Jodie Zeng

jzeng9@u.rochester.edu

CSC 242 Project 4

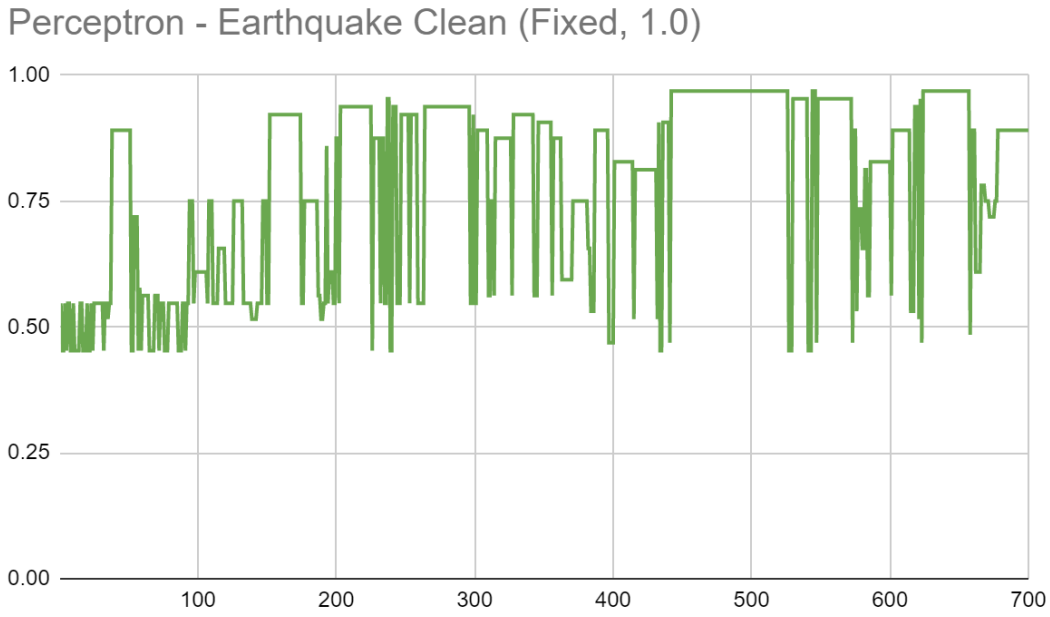
To build my project, I opened a command terminal in the folder containing all my java files and data files.

```
javac DecayingLearningRateSchedule.java Example.java
LearningRateSchedule.java LinearClassifier.java
LogisticClassifier.java MachineLearning.java
PerceptronClassifier.java VectorOps.java
```

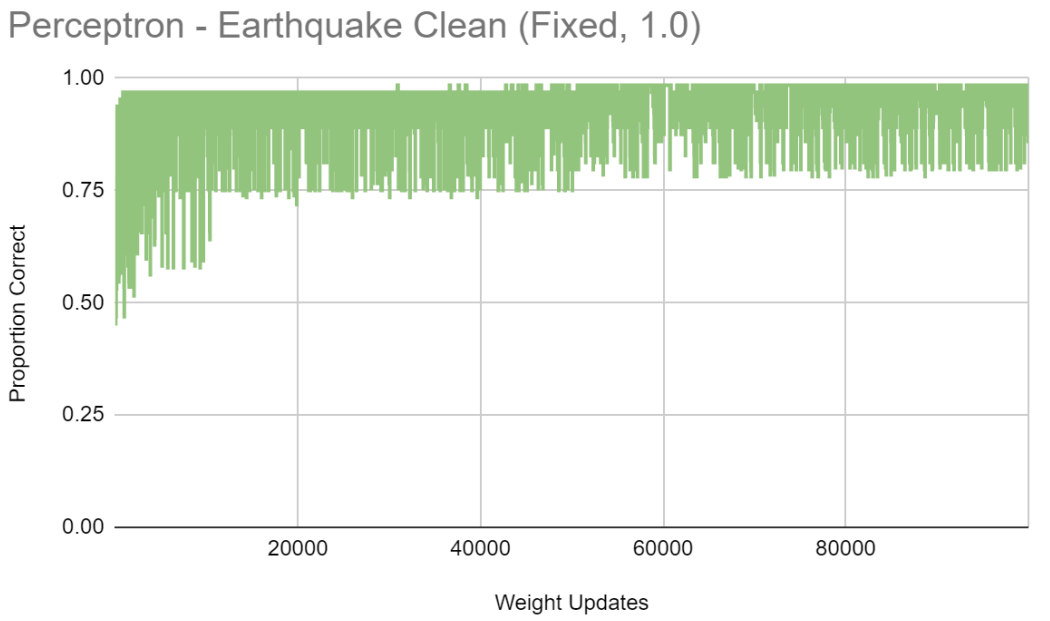
To run my project, enter the main class file, MachineLearning, then the file name of the data file, the type of classifier (“perceptron”/“logistic”), the alpha type (“decay”/ or a constant double), the starting weights (any double), and the number of steps (int). For example:

```
java MachineLearning earthquake-clean.data.txt perceptron decay 1.0
5000
```

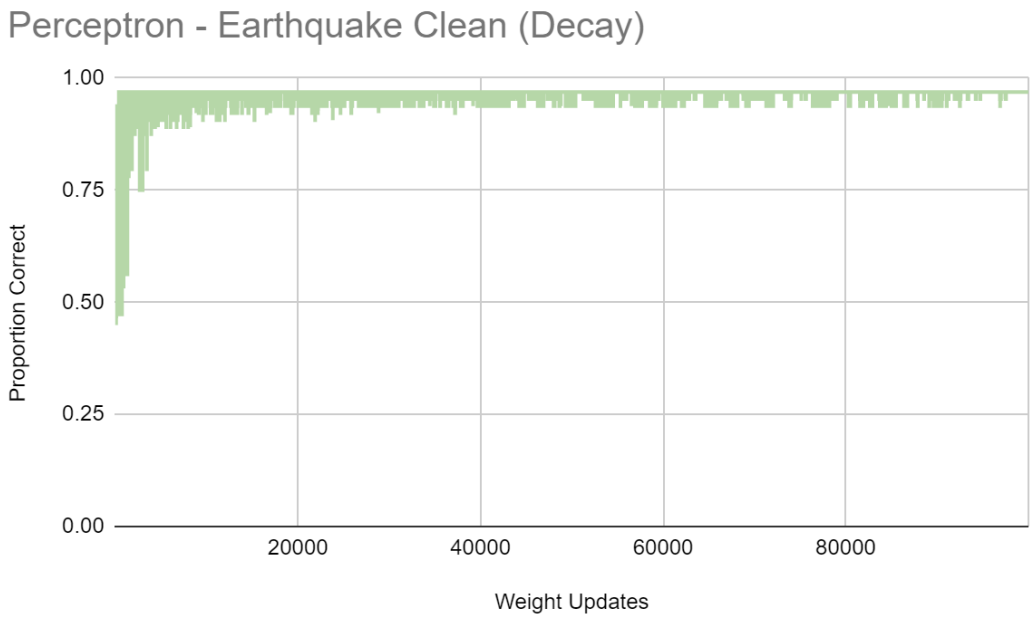
Perceptron Classifier:



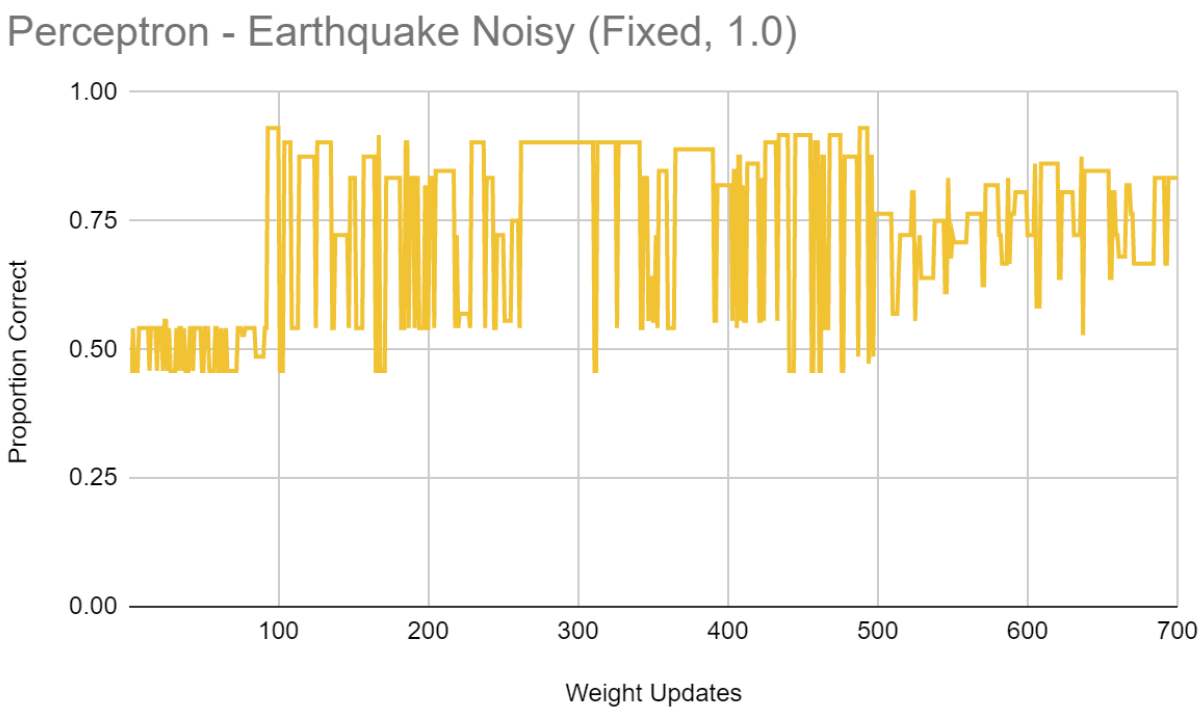

```
java MachineLearning earthquake-clean.data.txt perceptron 1.0 1.0 700
```



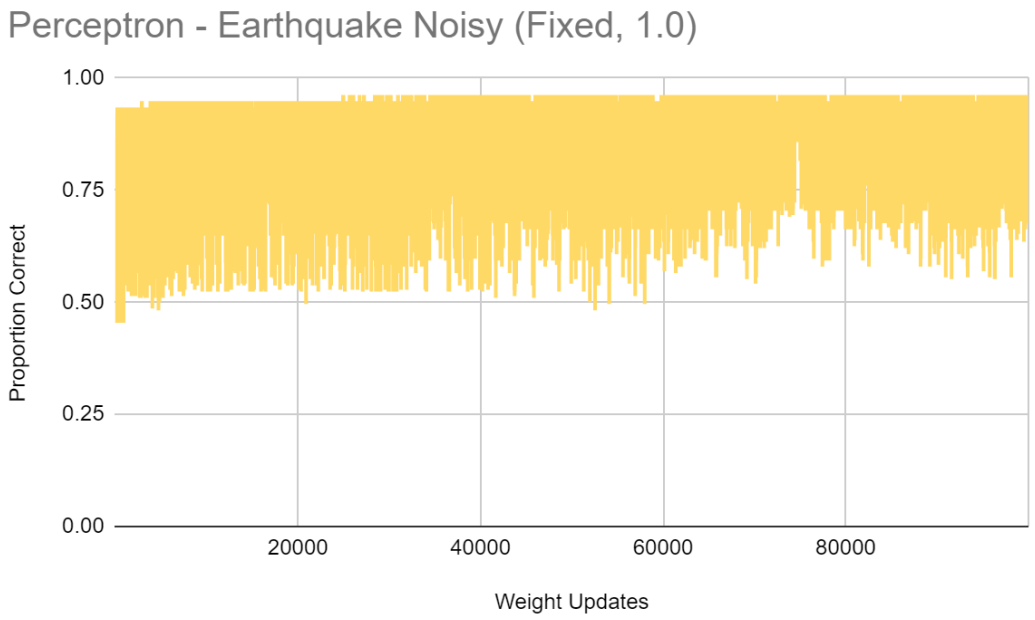

```
java MachineLearning earthquake-clean.data.txt perceptron 1.0 1.0
100000
```



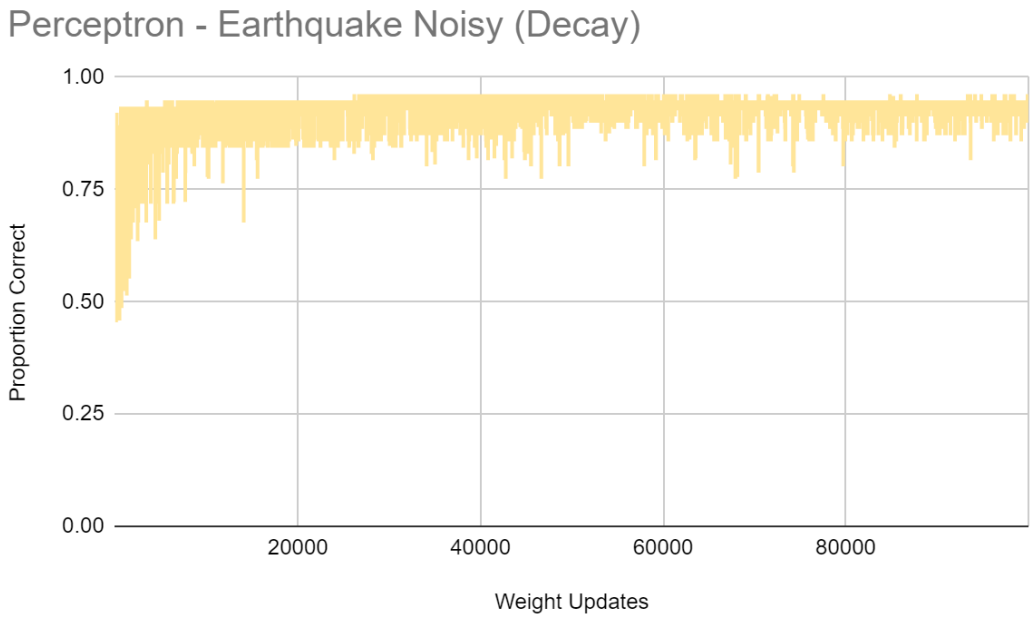

```
java MachineLearning earthquake-clean.data.txt perceptron decay 1.0
100000
```



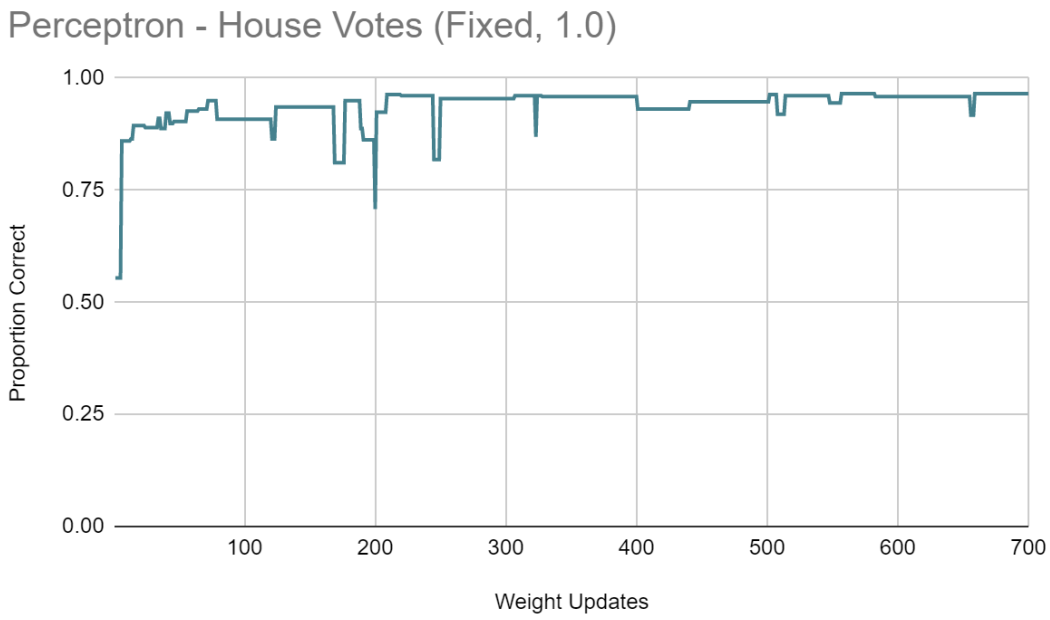

```
java MachineLearning earthquake-noisy.data.txt perceptron 1.0 1.0 700
```



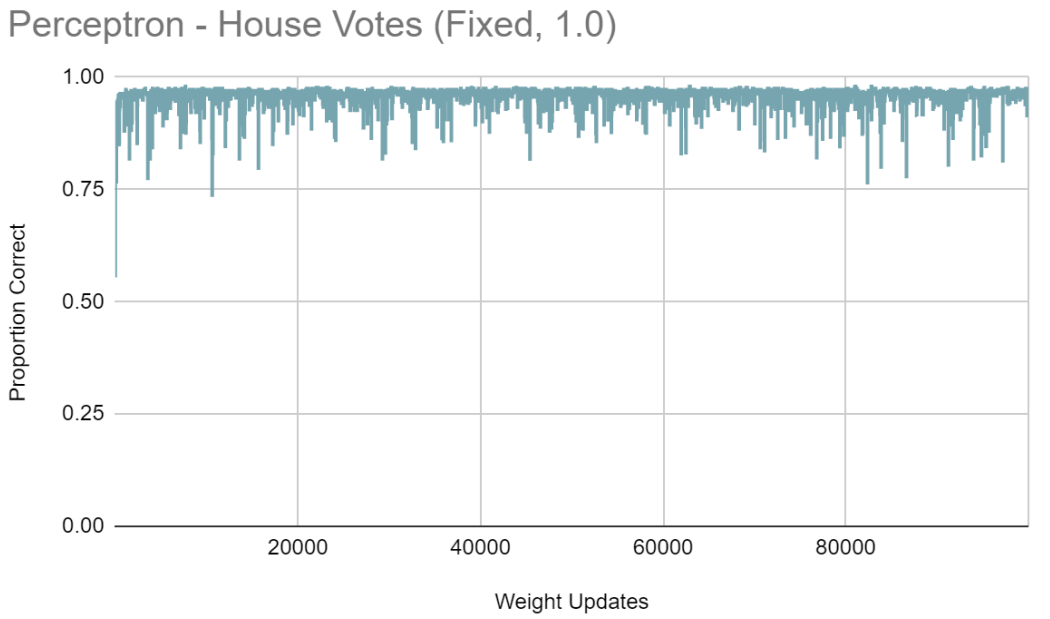

```
java MachineLearning earthquake-noisy.data.txt perceptron 1.0 1.0
100000
```



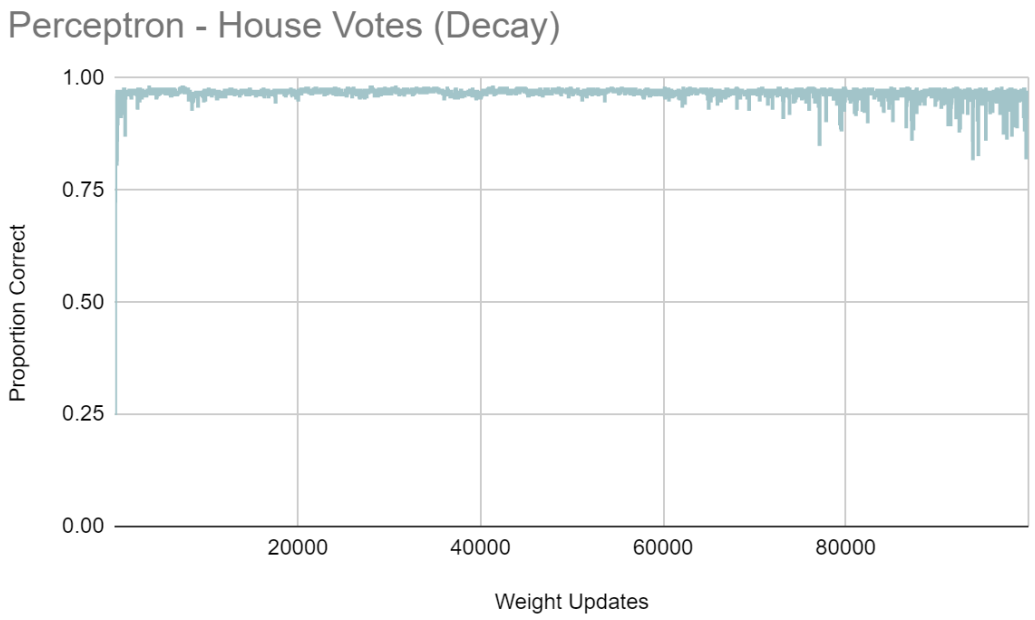

```
java MachineLearning earthquake-noisy.data.txt perceptron decay 1.0
100000
```




```
java MachineLearning house-votes-84.data.num.txt perceptron 1.0 1.0
700
```

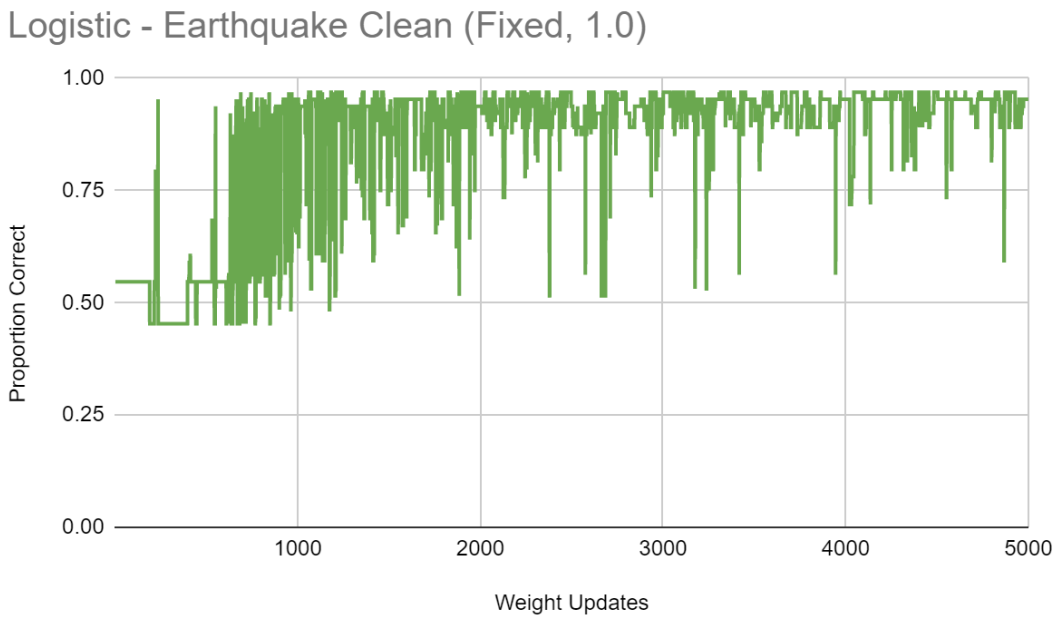



```
java MachineLearning house-votes-84.data.num.txt perceptron 1.0 1.0
100000
```

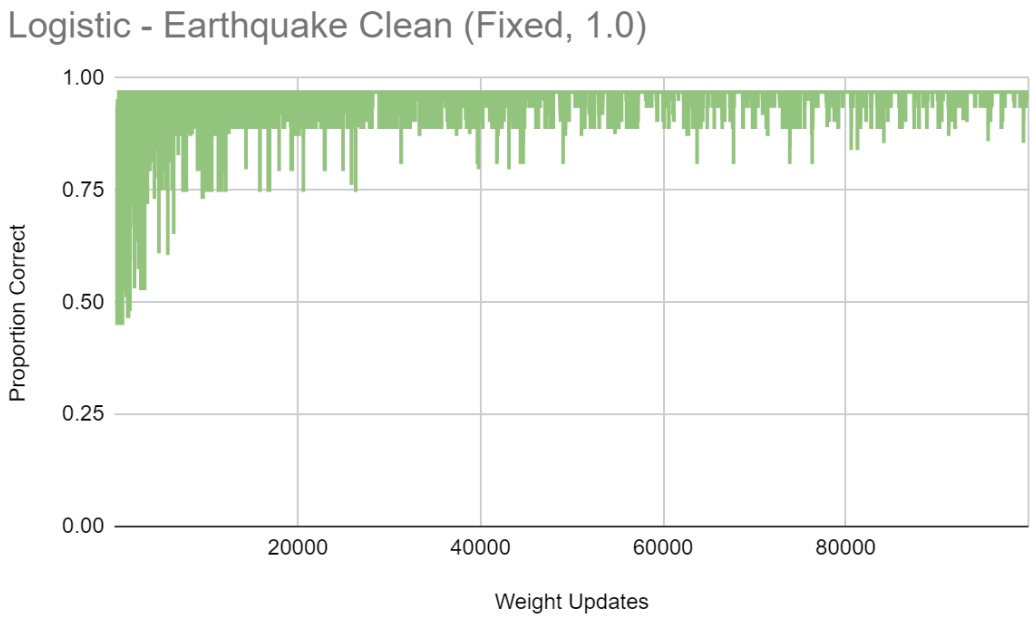



```
java MachineLearning house-votes-84.data.num.txt perceptron decay 1.0
100000
```

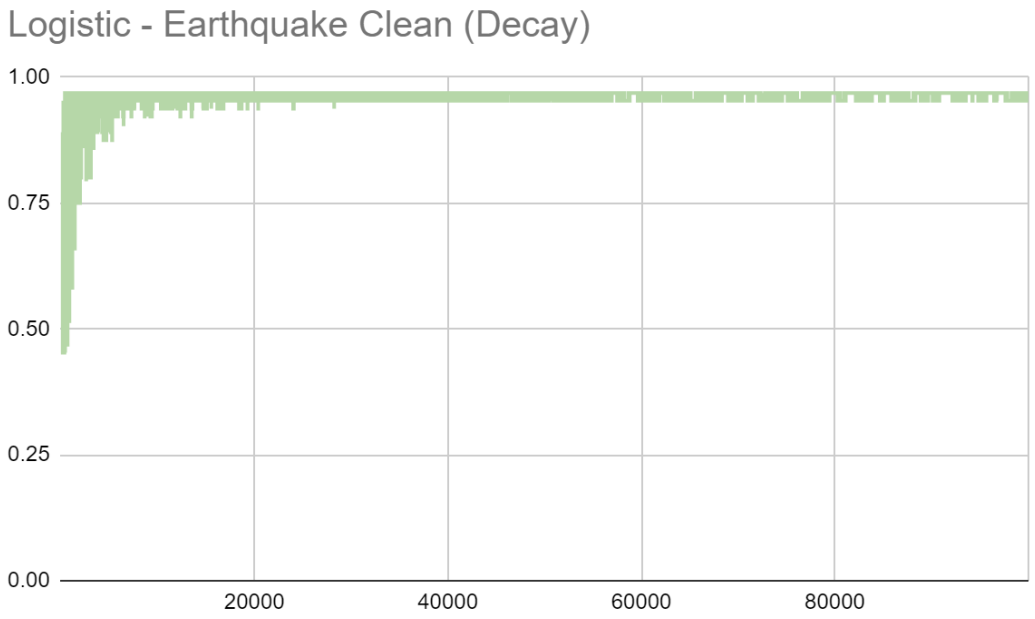
Logistic Classifier:



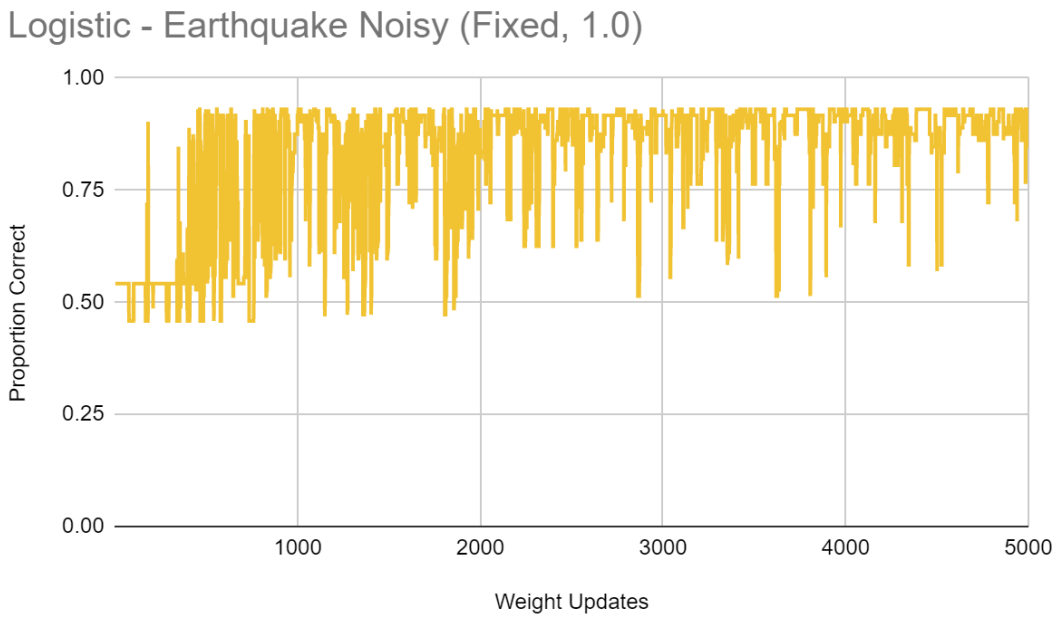

```
java MachineLearning earthquake-clean.data.txt logistic 1.0 0.0 5000
```



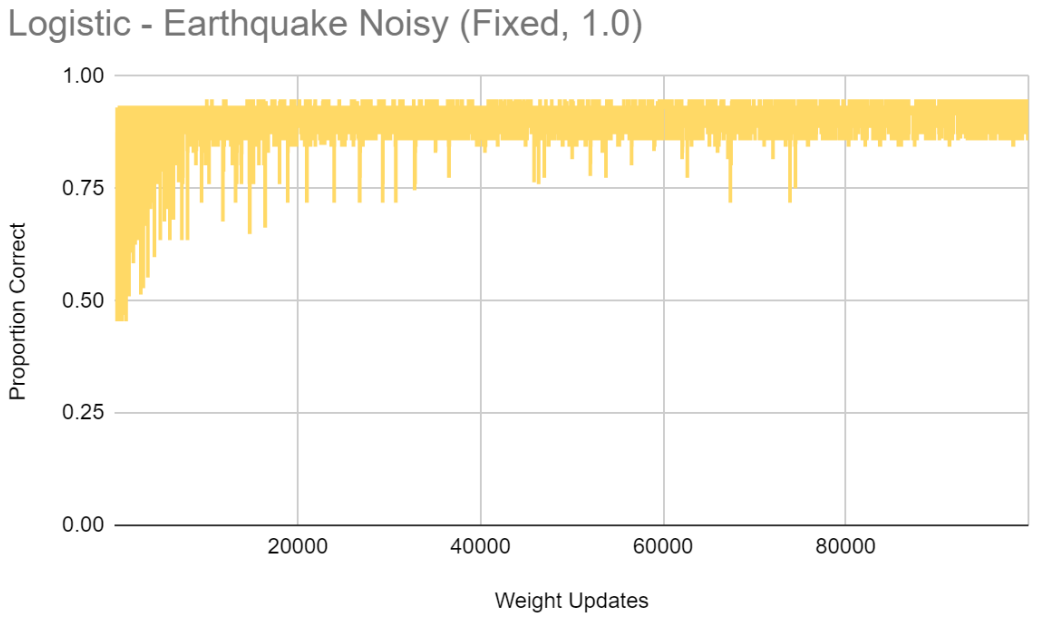

```
java MachineLearning earthquake-clean.data.txt logistic 1.0 0.0
100000
```



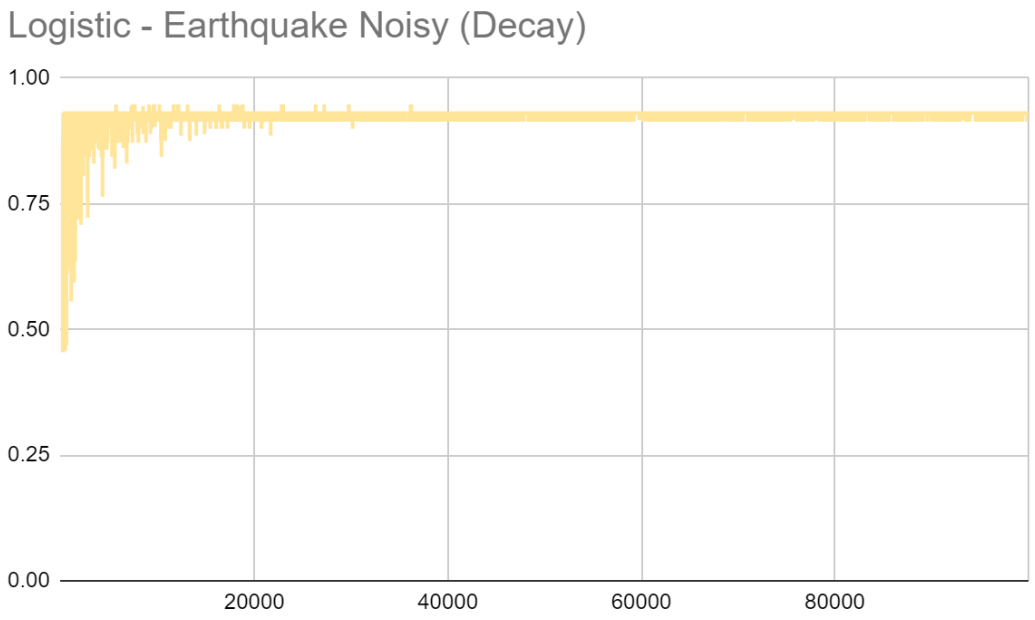

```
java MachineLearning earthquake-clean.data.txt logistic decay 0.0
100000
```



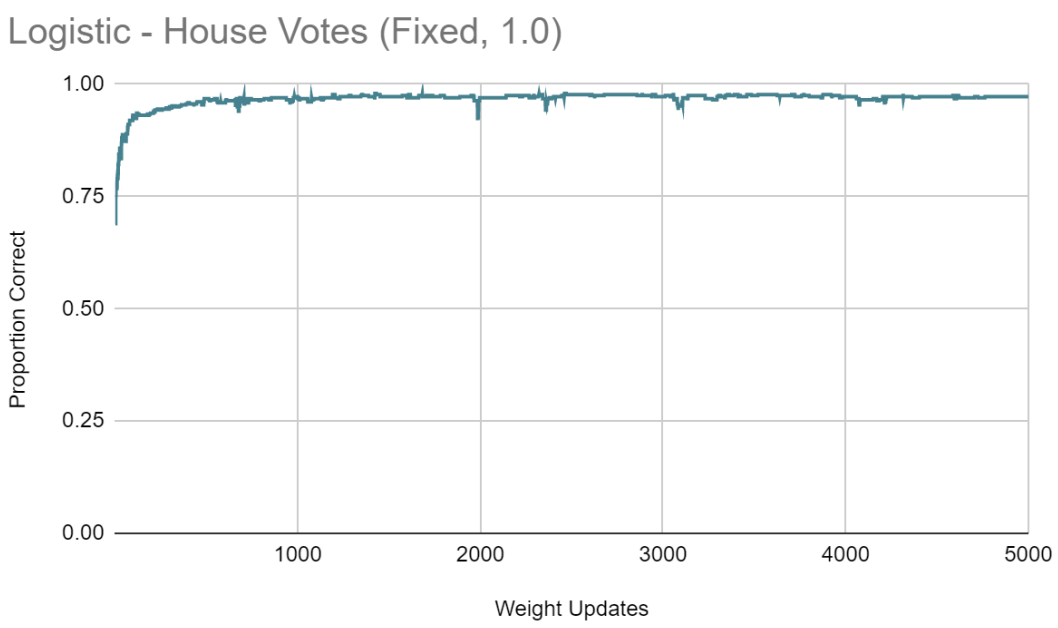

```
java MachineLearning earthquake-noisy.data.txt logistic 1.0 0.0 5000
```



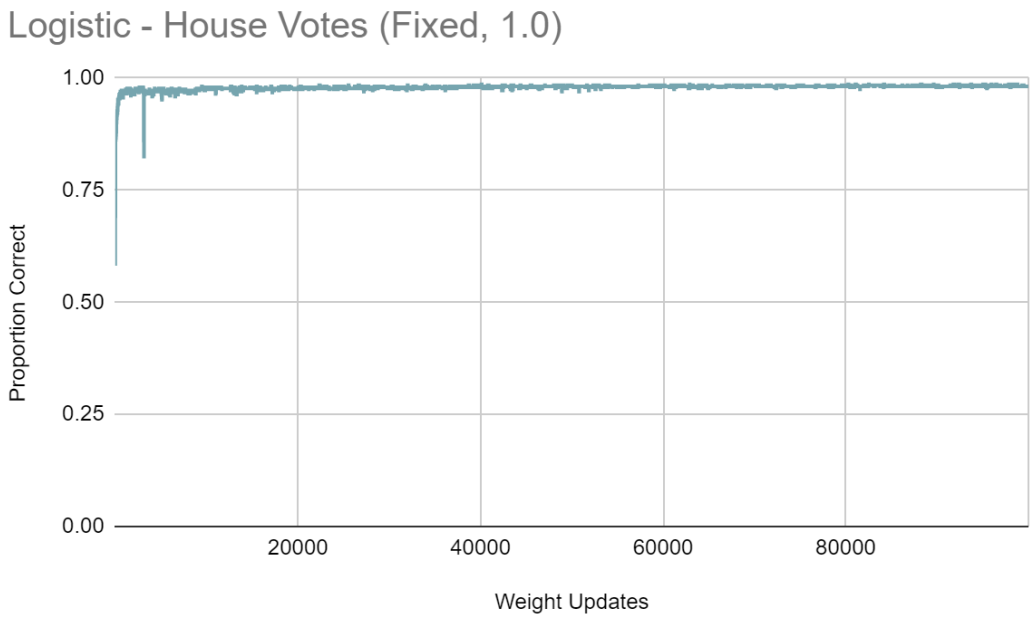

```
java MachineLearning earthquake-noisy.data.txt logistic 1.0 0.0
100000
```



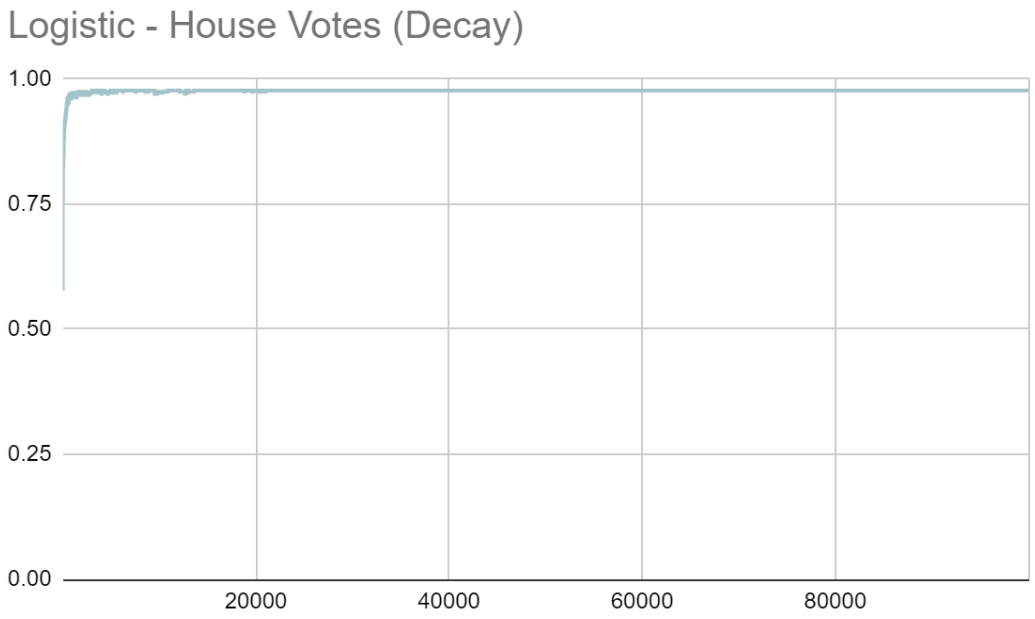

```
java MachineLearning earthquake-noisy.data.txt logistic decay 0.0
100000
```




```
java MachineLearning house-votes-84.data.num.txt logistic 1.0 1.0
5000
```

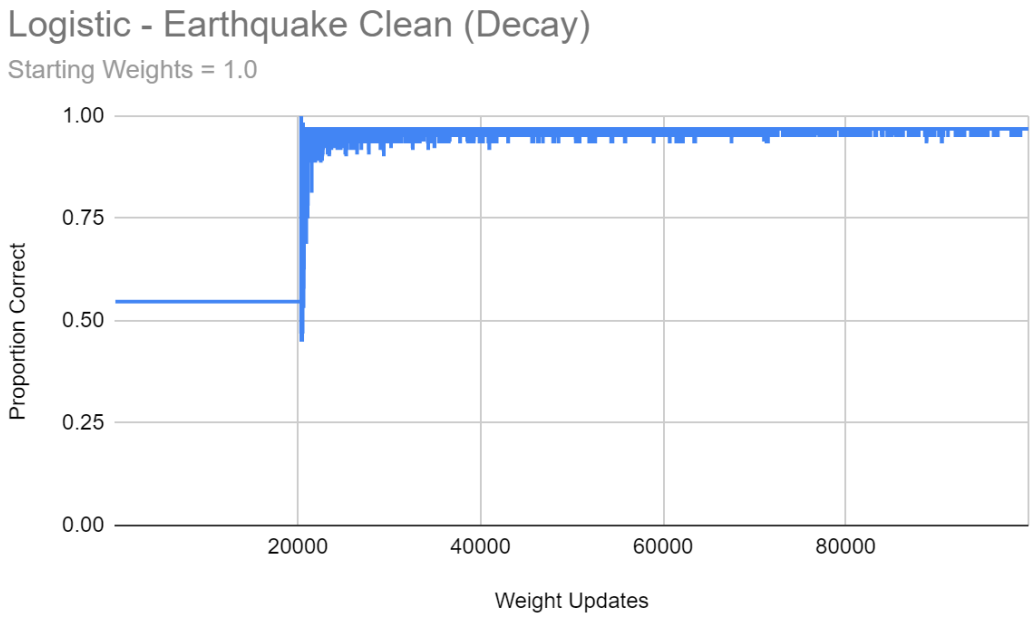



```
java MachineLearning house-votes-84.data.num.txt logistic 1.0 1.0
100000
```




```
java MachineLearning house-votes-84.data.num.txt logistic decay 1.0
100000
```

All the graphs displayed behave as shown in the textbook. The perceptron classifier's convergence is noticeably messier especially with a fixed learning rate. My logistic classifier's handling of the earthquake data specifically is also of note. For a reason myself, a TA, and the professor couldn't determine, it only behaves normally with starting weights at 0.0. Otherwise, it will remain at the same percentage correct for thousands of steps before eventually converging toward an expected outcome. For example, here is a graph of my logistic classifier for the clean earthquake dataset where the starting weights are 1.0. The weird thing is that the weights do



change between weight updates, but the accuracy does not change. Varying the starting weights changes how long it takes to start seeing a change in accuracy. The other weird thing is that I only saw this behavior on this specific dataset. The logistic classifier works as expected for the house votes data no matter the input I checked.