**Abe Melloh – Artificial Intelligence Final Project**

1. **Your objective should be stated in written form. What are you trying to accomplish? Predicting a number? Classifying? Your objective must reference the context of the problem, specifically.**

The purpose of this A.I. final project is to use the Bank Marketing Data to classify if clients of a Portuguese bank would subscribe to a bank term deposit (yes or no) depending on their answers to phone survey questions.  The data comes from <https://archive.ics.uci.edu/ml/datasets/Bank+Marketing> in two distinct versions-- a newer, more robust dataset with 20 inputs, and an older version with 17 inputs.

1. **Your final ANN model, in code.**

(See attached python file)

1. **Your final model and training algorithm, in words.**

The description below details how I used an ANN to predict which customers of a Portuguese bank would subscribe to a bank term deposit.

* 1. Import the CSV
  2. Converted the variable names to a list
  3. Create a list of the categorical variable names to eventually code into numeric values so I could put them through the ANN.
  4. Create dummy variables for the categorical variables
  5. Preprocessed the categorical variable data using min\_max\_scaler
  6. Put the categorical variable data into a data frame and then concatenated it with the rest of the original data.
  7. Spit the data into my independent and dependent variables. (If you look at the code, you’ll notice the names iris.target and iris.data. The iris name is kept from older code that was used in a previous neural net.)
  8. Create train and test sets and then scale them.
  9. Send the data through the ANN.
     1. A few things to note about the ANN. It’s a classifier ANN because we are simply trying to predict whether or not someone is going to make a term deposit—“yes or no”.
     2. It uses three layers
     3. It uses the vanilla sgd
  10. Fit the x and y training data
  11. Predict y\_hat
  12. Estimate the accuracy of the classifier on the x and y train by splitting the x\_train and computing the cross validation score five times, with different splits each time.
  13. Print the scores of the cross validation, the training mean accuracy, and the accuracy score of the y\_hat and y\_test.

1. **Your experimental plan for arriving at the final model.**  
     
   For arriving at the final model, I wasn’t going to reinvent the wheel. I first used the Iris classification code that we commented out for our last assignment. I used that and then tweaked my Portuguese dataset to fit into the model. Mostly this meant I was going to need to code out numeric values for the classification variables. I adjusted some parameters within the layers because I knew I didn’t want to wait a long time for the model to run a whole day for the model to run—as I knew I could possibly facing several hours of run time even with less stringent parameters. I was essentially willing to sacrifice a bit of accuracy so I could keep working on the project. I switched the learning rate from .001 to .01. and lowered the number of iterations from 200 to 100.
2. **How long it took to run all the models in your experimental plan.**  
     
   It took 33mins and 58 seconds to run the model.
3. **An explanation of the input variables and any preprocessing steps you took.**

The input variables were as follows:

(Note: Categorical variables that are written in bold below were numerically coded into multiple columns and didn’t get inputted directly, but rather as additional columns for each categorical subtype. For example, the variable Education didn’t get inputted as a single column, but rather as a column for each level of obtainment. This means that “primary” education would in fact have its own column inputted, meaning the column is simply zeros and ones—where 1 if “primary” education is the highest level of obtainment, 0 if otherwise. Same for “secondary” education, “tertiary”, etc.):

* Age (Age of the individual)
* **Job** (Occupation type- ie: management, blue-collar, retired, technician, etc)
* **Marital** (Single, Married, or Divorced)
* **Education** (Highest Education level – ie: “primary”, “secondary”, “tertiary”)
* **Default** (Has the individual ever defaulted on a loan, yes or no. (Originally coded as strings.))
* Balance (How much money is in the account)
* **Housing** (Does the individual have a housing loan, yes or no. (Originally coded as strings.))
* **Loan** (Does the person have a loan, yes or no.)
* **Contact** (How was the individual contacted: “phone” ,“cellphone”, “mail” etc.)
* **Month** (What month was last contact made)
* Duration (How long was the duration of last contact)
* **Campaign** (How many times was the individual contacted during the campaign)
* Pdays (Number of days passed by after the individual was contact from the previous campaign)
* Previous (Number of times the individual was contacted in the last campaign)
* **Poutcome** (Outcome of the previous marketing campaign—“failure”, “success”, “nonexistent”.)
* **Y** (Has the client subscribed to a term deposit, “yes” or “no”)

The Y variable is what we are trying to predict-- whether or not an individual made a term deposit. The column Days was removed from the data as coding out multiple days of the month would have led to a much larger data frame once the data had been numerically coded. Columns that has “Unknowns” were left. NAs were imputed with the mean.

1. **An explanation of your metrics and justification for your choice.**

I looked for accuracy because the model is a classification model. The success of the model lies on how well I can predict if an individual is going to make a term deposit.

1. **An explanation of your method to validate the model.**  
     
   I validated the model computing the cross validation score five separate times, with each computation using a different split of the data to reduce variance.
2. **Your results in terms of appropriate metrics for the objective and problem.**

The train mean accuracy was 88.9% while the accuracy of the vanilla sgd test was 88.8%.

This is pretty good news. This means that I can take new data that has the same variables as my training set and should be able to predict whether or not an individual makes a term deposit with 88.8% accuracy.

1. **A discussion and/or justification for how you used/didn’t use all of the following:**
   1. **selection of the optimum number of units** 
      1. The general rule of thumb is select a number between 1 and the number of input variables. Since I have 15 input variables, I went with 13 units.
   2. **type of network (feedforward, recurrent, backpropagation, and etc.),** 
      1. I went with a feedforward network as it is relatively simple and does a good job in classifier ANN models.
   3. **type of training (supervised, unsupervised),**
      1. Supervised learning because we know what we are trying to predict—whether or not someone makes a term deposit.
   4. **proportion of training and testing data sets (70:30, 80:20, and etc.),** 
      1. I used an 80/20 split because it’s the default when doing training and test sets.
   5. **number of input and output units (usually application dependent),** 
      1. I have 15 initial input variables and 1 output.
   6. **number and size of hidden layers (2N+1, experimental),**
      1. 3 hidden layers with 13 units. Three layers were used as this was the default of the Iris model.
   7. **number of repetitions during training (epoch),** 
      1. 100 iterations were used. I wanted something that would be sufficient for the train data but not cause my model to take forever to run.
   8. **choice of activation function (sigmoid, linear, Tanh, ReLU, and etc.)** 
      1. The activation function was a Softmax because that’s what you use when doing a classification model.
   9. **size of data set (number of records)**
      1. The dataset was 45212 rows.
   10. **learning rate**
       1. The learning rate was set at .01 to make it run a bit faster, as I was willing to sacrifice some accuracy for time.
   11. **momentum.**
       1. Not used because it was not in the iris code. Not particularly relevant here.
2. **Discussion and further work.**

Considering the computer issues I had, I focused more on getting a classifier to run than coming up with a really high accuracy rate. That aside, an accuracy rate of 88.8% is rather good. I think with some more time I’d adjust the number of layers, the learning rate, and the number of iterations (epochs) to see if I couldn’t increase the accuracy a bit.