python Adaboost.py --data linear --feature linear --max_iterations 100
Accuracy: training set: 1.0
Accuracy: validation set: 0.995
Accuracy: test set: 0.995

python Adaboost.py --data quadratic --feature linear --max_iterations 100
Accuracy: training set: 0.985
Accuracy: validation set: 0.95
Accuracy: test set: 0.935

python Adaboost.py --data quadratic --feature quadratic --max_iterations 100
Accuracy: training set: 1.0
Accuracy: validation set: 0.98
Accuracy: test set: 0.94

Python Adaboost.py --data noisy_linear --feature linear --max_iterations 100
Accuracy: training set: 0.95
Accuracy: validation set: 0.7
Accuracy: test set: 0.65

Observation:
For linear data using linear data feature, Adaboost works very well, since it can easily learn the boundary which can perfectly separate two different kinds of data. When it comes to the quadratic data set using linear data feature, Adaboost still works well but not that perfect. I think the reason that adaboost used many linear boundaries to separate data, but there are still some of them cannot be labeled correctly. For quadratic data using quadratic data feature, Adaboost works better than using linear data feature for quadratic data set. Lastly, for data with some noise, Adaboost works well enough for the trainning set but not well for the test data set. I think this is because some random noise are hard to predict even id we train adaboost very well.

I have tried different iteration number smaller than 100 for those four different data set. What I have observed is that the accuracy is lower than we iterating it 100 times especially for the data quadratic data set using linear data feature. Therefore, I think how many times we run the adaboost indeed affect the final accuracy no matter for the training set, validation set or the test set. Hence, it is important for us to pick a reasonable number for the iteration.