Intelligent Systems Engineering
ECE 449
Fall 2023

Laboratory Exercise #1

Resources: *Machine Learning with Python*, from Cuantum Technologies (required text).

- Chapters 2.2-2.5 for the Pandas, Matplotlib, Seaborn and Scikit-learn libraries
- Chapters 3.1-3.5 for data preparation
- Chapters 6.1-6.2 on multilayer perceptrons in Tensorflow / Keras.

Wine Dataset: https://archive.ics.uci.edu/dataset/109/wine

Cross-Validation: https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f

Stratified Sampling and Cross-validation: https://towardsdatascience.com/what-is-stratified-cross-validation-in-machine-learning-8844f3e7ae8e

Summary: You are to employ the multi-layer perceptron (MLP) network, using the backpropagation with momentum learning algorithm, to solve the Wine Dataset classification problem. Wine is a 3-class problem with 13 predictor attributes, which you can also download from the UCI Machine Learning Repository. You will use Tensorflow and Keras to build your multilayer perceptron, again using an automated pipeline from data ingestion through parameter exploration (this time using a tenfold cross-validation design) to the final evaluation of your MLP model.

Discussion: Neural network modeling involves several activities: preprocessing the data (including normalization, missing values imputation, and encoding), network design, and parameter exploration. This project will expose you to all elements of this process. You are already familiar with normalization, and there are no missing values in this dataset. The Wine dataset is a 3-class problem, the usual practice in neural network modeling is to use a "one-hot" encoding to represent a categorical variable with more than two values, so your class labels should be one-hot encoded. Use a Softmax activation in your last layer to train the network to recognize this encoding.

For your experiments in the Wine dataset, you will randomly split the Wine dataset into a "training dataset" and a holdout testing dataset; there are 178 examples, so reserve 10 % (18 randomly-selected examples) as the test set. Note that you should keep the class frequencies the same in the training and testing datasets, so you will need to use *stratified sampling* to divide your train and test sets. Functions for doing so are contained in Scikit-Learn,

You will perform a parameter exploration of the MLP using the tenfold cross-validation design in the training set (again using stratified sampling), and determine the best parameter settings based on the validation error. Then, train a fresh MLP using the whole training set and those best parameter settings, and determine the final out-of-sample error using your test dataset.

Requirements: You will ingest, preprocess, train and evaluate your MLP using an automated pipeline. Your parameter exploration must include both the network design (number of layers, neurons per layer) and the parameters of the learning algorithm (I suggest using the Adam optimizer, as it is a very common choice). Then you will evaluate the "best" choice of network design & parameters on the test dataset. Report the F1 score of your final experiments.

Tensorflow is not part of Scikit-Learn, so you most likely will not be able to use the "pipeline" container object as you did in Lab #1. Instead, you will create your own pipeline by passing the output of one library function to the next, and adding in the control flow needed to organize the tenfold cross-validation experiments.

Submission: All of the above should be automatically performed in a Python-based pipeline, and submitted as a Jupyter notebook. The TAs will grade your work by executing the notebooks on a fresh copy of the dataset, downloaded from the same source. Submit the notebooks via eClass in your lab section before the deadline.

Grading

MLP trained via cross-validation:	75%
Pipeline automated:	25%