Homework 9

Name: Shraddha Manchekar

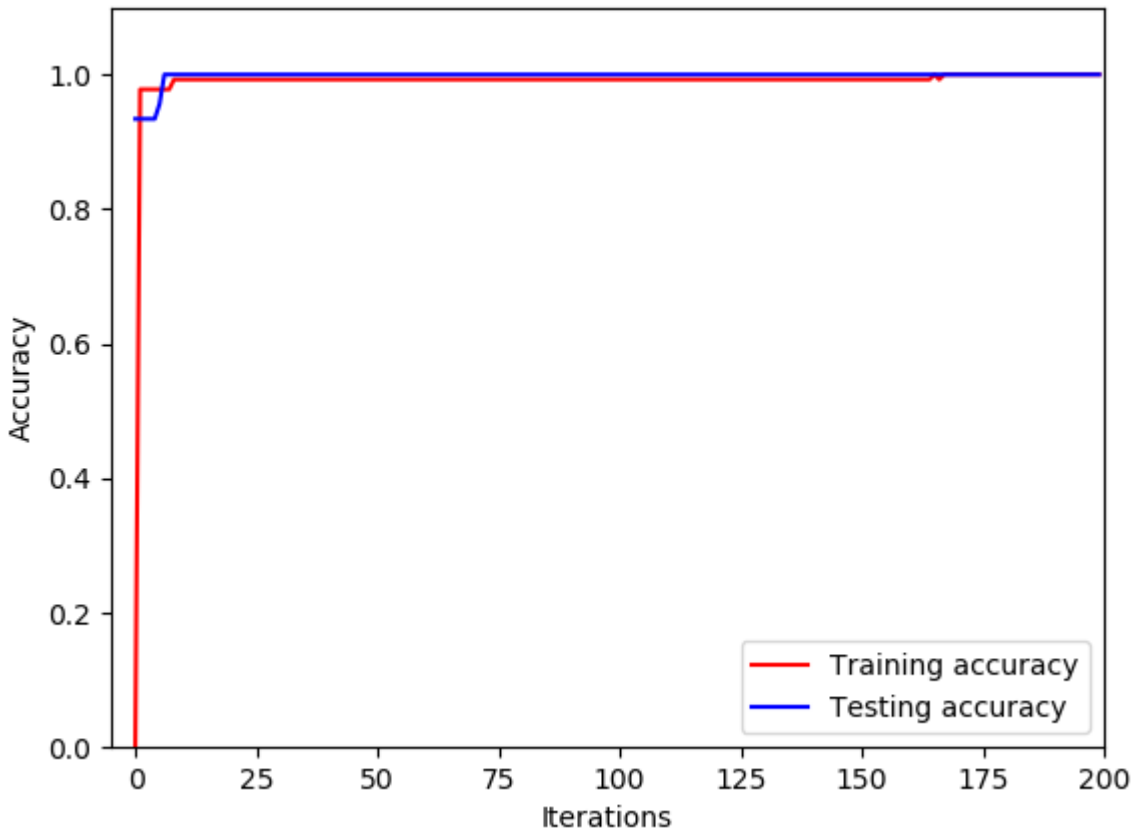
UID: 004945217

SVM:

```
C:\Users\shraddha_m26\Desktop\Stats Programming\Assignments\9>python
Stats202A_HW9_P2.py

After iteration 0, Train Accuracy = 0.000000,Test Accuracy = 0.934066
After iteration 10, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 20, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 30, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 40, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 50, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 60, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 70, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 80, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 90, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 100, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 110, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 120, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 130, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 140, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 150, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 160, Train Accuracy = 0.992647,Test Accuracy = 1.000000
After iteration 170, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 180, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 190, Train Accuracy = 1.000000,Test Accuracy = 1.000000
Training Accuracy:
[ 0.          0.97794118  0.97794118  0.97794118  0.97794118  0.97794118
  0.97794118  0.97794118  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
```

```
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706  0.99264706
  0.99264706  0.99264706  0.99264706  1.          0.99264706  1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.        ]
Testing Accuracy:
[ 0.93406593  0.93406593  0.93406593  0.93406593  0.93406593  0.95604396
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.        ]
```

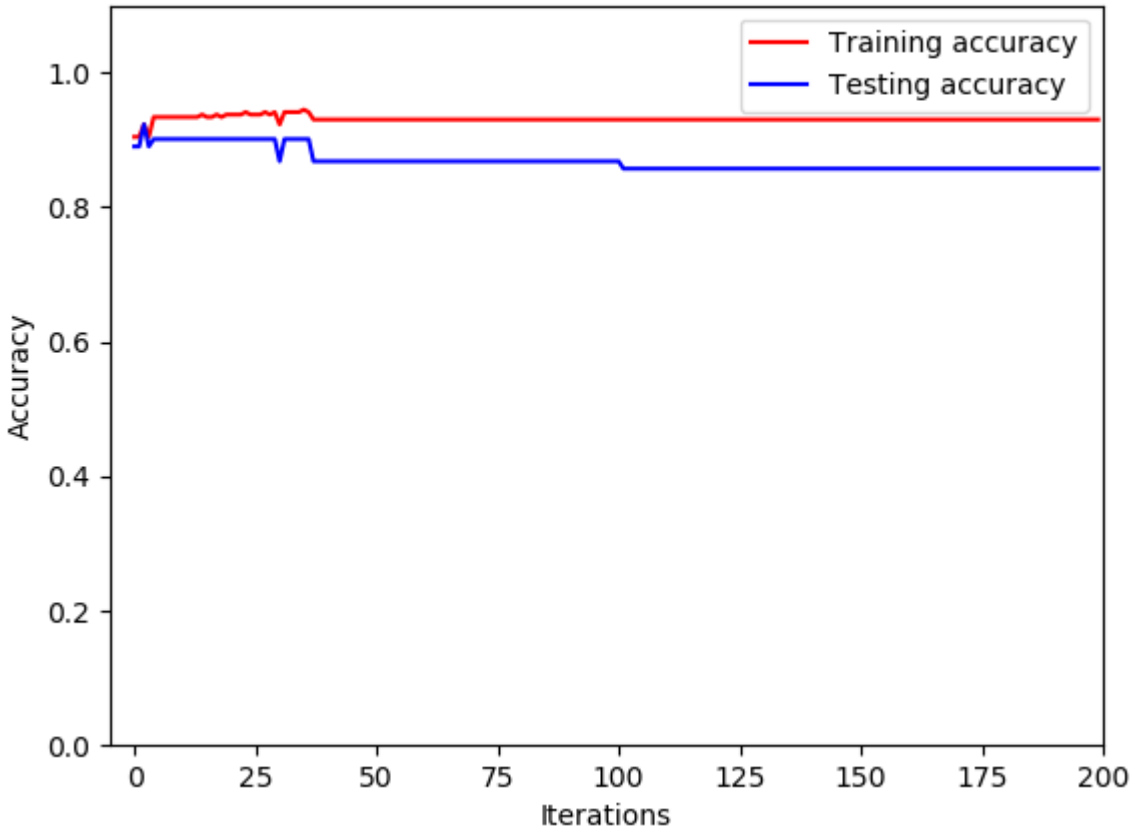


Observation: SVM (support vector machine) constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outlier detection. SVM uses the closest points in different classes as “support vectors” to estimate a convex, yet optimal hyperplane separating the classes. SVM has a regularization parameter that avoids over fitting. As seen from the graph, SVM avoids over fitting and provides excellent results for the test accuracy. SVMs are resilient to noise and take the least time to train.

Adaboost:

```
C:\Users\shraddha_m26\Desktop\Stats Programming\Assignments\9\submitted>python
Stats202A_HW9_P2.py
Iteration 0: , Train Accuracy = 0.904412,Test Accuracy = 0.890110
Iteration 10: , Train Accuracy = 0.933824,Test Accuracy = 0.901099
Iteration 20: , Train Accuracy = 0.937500,Test Accuracy = 0.901099
Iteration 30: , Train Accuracy = 0.922794,Test Accuracy = 0.868132
Iteration 40: , Train Accuracy = 0.930147,Test Accuracy = 0.868132
Iteration 50: , Train Accuracy = 0.930147,Test Accuracy = 0.868132
Iteration 60: , Train Accuracy = 0.930147,Test Accuracy = 0.868132
Iteration 70: , Train Accuracy = 0.930147,Test Accuracy = 0.868132
Iteration 80: , Train Accuracy = 0.930147,Test Accuracy = 0.868132
Iteration 90: , Train Accuracy = 0.930147,Test Accuracy = 0.868132
Iteration 100: , Train Accuracy = 0.930147,Test Accuracy = 0.868132
Iteration 110: , Train Accuracy = 0.930147,Test Accuracy = 0.857143
Iteration 120: , Train Accuracy = 0.930147,Test Accuracy = 0.857143
Iteration 130: , Train Accuracy = 0.930147,Test Accuracy = 0.857143
Iteration 140: , Train Accuracy = 0.930147,Test Accuracy = 0.857143
Iteration 150: , Train Accuracy = 0.930147,Test Accuracy = 0.857143
Iteration 160: , Train Accuracy = 0.930147,Test Accuracy = 0.857143
Iteration 170: , Train Accuracy = 0.930147,Test Accuracy = 0.857143
Iteration 180: , Train Accuracy = 0.930147,Test Accuracy = 0.857143
Iteration 190: , Train Accuracy = 0.930147,Test Accuracy = 0.857143
Training Accuracy:
[ 0.90441176  0.90441176  0.92279412  0.90441176  0.93382353  0.93382353
  0.93382353  0.93382353  0.93382353  0.93382353  0.93382353  0.93382353
  0.93382353  0.93382353  0.9375      0.93382353  0.93382353  0.9375
  0.93382353  0.9375      0.9375      0.9375      0.9375      0.94117647
  0.9375      0.9375      0.9375      0.94117647  0.9375      0.94117647
  0.92279412  0.94117647  0.94117647  0.94117647  0.94117647  0.94485294
  0.94117647  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
```

```
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706  0.93014706
  0.93014706  0.93014706]
Testing Accuracy:
[ 0.89010989  0.89010989  0.92307692  0.89010989  0.9010989   0.9010989
  0.9010989   0.9010989   0.9010989   0.9010989   0.9010989   0.9010989
  0.9010989   0.9010989   0.9010989   0.9010989   0.9010989   0.9010989
  0.9010989   0.9010989   0.9010989   0.9010989   0.9010989   0.9010989
  0.9010989   0.9010989   0.9010989   0.9010989   0.9010989   0.9010989
  0.86813187  0.9010989   0.9010989   0.9010989   0.9010989   0.9010989
  0.9010989   0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187
  0.86813187  0.86813187  0.86813187  0.86813187  0.86813187  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286  0.85714286
  0.85714286  0.85714286]
```



Observation: Adaboost combines the output of the other learning algorithms ('weak learners') into a weighted sum that represents the final output of the boosted classifier. Adaboost can handle sparse dataset and hence, can work with weak classifiers. But, it shows some variations after it reaches peak classification correctness. AdaBoost can be sensitive to noisy data and outliers. As seen from the graph, Adaboost reaches an effective solution quickly, but in subsequent iterations, it becomes sensitive to noise. Adaboost is the least accurate for the given dataset. It should be used when there is a class imbalance in the dataset.