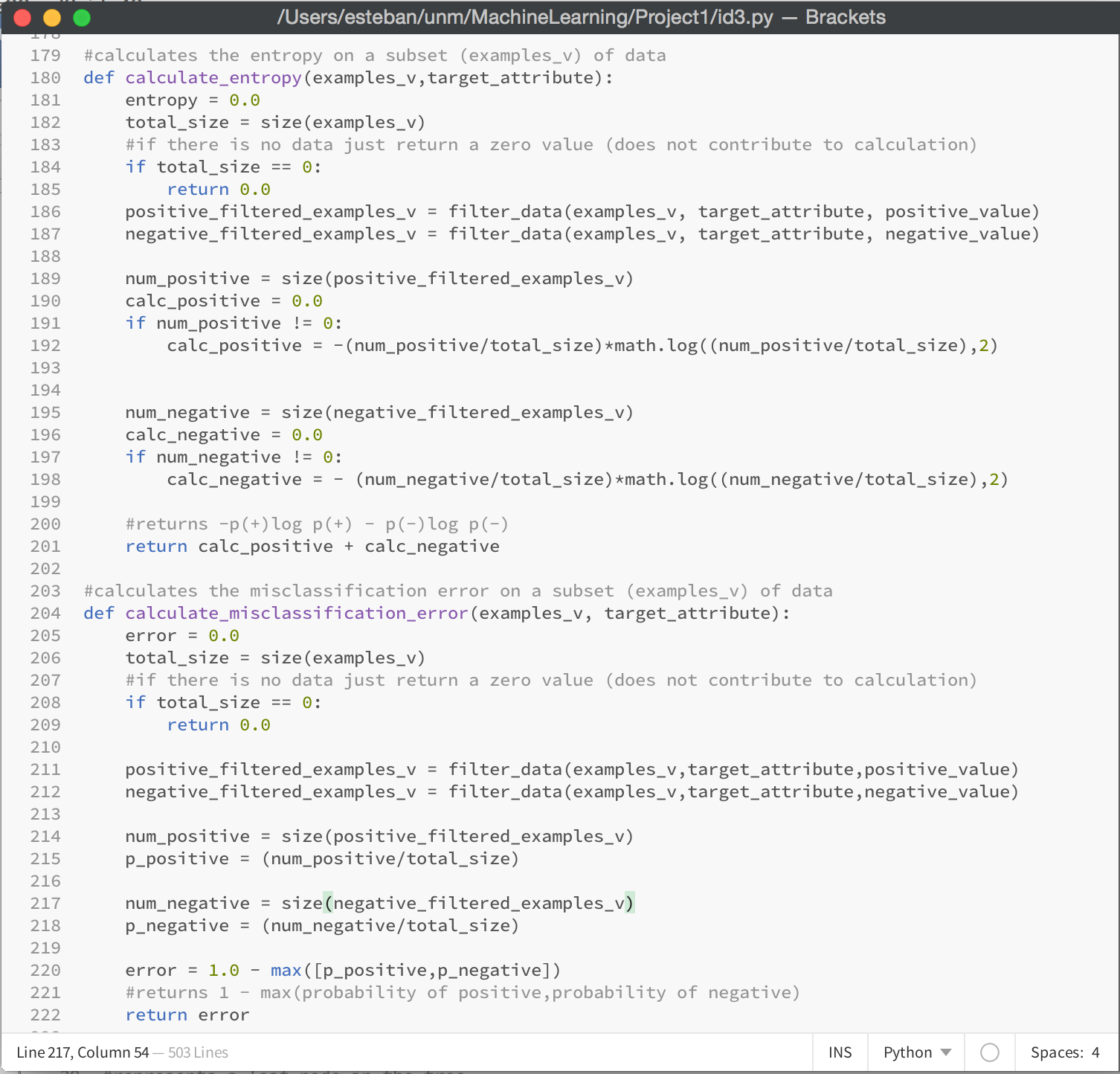


Figure 4. Entropy and misclassification calculations

Gain was used to determine the “best attribute” for each recursive call of the ID3 algorithm. The implementations for entropy and misclassification can be seen below (Figure 5).

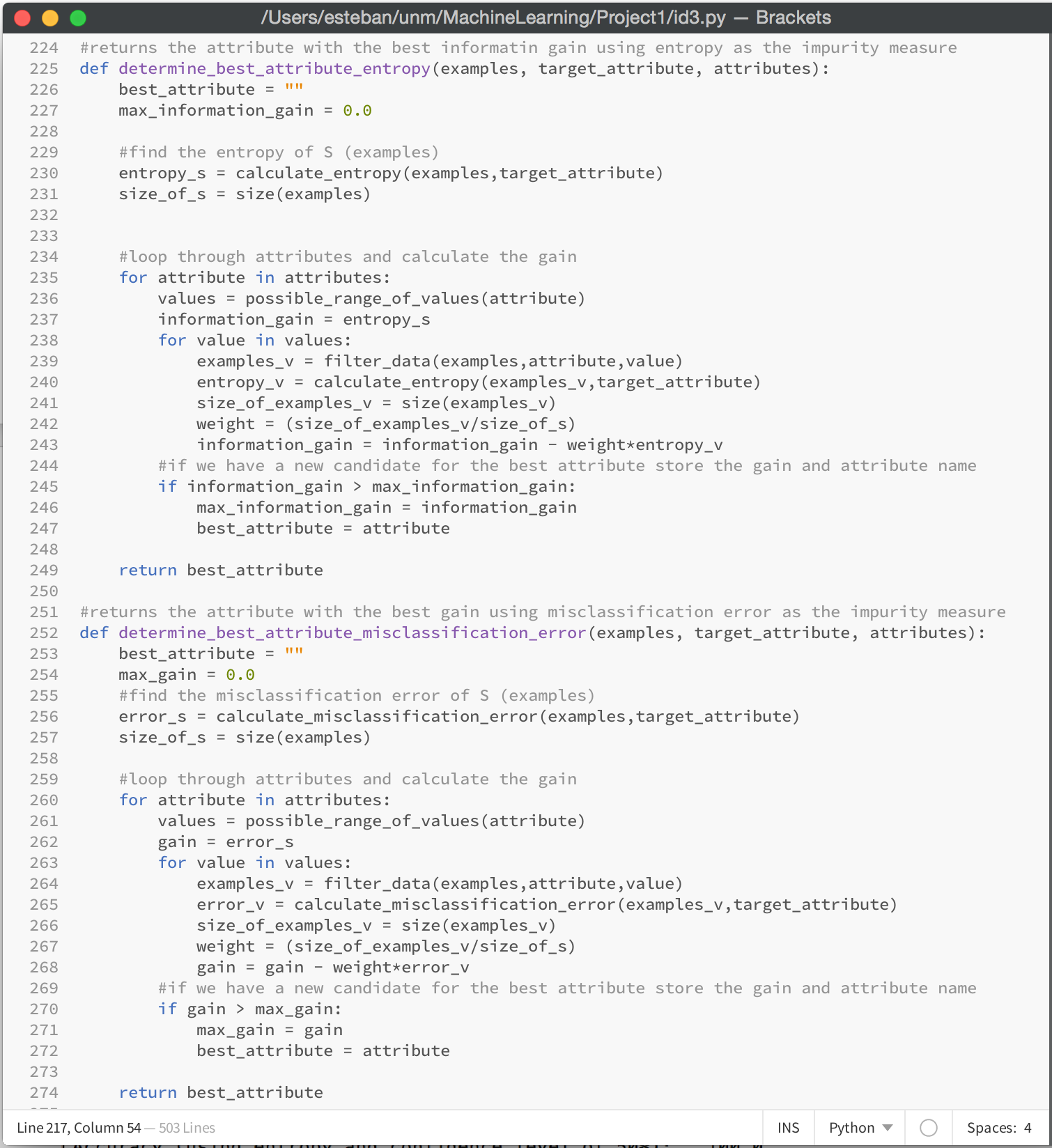


Figure 5. Gain calculations for entropy and misclassification

The Chi-square test was used for split stopping and the code for calculating the Chi-square value is shown below (Figure 6). The calculation was broken up into 2 functions, the first calculates the Chi-square for a specific value of an attribute and the second sums all those calculations (over all possible values of an attribute).

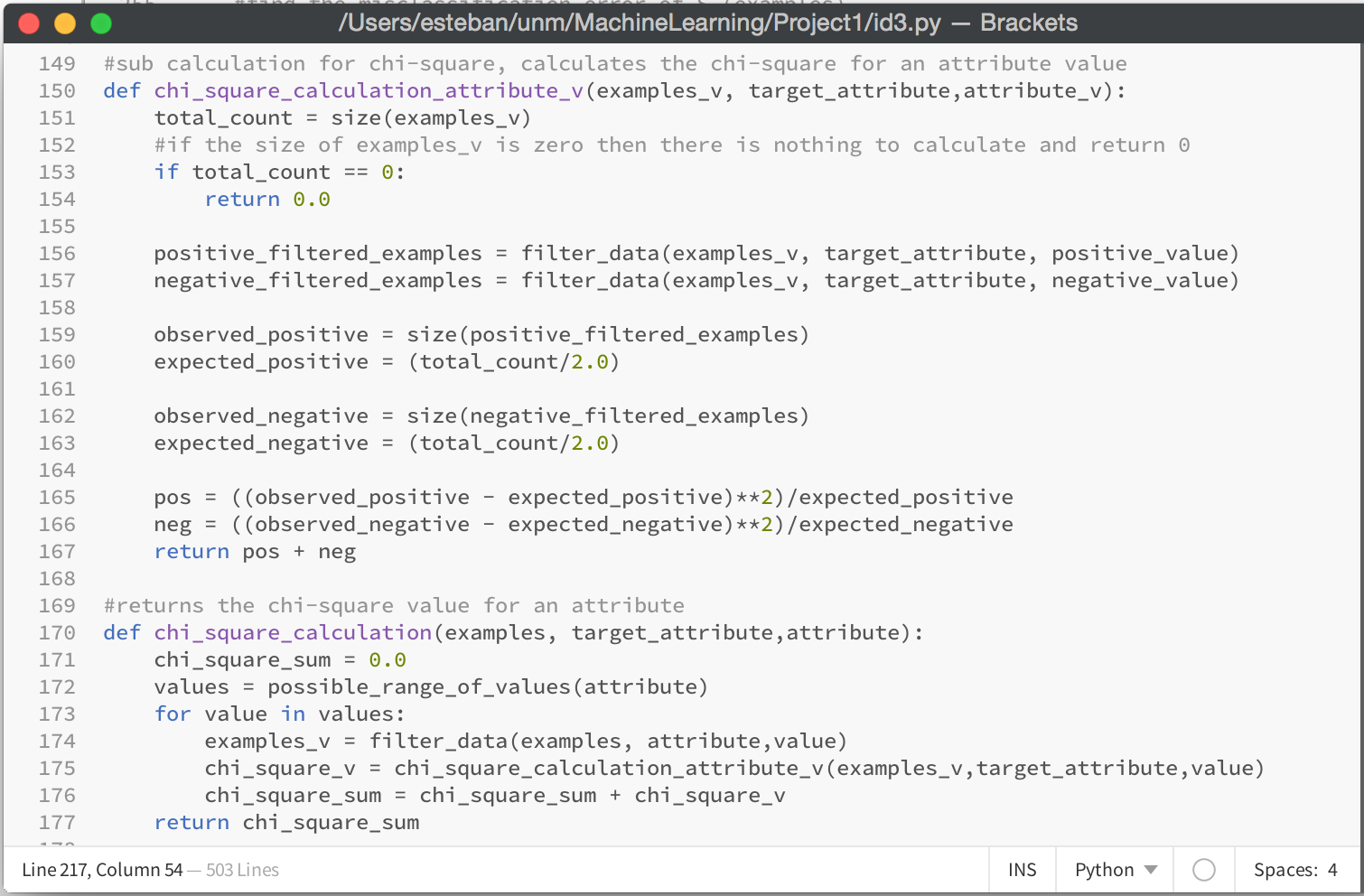


Figure 6. Chi-square calculation code

I implemented a number of helper functions to make the code more readable and maintainable. Check the full source code (***id3.py***) for more details.

**Accuracy Results**

All of my decision trees (entropy and misclassification error for 99, 95, 50, and 0 CL) produced 100% accuracy results (Figure 7). None of my trees got pruned using the Chi-squared test, all of my calculated values were less then the threshold values. The trees that were produced were small (4 levels for entropy and 5 levels for misclassification error, with most decision nodes having only one branch pointing to another decision node). The training data provided must have produced a near optimal tree, and no amount of pruning would improve performance (can’t get better than 100%). Most of the training data was classified at the root node and 8 of the 9 branches pointed to Leaf nodes.

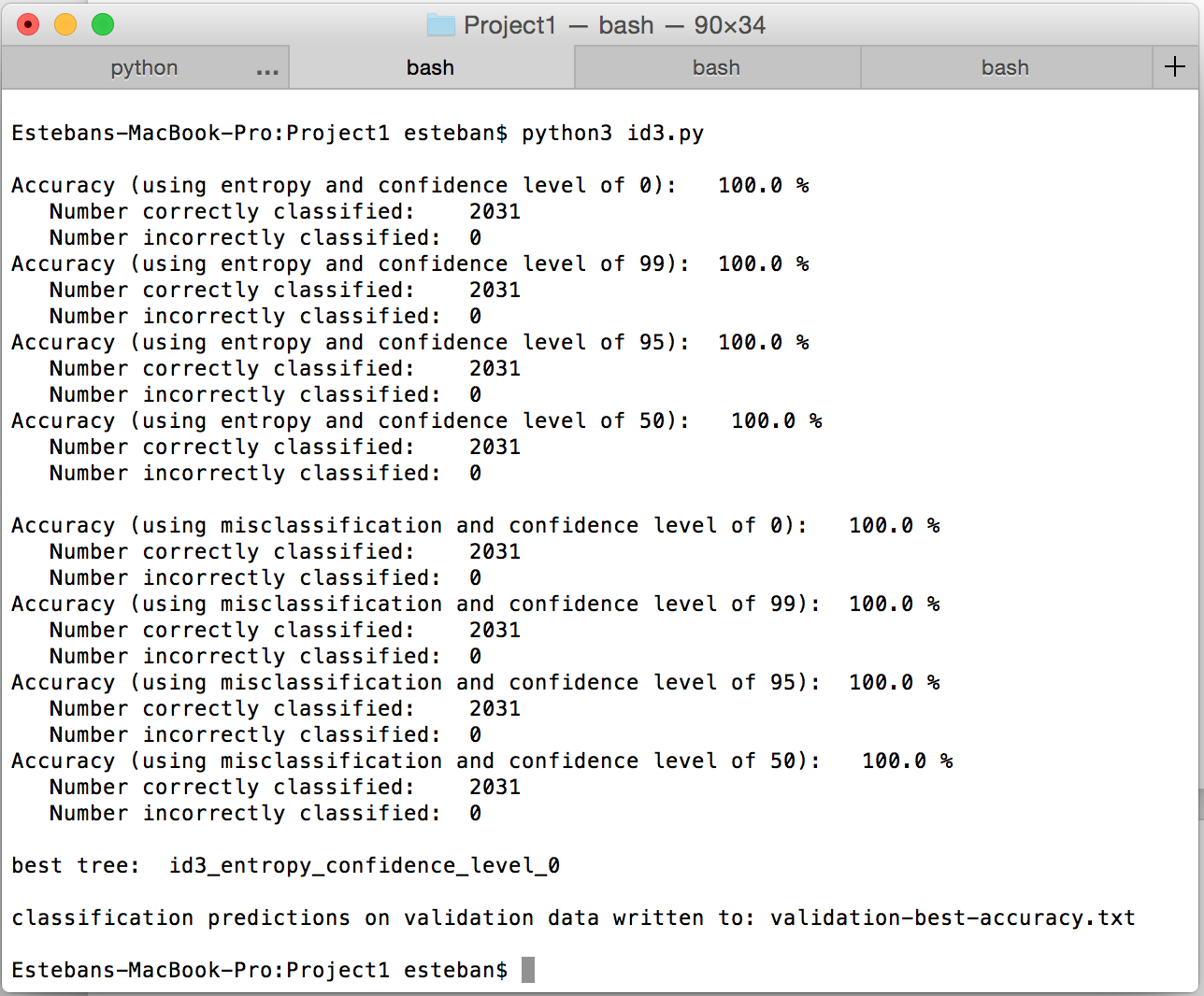


Figure 7. Classification results on the training.txt data

I used the entropy with CL 0% decision tree to classify the validation data. The 100% classification results can be seen below (Figure 8)

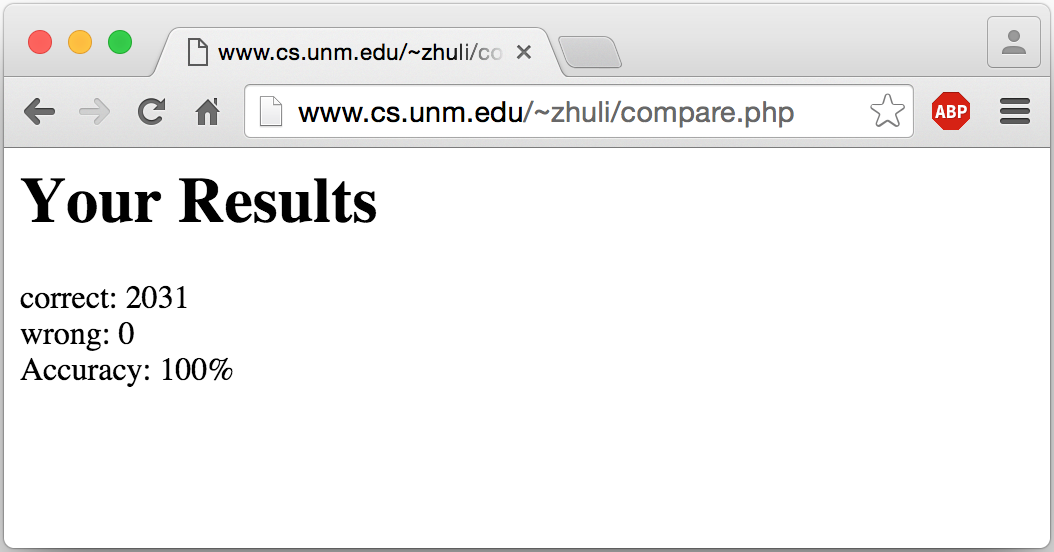


Figure 8. Accuracy for the validation.txt data

**Folklore Rules**

For proving or disproving the folklore rules I look at the data and/or my decision tree to come to a conclusion. I concluded that a folklore rule was true if the data or my decision tree backed-up the claim with a high probability (most of the data supported the claim). The included HTML export of my IPython Notebook shows how I queried the data.

Rule 1: Poisonous mushrooms are brightly colored:

**False**: I found there to be a fairly even split between edible and poisonous mushrooms for brightly colored (cap-color) mushrooms.

Rule 2: Poisonous mushrooms taste/smell bad:

**True**: The data provided strongly supports this (for smell bad). Almost all mushrooms with a bad odor were poisonous. My decision tree also supports this rule.

Rule 3: Poisonous mushrooms have a pointed or umbrella shaped cap:

**False**: The data provided strongly disproves this theory. Almost an even split between edible and poisonous mushroom having a pointed or umbrella shaped cap.

Rule 4: Edible mushroom have flat rounded shaped cap:

**False**: The data provided strongly disproves this theory. Almost an even split between edible and poisonous mushroom having a flat shaped cap.

Rule 5: Poisonous mushrooms have warts or scales on the cap:

**False**: The data provided strongly disproves this theory. Almost an even split between edible and poisonous mushroom having warts or scales on the cap.

Rule 6: Poisonous mushrooms have a bulbous cup or sac around the base:

**False**: The data provided strongly disproves this theory. Almost an even split between edible and poisonous mushroom having a bulbous cup or sac around the base.

Rule 7: Poisonous mushroom have a ring around the stem:

**False**: The data provided strongly disproves this theory. Almost an even split between edible and poisonous mushroom having a ring around the stem.

Rule 8: Poisonous mushrooms have gills that are thin and white:

**False**: The data provided strongly disproves this theory. Almost an even split between edible and poisonous mushroom having gills that are thin and white.

**Proposed Rules**

I could only find one rule not mentioned above to be true. The training data strongly (100%) supports “green spore print color mushroom are poisonous”.

**Rule 9**: Poisonous mushrooms have green spore print color: