Challenge Problem 1: Computer system failure data analysis

Created: *March 23, 2020*
Last updated: *March 29, 2021*

This assignment deals with predicting failure of application executions (referred to as "*job*") on Purdue ITaP's central computing cluster. This is data that we have collected, collated, and analyzed as part of a project from the National Science Foundation (NSF) (Computer System Failure Data Repository to Enable Data-Driven Dependability Research," Proposal No. CNS-1513197).

For each job, we have data about the resources the job uses and whether the job succeeded or failed. The resources for which we have data are:

1. Memory
2. Network
3. Local IO
4. Network File System (NFS)

We are releasing the training data, which has about 8% failure data (this is referred to as the "*positive class*"). You will build Machine Learning models in Python to predict whether a job will fail or not, given the resource usage data. We will evaluate your model on some test data that we are not releasing now and that we will use later at the time of the evaluation.

The assignment, including the data, is available at the following Github repo:

https://github.com/bagchi/application-failure-prediction

You will do the following steps for this assignment:

1. You will write code in Python. We have found the following packages to be useful for this task — `pandas`, `sklearn`, `numpy`.
2. You will use the training dataset `train_data.csv`.
3. You will create a Machine Learning model that achieves the highest balanced accuracy (giving equal weight to the positive and the negative class accuracy). We will call this "*Model Complete*". You can start with a Support Vector Machine (SVM) variant as a starting point, but will want to look at more advanced models beyond that.
4. Now suppose that you are only allowed to use three features from the dataset. You will use Principal Component Analysis (PCA) to identify which three features to use. You will again try to achieve the highest balanced accuracy with the three feature model. We will call this "*Model Abridged*".

5. Give a short writeup explaining:
    a. What did you do for the *Model Complete*? Give the confusion matrix and the balanced accuracy that you obtained on the training data.
    b. What did you do for the *Model Abridged*? Give the confusion matrix and the balanced accuracy that you obtained on the training data.
    c. Compare the result you achieved for *Model Complete* vs. *Model Abridged*.
    d. Upload this writeup.
6. Upload your Python code with a README that tells me how we can run your code.