CS 461
Program 3 – Neural Networks

For this program you'll be analyzing occupational data and trying to predict who might be open to changing jobs.

The data set originally comes from: https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists

You are given a CSV file containing a little over 19,000 data cases. The fields you have available are:

- enrollee_id : Unique ID for candidate. This is an identifier and should not be used as part of your classification model.
- city: City code. This is a numeric identifier for the city and again, should not be used as part of your classifier *unless* you choose to build a new synthetic variable based on how many cases are available from each city (a rough index of city size).
- city_ development _index : Development index of the city (scaled). A real number in 0-1, higher numbers are higher levels of development.
- gender: Gender of candidate. Missing data in many cases. The set is gathered from people who are working in (or thinking of working in) data science; it is, not surprisingly, overwhelmingly male.
- relevent_experience: Relevant experience of candidate. String coding, but value is boolean ("Has relevant experience" or "No relevant experience")
- enrolled_university: Type of University course enrolled if any.
- education_level: Education level of candidate. Graduate & Masters are most common, but other values at both ends exist. Your option as to how many of the other values to include before you code everything else into a residual "other" category.
- major_discipline :Education major discipline of candidate. 76% STEM, 15% blank, 9% all others. Your option as to how many of the others to specifically encode.
- experience: Candidate total experience in years. Note that ">20" is a value, as is "<1", so it's not just numbers.
- company_size: No of employees in current employer's company. Similar, but ranges: "<10", "50-99," etc.
- company_type : Type of current employer. Text, only a few different values.
- last*new*job: Difference in years between previous job and current job. Numeric ranges, including "never," ">4", etc.
- training_hours: training hours completed: integer.

- target: 0 – Not looking for job change, 1 – Looking for a job change. This is the output variable that you are trying to predict.

You will want to clean the data file up before turning it over to your neural network, removing unneeded columns, recoding data into a consistent format, etc. You do not need to turn in your code or any intermediate files for this; I assume you can manipulate text files by this point in your programming career.

Your task is to construct a neural network to predict who is likely to be looking for a new job.

- You may use either TensorFlow or Kattis for this. TensorFlow is installed on all Flarsheim labs, including Remote Labs and the Remote Connection into Flarsheim workstations.
    - You can install either on your own machine if you want to, of course, but are not required to do so. Both packages are fairly large and resource intensive. Also, they use the GPU for computing where possible.
    - If your laptop uses the integrated graphics on the motherboard, you would probably have to download the source and compile it, and performance won't be very good.
- Build a feed-forward network with 2 neurons in the output layer. Experiment to see what works better, but for this small number of variables and relatively small number of cases (and for neural networks, 19,000 cases is a small data set), you won't need hundreds of neurons in layer upon layer. In fact, I'd suggest not going above 2 hidden layers unless you have evidence it improves the model. (The bigger the model, the more data-hungry it is.)
- As most variables are categories, you'll probably use 1-hot coding for most variables. (The development index is an exception, obviously.)
- Divide the data 70% training, 15% test, 15% validation. Note that the data is unbalanced; the number of people looking for jobs is much smaller than those not looking. You will need to take this into account in building your model.
- Use a linear or tanh transfer function at the output later (i.e. either leave it untransformed as a weighted total+bias, or to a sigmoid with range ±1) and use softmax to turn your output into a probability.

Once you've built your model, write a short report describing your model (how many layers? How did you train it? How did you cross-validate? Etc) and reporting its overall accuracy. Provide references for any code you used that you did not write yourself. Submit your report along with your source code and cleaned data file.

ACADEMIC HONESTY: Any code you submit for a grade is expected to be your own work. However, discussing different network configurations, pros and cons of various approaches, comparing results, etc., is fine. So are questions like "why am I getting this error message?," "what does this parameter in

the API do?," "I'm getting numbers but I'm not sure what they're telling me," etc. Pseudocode or proof-of-concept code is fine as well.