*The Network*

My network is a dense network consisting of 5 hidden layers of 1000, 500, 250, 125, 19 and 5 nodes respectively. It is configured to run for 3 training epochs in batches of 5 examples. For the estimator, I chose to use the premade DNNClassifier class with 4 classes. All settings are left as the defaults except for the loss reduction function which is set to sum over the batches instead of each single example. I came to these settings primarily by trial and error. I experimented with different optimizers but none gave a measurable performance boost. The network size, batch size, and number of training epochs are set globally for ease of adjustment. This network consistently classifies approximately 70% of validation examples into the correct category with certainty greater than 70% using the dataset described below.

*The Data*

I used an integer between 0 and 3 to code the population quartiles. Department, Year and Population columns were dropped, as was a column titled Randomizer, which I created to aid in shuffling the data in MS Excel. All numeric columns were normalized to standard deviations from the column mean. This had the greatest effect on the performance of my network. No configuration performed better than random on the raw data and no network performed worse than 50% on the normalized data with most being in the 60% – 70% range. The normalization was also done in Excel but I’ve become aware that the Pandas library also allows this functionality. I may try to implement that in a later revision.

*Other Thoughts*

For this dataset, the number and depth of the hidden layers seems to have very little effect on overall performance. In fact, a network with no hidden layers performs about as well as this one given the same data.

In a future revision, I would like to implement some sort of genetic modification scheme to improve the performance of the network. It would also be interesting to record the configurations and outcomes to see if I derive the type and degree of effect that tuning each hyper-parameter has.