1. Pocket Algorithm

1.2.a)

The “tol” parameter determines how small of a difference there must be between two consecutive error values before training is terminated. If the difference between the previous loss and the current loss is less than “tol”, then training is stopped. This ensures that training ends once we are no longer making any significant reductions in loss.

1.2.b)

No, it does not guarantee that the algorithm will pass over the training data 5000 times. The “max\_iter” parameter only sets an upper bound for the number of iterations over the training data, so it is possible that the training will end before “max\_iter=5000” iterations are run due to the “tol” stopping criterion being met. We can ensure that the algorithm will pass over the training data “max\_iter=5000” times by setting the parameter “tol=None”, so that there is no stopping criterion for the training other than the maximum number of iterations.

1.2.c)

We can set a model’s weight values through the “coef\_init” and “intercept\_init” parameters in the Perceptron’s “fit()” function.

1.2.d)

The scikit-learn Perceptron performs slightly better than the NumPy implementation, as the scikit-learn Perceptron only has two false negatives while the NumPy implementation has three false negatives.

2. Linear Regression

2.1.a)

When fit\_LinRegr is called within subtestFn, the input to linalg.inv, X\_tr \* X, is a singular matrix. This is because the columns of input X\_train are not linearly independent and thus X\_train is a singular matrix.

2.1.b)

Linalg.inv raises “LinAlgError: Singular matrix”

2.1.c)

Linalg.inv computes the regular inverse of an invertible (non-singular) matrix, while linalg.pinv computes the Moore-Penrose pseudoinverse which exists for all square matrices and is equal to the regular inverse for invertible matrices. After using linalg.pinv, the model’s weights are [-2.10942375e-15 2.00000000e-01 4.00000000e-01] for subTestFN.