CS 534: Homework #5

Submission Instructions: The homework is due on Nov 17th at 11:59 PM ET on Gradescope. A part of your homework will be automatically graded by a Python autograder. The autograder will support Python 3.10. Additional packages and their versions can be found in the `requirements.txt`. Please be aware that the use of other packages and/or versions outside of those in the file may cause your homework to fail some test cases due to incompatible method calls or the inability to import the module. We have split homework 1 into 2 parts on Gradescope, the autograded portion and the written answer portion. If *either of the two parts is late, then your homework is late.*

1. **(5+3+2=10 pts)** (**Illustrating the "curse of dimensionality"**)

   (Written) For a hypersphere of radius a in d dimensions, the volume is related to the surface area of a unit hypersphere (S) as

   $$V = \frac{S \times a^d}{d}.$$

   (a) Use this result to show (via a formal mathematical proof) that the fraction of the volume that lies at values of the radius between $a - \epsilon$ and a, where $0 < \epsilon < a$, is given by $f = 1 - (1 - \epsilon/a)^d$. Hence, show that for any fixed ϵ, no matter how small, this fraction tends to 1 as $d \to \infty$.

   (b) Evaluate the ratio f numerically by plotting the results for different values of $\epsilon/a = 0.01$ and $d = 1$, 10, 100, and 1000.

   (c) What conclusions can you draw from the plot?

2. **(2+3+2+5+2+5+4+3+5+5+3=39 pts) Preprocessing Loan Defaults Using PCA & NMF**

   Use the `sklearn.decomposition` package to perform dimensionality reduction on the `loan_default.csv` problem from Homework #3. The functions specified should be in the file 'q2.py'

   (a) (Written) How did you partition the loan data to assess the model performance and choose the hyperparameter (model selection)? This can be repeated from Homework #3 and Homework #4. You might find it useful to read all of this homework to decide if you want to change your assessment strategy.

   (b) (Written) Pre-process the `loan_default.csv` data from Homework #3 to deal with categorical data. Specify how you are specifically converting the non-numeric data.

   (c) (Code) Write the function `preprocess` which takes in a pandas dataframe (a direct result from `pd.read_csv(loan_default.csv)`) and outputs a 2D numpy array.

   (d) (Written) Based on your data from 2c, your model assessment strategy in (a), and your functions from Homework #3, perform feature selection to determine which variables you plan to keep. You should provide the rationale for *your feature selection process.* You may find it useful to save the train/test data as a separate file (e.g., `loan_fs_train.csv`, `loan_fs_test.csv`) as you will be analyzing its performance below.

   (e) (Written) Discuss why you should normalize the data before performing PCA.

(f) (Code) Write a function `run_pca(train_x, test_x)` which will take 2 inputs, `train_x` and `test_x`, both of which are 2D numpy data following the format of 2c, normalizes both numpy data, determines the number of components to capture 95% variance of the normalized training data, and returns the principal components and the loadings. Note that the normalization should be learned from the training data and applied to the test data. The function should output a tuple in the following order (1) the number of components as an integer, (2) the principal components as a numpy 2d array of shape (components $\times$ features), (3) the train loadings as a numpy 2d array of shape train samples $\times$ components, and (4) the test loadings as a numpy 2d array of shape test samples $\times$ components.

(g) (Written) Using the PCA function you wrote above in 2f, report how many components were needed to capture 95% of the variance in the normalized data. Discuss what characterizes the first 3 principal components (i.e., which original features are important).

(h) (Written) Using the PCA function you wrote above in 2f, transform your train and test data using PCA. You might find it useful to save the train/test data as a separate file (e.g., `loan_pca_train.csv`, `loan_pca_test.csv`)

(i) (Code) Write a function `run_nmf(train_x, test_x, k)` which will take 3 inputs. `train_x` and `test_x` are 2D numpy data following the format of 2c, and k is an integer. The function should output a tuple in the following order (1) the reconstruction error (a float), (2) the factors components as a numpy 2d array of shape (k $\times$ features), (3) the train loadings as a numpy 2d array of shape train samples $\times$ k, and (4) the test loadings as a numpy 2d array of shape test samples $\times$ k.

(j) (Written) Using the function above, 2i, determine the optimal number of components. You should justify your choice via a plot of the reconstruction error as a function of k. Discuss what characterizes the first 3 factors (i.e., which original features are important).

(k) (Written) Using the NMF function you wrote above in 2i, transform your train and test data using NMF and the optimal k you found from the previous question. You might find it useful to save the train/test data as a separate file (e.g., `loan_nmf_train.csv`, `loan_nmf_test.csv`)

3. **(3+4+6+7+3+20+8=51 pts) Model Bake-off for Predicting Loan Defaults**

We will consider many of the models we learned in conjunction with feature selection and dimensionality reduction and determine the 'best' model for this dataset. The functions specified should be in the file 'q3.py'. All the models will be evaluated in terms of AUC, F_1, and F_2. Note that the input for each of the following functions below should be 4 numpy arrays `train_x`, `train_y`, `test_x`, and `test_y`. Both `train_x` and `train_y` will be numpy 2d arrays that are obtained from 2b, 2d, 2h, and 2k. `train_y` and `test_y` are 1d numpy arrays associated with the features. The output for each of the following function should be a dictionary with the following keys: 'train-auc', 'train-f1', 'train-f2', 'val-auc', 'val-f1', 'val-f2', 'test-auc', 'test-f1', 'test-f2', and 'params'. Note that train and validation should be determined according to your model selection strategy (i.e., if it is k-fold CV it should be the average) and the test metrics should be calculated using *all* the training data. The 'params' key should contain as the value the dictionary that is passed into a hyper-parameter optimizer within `sklearn.model_selection` such as GridSearchCV. Your dictionary can also include other things that might be useful for the other parts of the questions.

(a) (Code) Write a function `build_logr` that builds an unregularized logistic regression model. Note that the value in 'params' can be an empty dictionary, it just needs to exist in your dictionary.

(b) (Code) Write a function `build_dt` that builds a decision tree accordingly. You might find it useful to base your parameter space on Homework #3 results. Note that the value 'params' should at least contain the parameters that were tested in Homework #3 (i.e., max_depth and min_samples_leaf).

(c) (Code) Write a function `build_rf` that builds a random forest accordingly. You might find it useful to limit your parameter space to consider only the number of estimators, max depth, and minimum number of samples per leaf.

(d) (Code) Write a function `build_svm` that will build a kernel-based SVM. You should only consider linear and polynomial (of a small degree) kernel as RBF will be extremely expensive.

(e) (Code) Write a function `build_nn` that builds a neuarl network accordingly. You might find it useful to base your parameter space on Homework #4 results.

(f) (Written) For each of these 5 models above, build a model for each of the 4 datasets (2b, 2d, 2h, 2k and report the results in a Table (you should have 20 model+dataset combinations across the 3 metrics). Note that for the original dataset, you may find it useful to preprocess it using Standardization or MinMaxScaling.

(g) (Written) Using the results from 3f, compare and contrast the various models with respect to interpretability, computation time, and predictive performance. If you were the loan officer using this model, which one would you use to guide whether or not to give a loan?