Tristan Basil
CS460G-001 Machine Learning – Dr. Brent Harrison
2/7/2018

Problem Set 1 - Report

This assignment implements a decision tree classifier that is used to classify four synthetic datasets, and a decision tree classifier used to classify one real dataset. There are two included programs: Program1.py for the synthetic datasets, and Program1VideoGame.py for the real video game dataset. Both are implemented in Python 2. This report will provide a brief overview of each decision tree classifier with implementation decisions of each version, and visual classifiers for the synthetic decision tree.

Part 1: Classify Synthetic Data

The synthetic data decision tree classifier (Program1.py) implements the ID3 algorithm via information gain and entropy. It is implemented in Python 2 and uses the following packages:

1. **matplotlib.pyplot** to visualize the data points,
2. **numpy** for the arrange function, which effectively creates for loops with floats,
3. **math** to calculate logarithms in the information gain algorithm,
4. **sys** to get command line arguments (the filename), and
5. **copy** to deep copy sets/lists.

It has a maximum tree depth of 3 and is run with the following command:

```
$python Program1.py [filename]
```

Where [filename] should be replaced with any of the synthetic datasets (synthetic-1.csv, …, synthetic-4.csv). This will print the completed decision tree, the error of testing the training data, and the visualized data in a new window. There were a few implementation decisions worth mentioning:

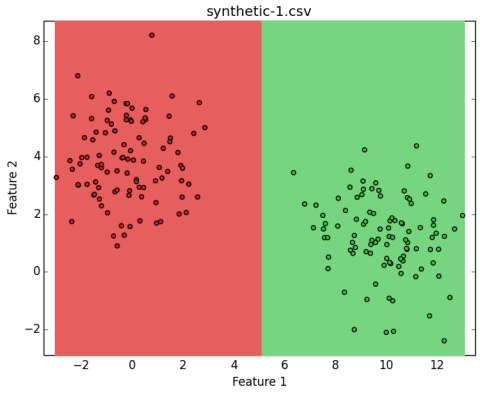
1. The continuous features were discretized into 10 bins each. Calling the ID3Tree constructor with a different value for numBins can change this amount, however.
2. The bins for each feature are equidistant, with the maximum and minimum values of the respective feature acting as the upper and lower limit. This was because it wasn't immediately clear what the domain of each feature value was.
3. There isn't logic to prevent a feature from falling into 2 bins. This is because with floats, it is unlikely that a feature would fit into 2 bins (due to inclusive upper and lower bounds) unless a feature value range of 0 was provided in the training data.
4. In the ID3 algorithm, if a node is reached and a simple majority of the examples' class labels is used to determine its label, the 0 path will be taken arbitrarily if there are an equal number of 0's and 1's.

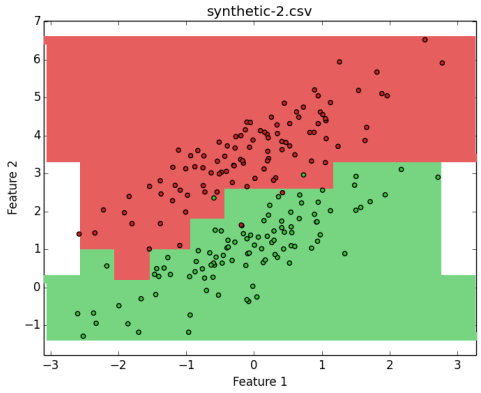
The final error results for synthetic data set with 10 bins and a maximum depth of 3 follow:

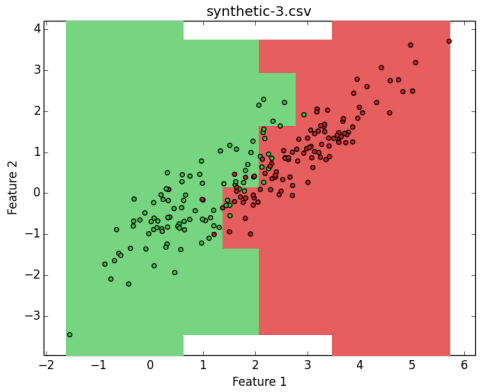
1. synthetic-1.csv error: **0.0**
2. synthetic-2.csv error: **0.02**
3. synthetic-3.csv error: **0.105**
4. synthetic-4.csv error: **0.03**

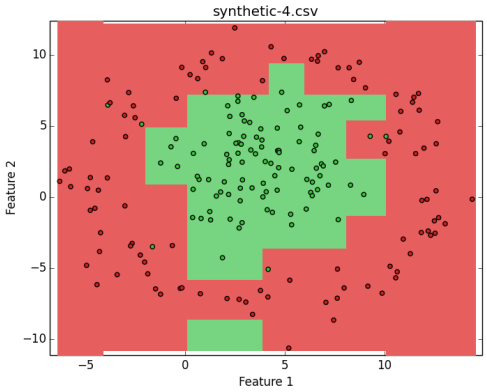
Part 2: Visualize your classifiers

The visualizations for each synthetic dataset follow, using a bin size of 10 to discretize each feature. Red and green scatter points indicate a real example with a class value of 0 or 1, respectively. The red and green backgrounds indicate a decision tree class value prediction of 0 or 1, respectively.









Part 3: Classify Real Data on Video Game Sales

The synthetic data decision tree classifier (Program1VideoGame.py) implements the ID3 algorithm via the information gain and entropy. It is implemented in Python 2 and uses the following packages:

1. **math** to calculate logarithms in information gain,
2. **sys** to get command line arguments (the filename),
3. **copy** to deep copy sets/lists, and

4. **collections** (Counter) to count the most common value in a dictionary.

It has a maximum depth of 3 (although this can be changed) and is run with the following command:

```
$python Program1VideoGame.py Video_Games_Sales.csv
```

Note that executing this program can take around 50 seconds. This will print the completed decision tree and the error of testing the training data. There were a few implementation decisions worth mentioning:

1. For missing string data in Video_Games_Sales.csv, the value is treated as the string 'N/A' and is classified the same way. This made classification more consistent.
2. For string feature data including commas, the commas are replaced with the character '/' and classified the same way. This made reading and parsing the file much simpler.
3. The continuous features were discretized into 10 bins each. Calling the ID3Tree constructor with a different value for numBins can change this amount, however.
4. The bins for each feature are equidistant, with the maximum and minimum values of the respective feature acting as the upper and lower limit. This was because it wasn't immediately clear what the domain of each feature value was.
5. The class labels were discretized into 10 bins as well, with a minimum of 0 and a maximum of 100. This was because it was clear that the domain of the scores was 0-100.
6. There isn't logic to prevent a feature value from falling into 2 bins. This is because with floats, it is unlikely that a feature would fit into 2 bins (due to inclusive upper and lower bounds) unless a feature value range of 0 was provided in the training data.
7. Logic exists to prevent a class value from falling into multiple bins. In the event that one does, it will arbitrarily fall into the lower of the two bins. This is because integers could easily fit into two bins with inclusive bounds.

The final error results for the real data set with 10 bins and maximum depth 3 is **0.0603713266937**.

Resources Used:

- https://stackoverflow.com/questions/32796531/how-to-get-the-most-common-element-from-a-list-in-python (how to use collections library to get the most common element from a dictionary object)
- https://matplotlib.org/tutorials/introductory/sample_plots.html#sphx-glr-tutorials-introductory-sample-plots-py (how to use matplotlib)