Homework 1 Write-Up

By: Alexander Polus
For: Dr. Harrison

My programs can be found within the data folder. This was best for reading in the csv data.

For this assignment, we were asked to generate a machine learning model using a decision-tree fitting method. This model was then applied to four sets of synthetic data and then slightly modified to be applied to one set of real world data, pertaining to video game sales. Both implementations used the ID3 algorithm, which built a tree recursively. Then, I developed a method (walk_tree()) that would traverse the tree given two sets of input data, make a prediction.

All code was written in Python3

Synthetic Data:

```
Usage: Python3 SyntheticModel.py <filename>
```

To make my approach generic for all 4 datasets, I discretized my data by formatting each column of feature data into 5 bins, leaving the class label unaltered. Then, each value in columns A and B would be an integer value 0-4, and the class label value would be either zero or 1. The bins were created by taking the maximum point minus the minimum point for each column, dividing by 5, and using that interval to build 5 'evenly-sized' bins.

After discretizing the data, I pass that table (2D array) into my ID3 algorithm, which builds a "tree" which is an array of nodes. Each node keeps track of it's children, it's prediction value of class label (if it's capable of giving one, it not it's -1), it's own index in the tree, its parent's index in the tree, the majority class of the node, and what feature it used to generate its sub-table. Then, by traversing the nodes using the walk_tree method, a decision can be reached.

Finally, the program will output its accuracy, indicating what percentage of testing labels were equal to the training labels for identical input.

For each model, here are the percentages for model accuracy and training set error (100 – model accuracy %):

Synthetic-1:
Model Accuracy: 100.0 %
Training Set Error: 0.0 %

Synthetic-2:
Model Accuracy: 94.5 %
Training Set Error: 5.5 %

Synthetic-3:
Model Accuracy: 87.0 %
Training Set Error: 13.0 %

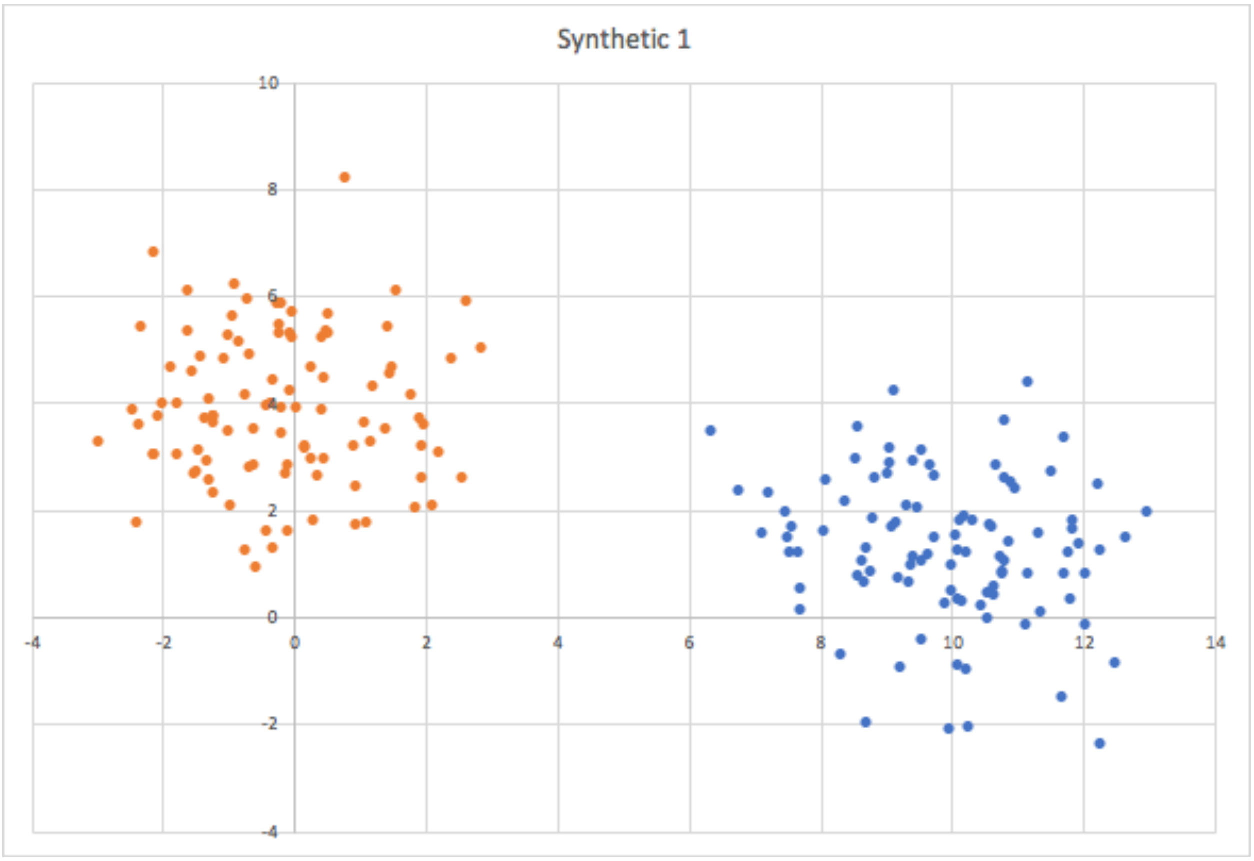
Synthetic-4:

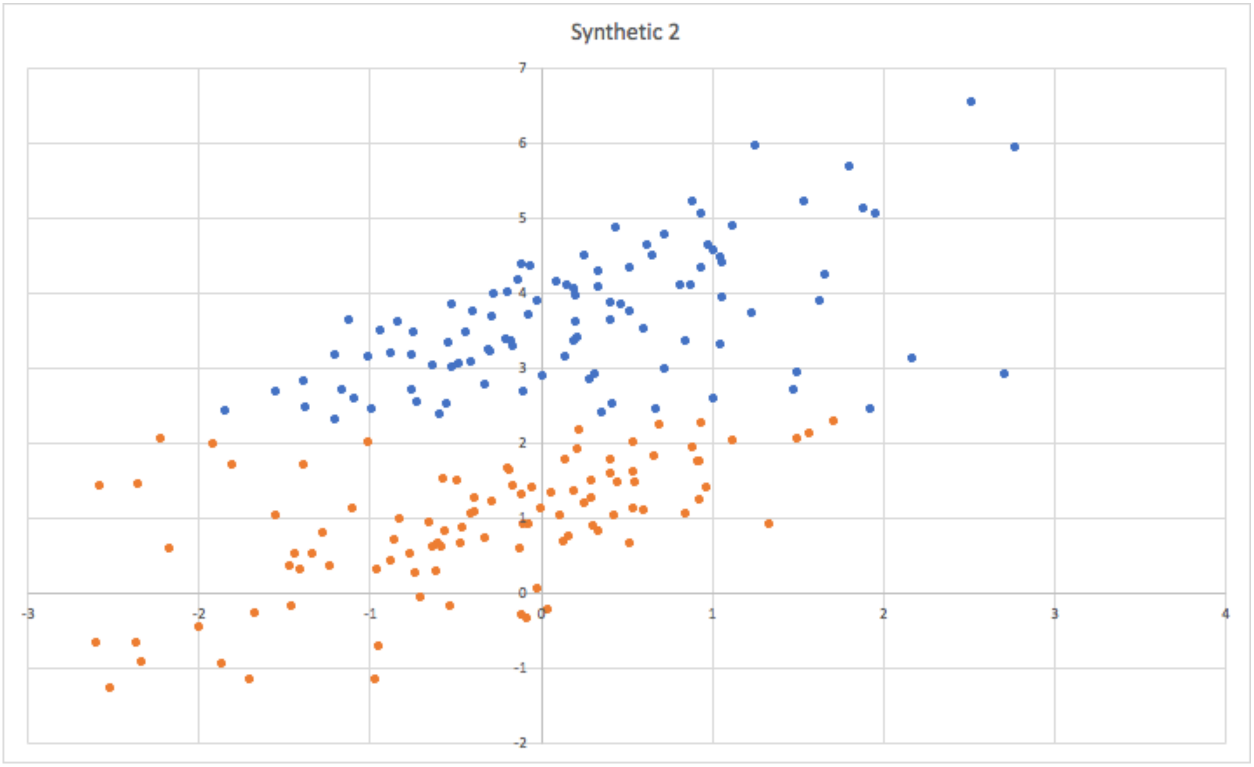
Model Accuracy: 96.0 %
Training Set Error: 4.0 %

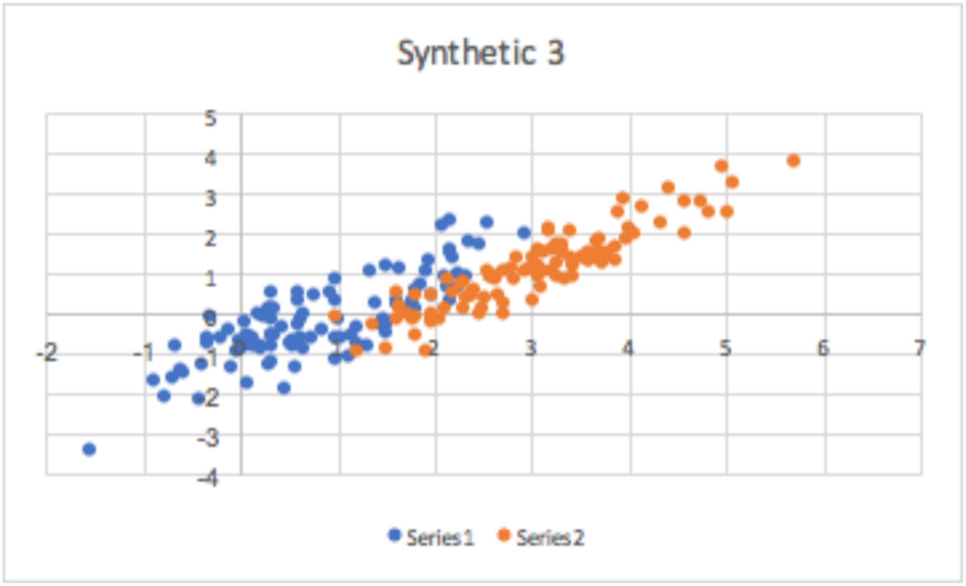
Therefore, based on these numbers, the model was effective in order from most to least for Synthetic-1, Synthetic-4, Synthetic-3, and Synthetic-2.

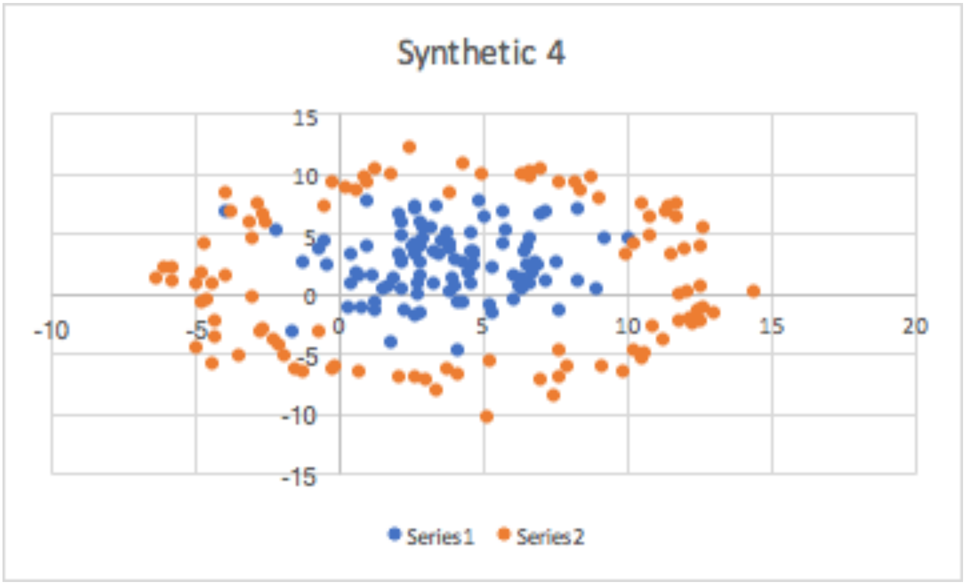
The data is visualized below for all synthetic datasets. Orange represents a zero, blue represents a one. Column A is on the X axis, and column B is on the Y axis.

I was able to generate these plots, but I could not generate a color-coded decision boundary. Since I have 5 bins for each value, I would have a 5x5 matrix to color in, but I could not figure out a way in my program to keep track of which square should be what color, and could not figure out how to color a specific area either. Matplotlib didn't work well with the way the data was presented, and I could not use my walk_tree method to properly color in the squares.


Synthetic 1
10
8
6
4
2
0
-2
-4
-4
-2
0
2
4
6
8
10
12
14


Synthetic 2
7
6
5
4
3
2
1
0
-1
-2
-3
-2
-1
0
1
2
3
4


Synthetic 3
5
4
3
2
1
0
-1
-2
-3
-4
-2
-1
0
1
2
3
4
5
6
7
● Series1 ● Series2



<u>Video Game Sales Data:</u>

For this portion, I created two models: one with a max depth of three as required by the project, and one without a max depth in order to maximize its effectiveness. The program,

Limit_3_VideoGameModel.py is the one I would like to submit for grading.

Usage: python3 Limit_3_VideoGameModel.py

It will take around 30 seconds to run, and has a training set error of 6.9707401032702165 %

To implement this, I discretized each column of data and turned every attribute into a number. For each value, I recorded a list of possibilities, and set each value equal to its respective index in the list of possibilities. This might seem complicated due to the fact that each column will likely have a different number (and possibly very high number) of bins, but it turned out to be very effective, especially in the other model that is not limited by depth three.

To discretize the label, I turned every value into either 15, 25, 35, 45, … , 95, based on if the value was between 0-19, 20-29, 30-39, etc…

I chose 5 as the least significant digit because I didn't want my program to predict the best rating or worst rating ever (99 or 0).

Otherwise, this model works identical to the synthetic data model, the only difference being that it outputs one of those numbers [15,25,…] instead of [0,1].