**Feed Forward Network: Appraisal of Diamond Value**

Kaden VanPortfleet

Davenport University

CSCI 325: Deep Learning

Dr. Naiwei Liu

October 16, 2023

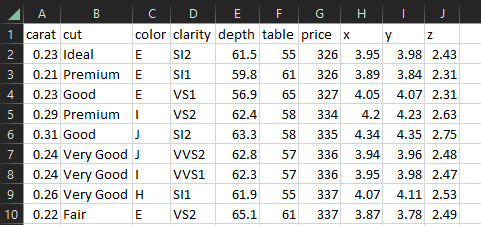
**Introduction:**

The process of evaluating the value of a diamond requires significant measurements of multiple qualities. The process of which can be quite tedious. Aspects of each diamond such as carat, clarity, color, cut, and its size and dimensions are all significant factors in the appraisal process. Given these measurable characteristics and understanding that these variables all contribute to the cost of a diamond, we can quickly see how using algorithms to aid in the appraisal process can provide useful. In this paper, we will discuss the process of setting up, training, and utilization of a neural network.

**Dataset & Pre-Analysis:**

Like all neural networks, we will need a hefty dataset in order to train our network. The larger the dataset, the more accurately our model should be able to predict results of new data it is fed. For our network, the dataset we are going to use is the “Diamonds” dataset posted by Joakim Arvidsson on Kaggle.com (Arvidsson, 2023). This dataset original comes from GitHub repository “tidyverse/ggplot2” as part of its data library (Tidyverse, 2023). The file is in the .*CSV* format.

Before importing the dataset, we should look inside the data for any pre-conditioning that may be needed:



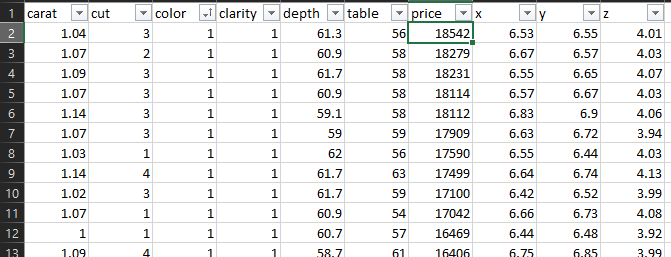
We can see 10 variables for each data point:

1. Carat – This is the weight of the diamond. An increase in this value should naturally cause the value to increase. The carat and price are directly correlated.
2. Cut – The quality of the shaping of the diamond. A higher rating/increase in this value should naturally cause the value to increase. The cut and price are directly correlated.
3. Clarity – The transparency of the diamond. A higher rating/increase in this value should naturally cause the value to increase. The cut and price are directly correlated.
4. Table – This variable is the percentage width that the top of the diamond is compared the maximum width. This value is highly subjective and does NOT clearly correlate with price of the diamond.
5. Depth – The height of the diamond. A large depth could indicate a larger diamond, but an abnormally high or low depth-to-width ratio could harm its aesthetics. This value is loosely correlated to the price.
6. X, y, z – The dimensions of the diamond in millimeters. These values determine the size of the diamond. While having these values grow larger should increase the value, certain combinations of them may be deemed unattractive. These do not clearly correlate to the price.

Remember, our goal is to evaluate and appraise these diamonds and return an estimated price given the above details. Seeing as some of our variables are not based on correlation but instead on subjective looks, this problem is not very suitable for regression. Instead, we can leverage a deep learning neural network to predict our diamond prices given a history known prices.

**Data Cleaning & Reformatting:**

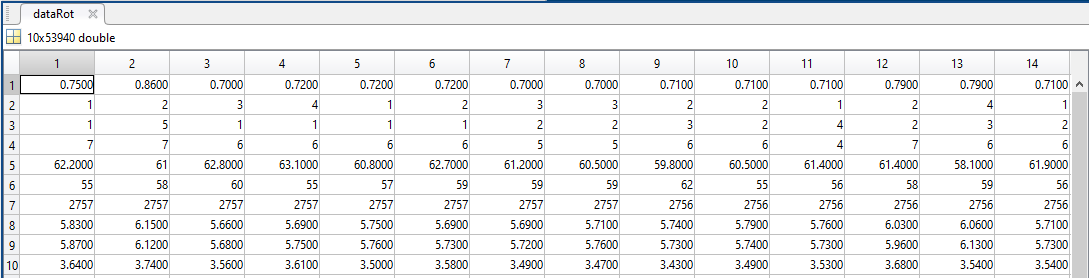
Looking at the first couple rows of our dataset, we can see that we have a mixture of different data types. For our purposes (building a neural network), we would like to have all of these values be the same type. This way, when we convert this data to a matrix (more on that later), multiplication of weights can happen to all parameters at the same (via matrix multiplication).

For our specific instance, we can map the values for ‘color’ and ‘clarity’ to scalar integers based on ranking. Color is defined from ‘D’ to ‘J’, where each letter is a specification of color from colorless to faintly colored. For our purposes we can re-label these 1 through 7 (1 being clearest, 7 being the most colored). Additionally, the specification of clarity is a rating that has a hierarchy from ‘I1’ to ‘IF’. For use in the neural network, we will map these rankings to numbers 1 through 8, with 1 being the highest clarity and 8 the lowest. This is the resulting reformatted data:  
  
*Note: The above picture is sorted for display/usage purposes. The dataset used in the training is NOT sorted and is left raw.*

**Code / Data Preparation:**

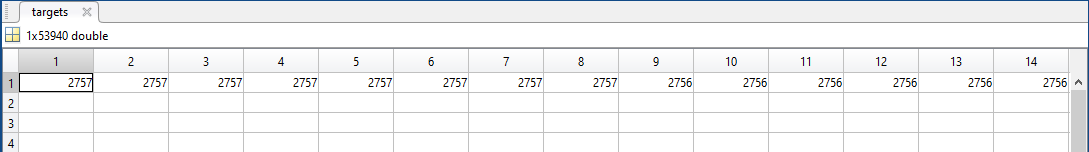
To begin, let us import this data into our coding environment. For this project, we used MATLAB:  


In the above screenshot, we can see the method ‘*readmatrix()*’ is used to import the *.CSV* file as the variable ‘data’. Additionally, we would like to have each data point be a distinct column. In our dataset, it is instead represented by a row. To remedy this, we rotate the matrix -90 degrees using ‘*rot90()*’:  


Now we have the following matrix:  


It is 10 rows by 53940 columns (i.e., 53,940 datapoints with 10 attributes each).

This imported data contains our ‘target’ value of price. Remember, the goal of the neural network is to appraise a diamond based on its attributes and return a price. Since our dataset includes our targets, we can extract these values and use it to train against. First, let’s create a new 1x53940 matrix with the price information:

The above code returns the 7th row at all columns from our rotated dataset. This is the returned matrix:  


Since we have isolated the ‘target’ values, we can remove them from the dataset:  

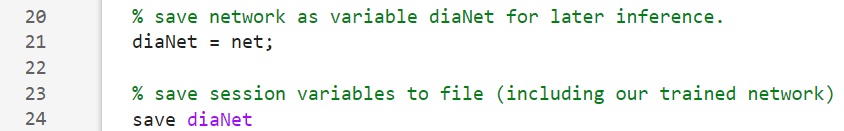

We create a new matrix ‘*dataPrep*’ with the all the rows of ‘*dataRot*’ *except* the price row (row 7).

**Training:**

With our data prepped and our targets defined, we can begin training. MATLAB includes a built in method of creating a feed-forward neural network called ‘*feedforwardnet()*’ we initialize a new neural network of this type as ‘*net*’ (MATLAB, n.d.):  

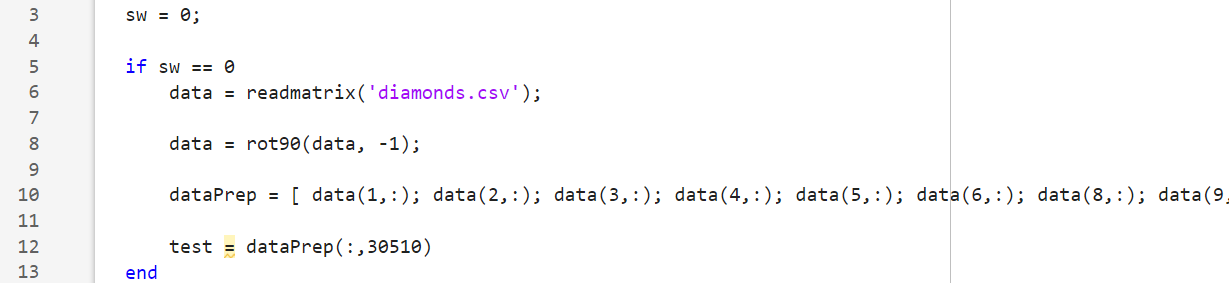

The value passed to the ‘*feedforwardnet()*’ method defines the number of hidden layers in the network. We will choose 10 for simplicity as it provides adequate accuracy and is very quick to train. We then can pass in our network, data, and targets to the method ‘*train()*’:  


*Note: The size of the dataPrep and targets matrices must have the same number of columns in order to train. Each data point is represented by a column.*

Finally, we can save the network to a named variable ‘*diaNet*’ and save the session variables to a file called ‘*diaNet.mat*’:  


**Inference & Output Analysis:**

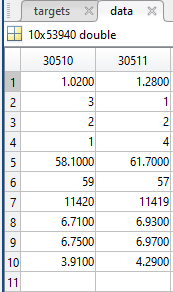
We now have a neural network saved in our session variables file ‘diaNet.mat’. In order to use this network, in a separate script, we load the session variables with the ‘*load()*’ method:  

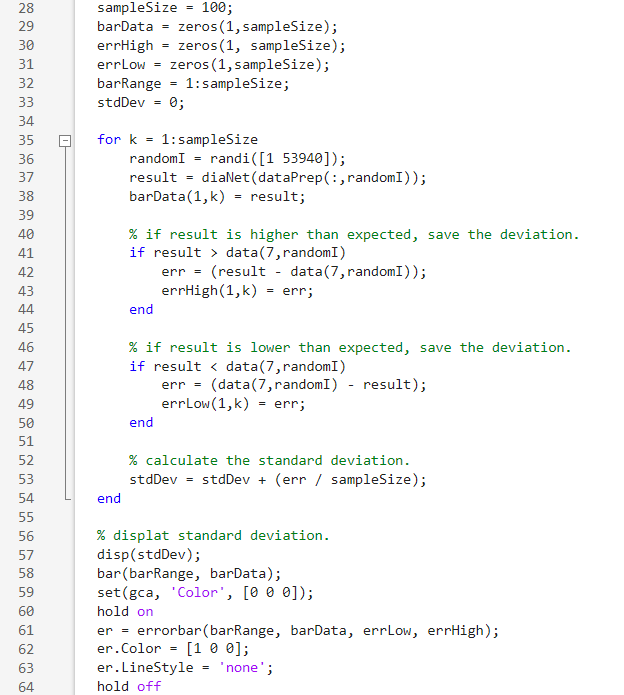

Here, we are loading the ‘*diaNet*’ variable from the ‘*diaNet.mat*’ file (this is our neural network we trained). To check its accuracy, we can choose a random data point and feed it to the network (data point 30510 was chosen randomly):  


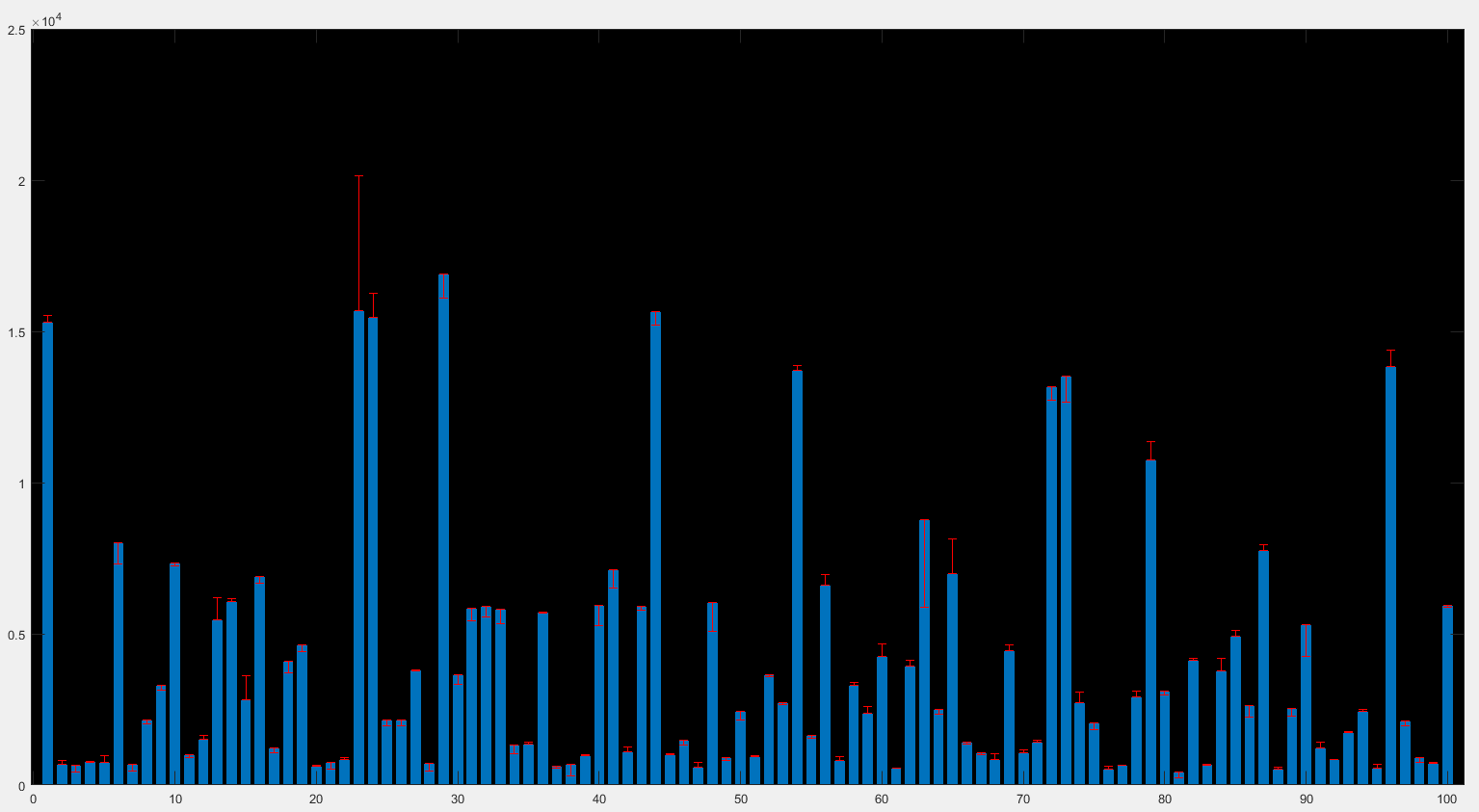


*Note: Gap in lines is due to logic control. After line 12, variable ‘*test*’ is not altered while ‘sw’ == 0.*

We get the result:  


According to our neural network, the diamond specified at column 30510 should be valued at $12,284.15. Let is cross-reference this value with its known appraisal value:  


Our neural network is over-estimating the price of this diamond. The difference in value is $864 or 7.28991%. Despite $864 sounding like a large delta, a ~7% error is quite respectable and perfectly viable for getting an idea as to the value. But this is just a single data point. Let us randomly sample 100 data points and check the accuracy of the neural network:  


In a nutshell, the above code defines a sample size and constructs a bar chart of the values of randomly sampled data from the dataset ran through our neural network. We then add error bars to the data that represent what the *actual* value of that diamond is supposed to be. This yields the following chart:  


As we can see that the accuracy of the network is quite good. There are a few outliers such as bar 24 which undervalued that specific diamond by ~$500. In the code above we also calculated the average deviation of the network’s output compared to the expected. For this particular iteration, the program reports that the neural network averaged an error delta of 319.3665. Meaning, the network on average was off by $319 on its estimations.

Reference:

Arvidsson, J. (2023, September 28). Diamonds. Kaggle. <https://www.kaggle.com/datasets/joebeachcapital/diamonds>

MATLAB. (n.d.). MATLAB | Documentation. MATLAB & Simulink. <https://www.mathworks.com/help/index.html>

Tidyverse. (2023). Tidyverse/GGPLOT2: An implementation of the grammar of graphics in R. GitHub. https://github.com/tidyverse/ggplot2