WRITEUP

WU1

We trained OAA and AVA decision trees of depth = 3 and k = 5 on the small wine dataset for part (a) and then with depth = 3 and k = 20 on the full wine dataset for parts (b) and (c).

OAA

a. Branching Y on ‘citrus’ and ‘grapefruit’ seem to most likely indicate Sauvignon-Blanc, while branching N on ‘citrus’ and ‘lime’ seem to most likely indicate not Sauvignon-Blanc. Branching Y on ‘cherry’ and N on ‘cassis’ and ‘verdot’ seem to most likely indicate Pinot-Noir, while branching N on ‘cherry’, ‘raspberry’, and ‘strawberry’ seem to most likely indicate not Pinot-Noir.
b. Training a depth-3 OAA decision tree on the full wine dataset took 0.3630 seconds and yielded 37.0% accuracy. Branching N on ‘peaches’ most likely indicates Viognier.
c. The accuracy with useZeroOne=True is 24.8%, or about 12% worse than confidence prediction.

AVA

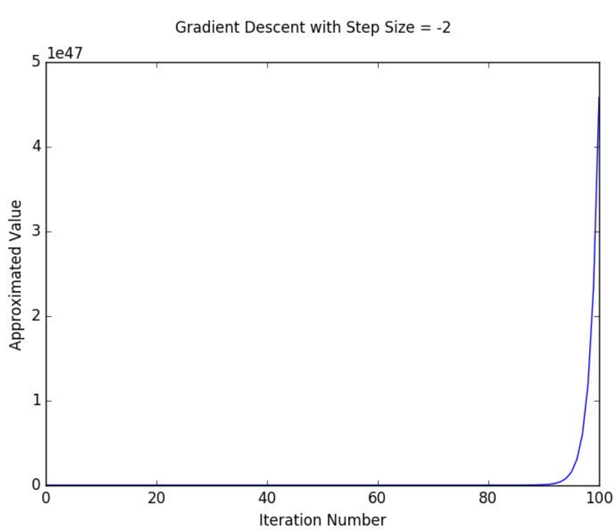
a. Branching Y on ‘citrus’ and ‘crisp’ seem to most likely indicate Sauvignon-Blanc, while branching N on ‘citrus’, ‘crisp’, and ‘lime’ seem to most likely indicate not Sauvignon-Blanc. Branching N on ‘crisp’, ‘lemon’, and ‘lime’ seem to most likely indicate Pinot-Noir, while branching N on ‘crisp’ and ‘pear’ seem to most likely indicate not Pinot-Noir.
b. Training depth-3 AVA decision trees on the full wine dataset took .3590 seconds and yielded 26.9% accuracy. Branching N on ‘peaches’, ‘pear’, and ‘milk’ most likely indicate Viognier.
c. The accuracy with useZeroOne=True is 26.4%, or about the same as confidence prediction.

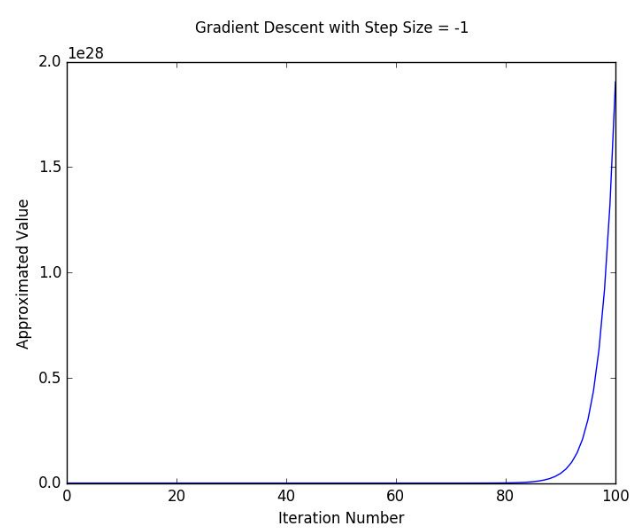
WU2

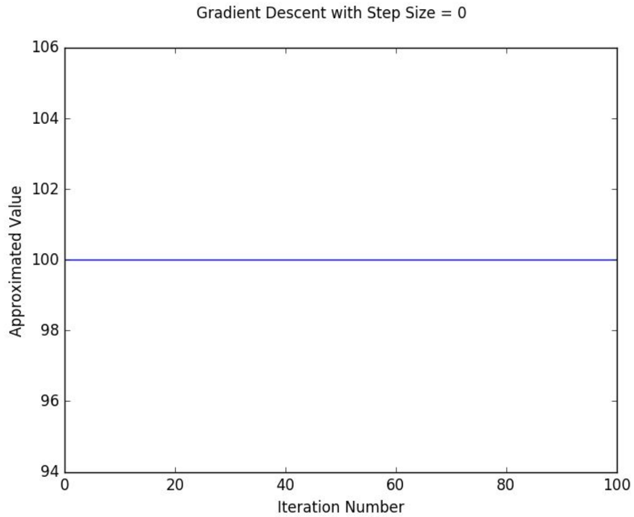
The test accuracy with a balanced tree on the full wine dataset using a DecisionTreeClassifier with max depth 3 is 30.9%.

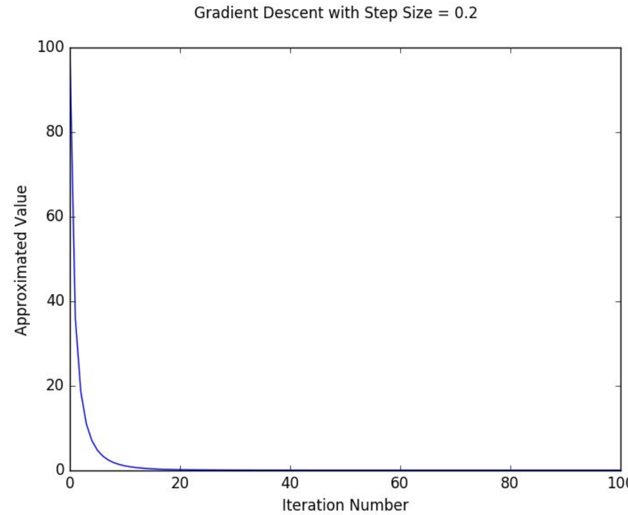
WU3

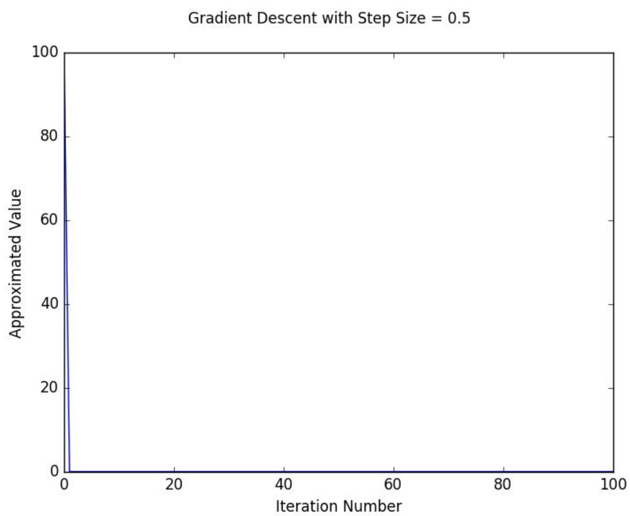
For negative values of step size, the gradient descent algorithm diverges on the function f(x) = x^2. For a step size of 0, the algorithm predicts the wrong value but remains constant at that value. For step sizes 0.2, 0.5, and 1 the algorithm successfully finds the global minimum, although 0.5 manages to find it in only one step. Interestingly, for step sizes above 1, the algorithm begins displaying a weird behavior in which it seemingly diverges but then returns to converge again. We are not sure what this means.

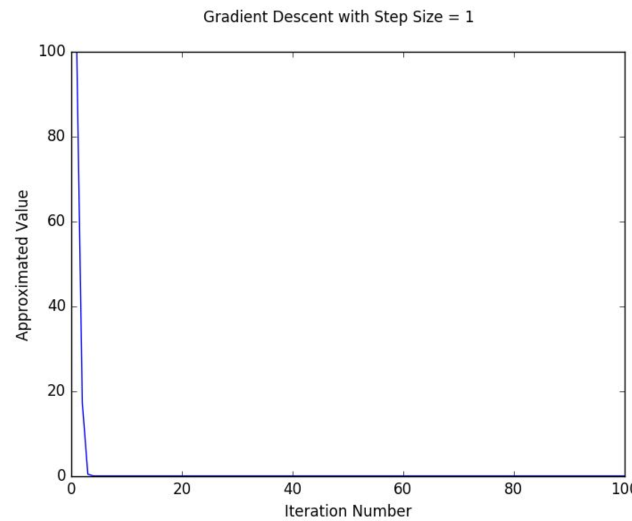


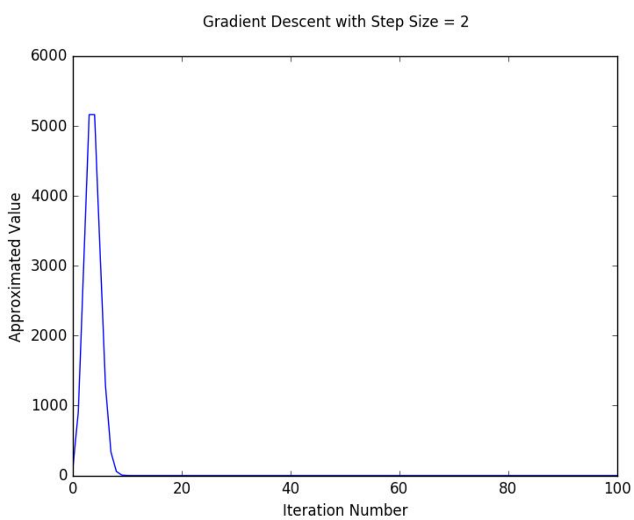


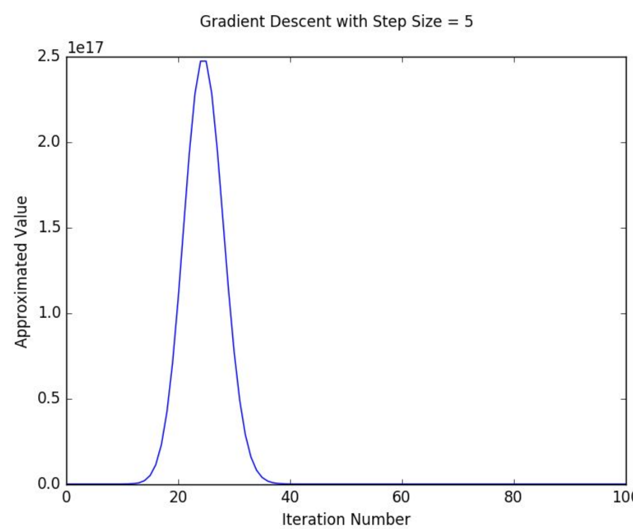






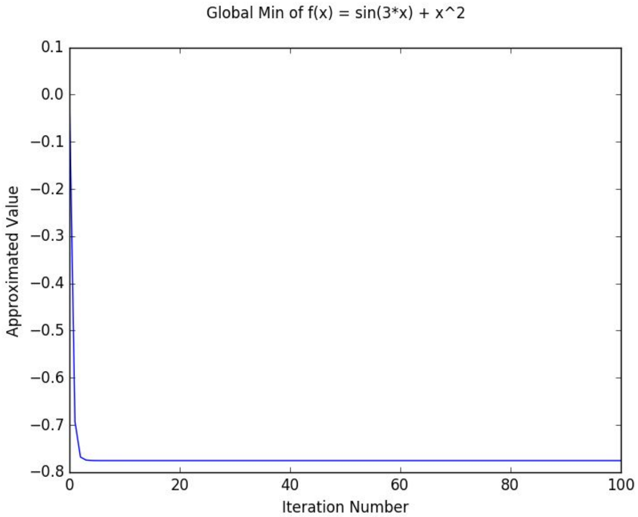


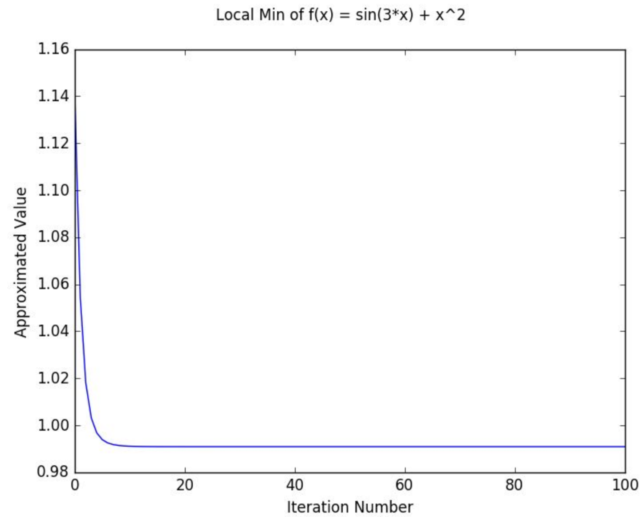


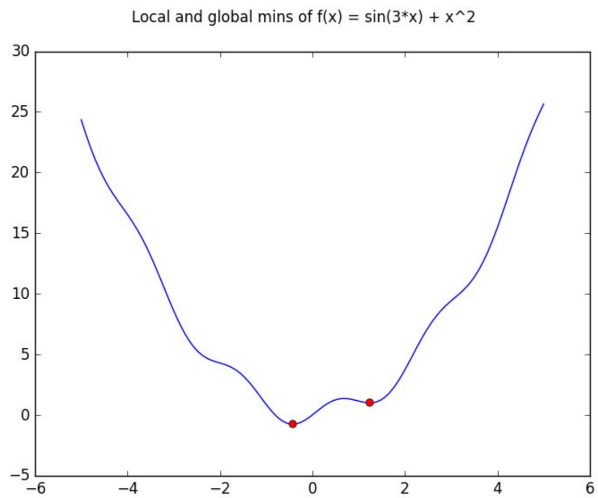


WU4

For a non-convex univariate function such as f(x) = sin(3x) + x^2, the gradient descent algorithm with 100 iterations and step size = 0.1 finds the global minimum at (-0.4273, -0.7760) if the starting value is 0. The algorithm gets stuck on a local minimum at (1.2446, 0.9908) if the starting value is 1.







WU5

Using SquaredLoss, the training accuracy was 24.3% and the test accuracy was 31.4%. Using LogisticLoss, the training accuracy was 99.6% and the test accuracy was 97.4%. Using HingeLoss, the training accuracy was 75.3% and the test accuracy was 68.6%. All three models used lambda = 1, iteration = 100, and step size = 0.5. It seems that LogisticLoss had the best results, so using its weight vector, we found the following top 5 negative and positive weights along with their corresponding words:

Top 5 negative weights + words
[-1.1695, -0.7653, -0.6836, -0.6296, -0.5322] ['tannins', 'black', 'dark', 'cherry', 'blackberry']

The words associated with these top negative values are the ones which have the greatest impact on predicting the negative class, which in this case is red wine.

Top 5 positive weights + words
[0.6064, 0.6892, 0.7109, 0.7701, 0.8833] ['tropical', 'acidity', 'lime', 'crisp', 'citrus']'

In contrast to the top negative words, the top positive words are those which are most likely to predict white wine, the positive class.