**Answer 1 C:**

Confusion Matrix:

TP FN FP TN

160.0 31.0 11.0 258.0 Decision Tree

160.0 31.0 21.0 248.0 Linear Regression

170.0 21.0 25.0 244.0 Logistic Regression

-------------------------------------------------------------------------------------

**Answer 3 B**: Purpose of training algorithm is to train the weights of the neural network in such a way that the 8 bit input is converted to 3 bit binary value and later converted back to 8 bit output, same as input.

**Answer 3 C:** This encoder-decoder neural network will not work with 1 or 2 hidden units. The reason being, network needs to return 8 bit output which will not be possible if hidden units are 1 or 2.

1 hidden unit network can only return two outputs and 2 hidden unit network can return 4 output. To be able to get all 8 outputs we need to have at least 3 units.

**Answer 5:**

**Andrew NG:**

Andrew NG wrote about below things to write better Machine Learning algorithms:

1: Debugging Learning Algorithm: If the model we are trying to build is not performing well, few common things to try are increasing the training data size, reduce the feature size, increase the feature size, and try different values for hyper parameters of the models and different convergence techniques (if possible).

A better approach would be to analyze the problem. Checking the bias and variance of the algorithm can tell us if we are overfitting the data or if the model is simply too complex (too many features).

In case of variance issue training error will be much smaller than the testing error. In case of bias issue training error will be high too.

Other things to check are if our algorithm is converging correctly and if the objective function we are trying to maximize/minimize is correct.

2: Error Analysis: If there are many components to the learning algorithm and they are setup in a pipeline structure (one component feeding data to another) we could check how much error each component contributes. Plugging ground truth for a component and checking the overall error could give us some insight about the error contribution of that component.

3: Ablative Analysis: This gives us a measure of correctness of our model. This tries to explain the difference between base performance and our model’s current performance. We should analyze how much each of the features is actually helping our model and get rid of those features which aren’t contributing much.

4: Getting started on a problem: There are two ways to start working on any problem

1: Spend time on understanding the problem, collecting the correct dataset and choosing the features and designing the algorithmic architecture of for the problem. This helps in getting the cleaner and more scalable implementation. After implementation, run it and do the analysis.

2: Implement a working solution quickly, do the error analysis and figure out what works and what doesn’t for the problem and fix those errors.

**Pedro Domingos :**

A machine learning problem consists of three components:

1: Representation: It deals with deciding the hypothesis space for the problem. Hypothesis set describes which kind of model we are looking for to solve the machine learning problem.

2: Evaluation: This deals with finding the objecting function for the classifier. Objecting function describes how good a classifier is performing.

3: Optimization: This deals with optimizing the objecting function of the classifier to get the maximum accuracy/efficiency from the classifier.

In Machine learning, we always care about the out of sample error or generalization. How our classifier performs on an unseen dataset decides how well /bad classifier is performing. Having low in sample error and high out of sample error is not something we want out classifier to have. This would simply mean that classifier has over fitted the data.

An over fitted model would perform badly on the unseen data. To avoid overfitting, we can use cross validation. Using cross validation to fix the value of hyper parameters is another way to avoid overfitting. We could also use regularization to prevent the overfitting and get simpler models.

A complex model with many features is something to avoid. Generalization becomes difficult in the higher dimensional models. These models are computing intensive too.

One of the best ways to ensure good efficiency of algorithm is to choose good features. Having lots of feature with no significant impact on output doesn’t help. Smaller and more related feature set generally lead to much better performance in the classifier.

Having more training data always helps. If we have sufficient training data, we could train a very complex model with lots of features too which would be very difficult if the data is less compared to the dimensionality of the feature set. If we have lots of training data, our model with many features will not be able to over fit the data and would generalize.