Assignment 7

Lab Assignment 7

Breast cancer usually starts from an uncontrolled growth of the cells that make up the milk-producing ducts. While fairly uncommon with men (less than 0.1% experience it), according to BreastCancer.org, one in eight women (12%) end up developing a malignant form of breast cancer over the course of their lifetime. These invasive cells form tumors that destroy nearby tissue, can spread to other parts of the body, and if not duly addressed, may result in death. To put things into perspective, in the U.S., roughly 600 women die per year due to pregnancy related complications... yet over 40,000 die per year due to breast cancer.

Breast cancer doesn't develop over night and, like any other cancer, can be treated extremely effectively if detected in its earlier stages. Part of the understanding cancer is knowing that not all irregular cell growths are malignant; some are benign, or non-dangerous, non-cancerous growths. A benign tumor does not mean the mass doesn't increase in size, but only means it does not pose a threat to nearby tissue, nor is it likely to spread to other parts of the body. The mass simply stays wherever it's growing. Benign tumors are actually pretty popular, such as moles and some warts. Being able to properly assess if a tumor is actually benign and ignorable, or malignant and alarming is therefore of importance, and also is a problem that might be solvable through data and machine learning.

In this lab, you'll be using the Breast Cancer Wisconsin Original data set, provided courtesy of UCI's Machine Learning Repository. A copy of the dataset is located at Module5/Datasets/**breast-cancer-wisconsin.data**. Here are the column names, which you can read more details about on the dataset's information page: `['sample', 'thickness', 'size', 'shape', 'adhesion', 'epithelial', 'nuclei', 'chromatin', 'nucleoli', 'mitoses', 'status']`.

1. Open up the starter code located in Module5/**assignment7.py**, and as usual, read through it entirely.
2. Load up and clean up the dataset, and follow the written directions to split your data, do feature scaling since the features use different units, and then implement PCA and IsoMap so you can test the performance of both, as the technique used to reduce the dimensionality of the dataset down to two variables.
3. Train KNeighborsClassifier on the 2D projected training dataset, the score KNeighborsClassifier on the 2D projected testing dataset.
4. Finally, plot the decision boundary for visual confirmation.

Lab Question 1

1/1 point (graded)

Code up everything as instructed in the assignment. Experiment with various SKLearn preprocessing scaler classes, such as: MaxAbsScaler(), MinMaxScaler(), StandardScaler(), Normalizer(), RobustScaler(), and of course no scaling at all.

Overall, which produced the best result (highest accuracy when scoring against testing data)?

Lab Question 2

1/1 point (graded)

It's important to always keep the objective of the problem you're solving in mind. In this case, your goal is to come up with a way to classify tumor growths as benign or malignant, based off of a handful of features. This is so that a simple test can be administered to see if further action need be taken when a tumor is discovered.

There are two types of errors this classification can make, and they are NOT equal. The first is a false positive. This would be the algorithm errantly classifying a benigh tumor as malignant, which would then prompt doctors to investigate it further, perhaps even schedule a surgery to have it removed. It would be wasteful monetairly and in terms of resources, but not much more than that.

The other type of error would be a false negative. This would be the algorithm incorrectly classifying a dangerious, malignant tumor as benign. If that were to occur, the tumor would be given time to progress into later, more serious stages, and could potentially spread to other parts of the body. A much more dangerious situation to be in.

The KNeighbors classifier in SciKit-Learn gives you the ability to specify weights when initializing the object. By default, these weights are set to 'uniform', so every "K" neighbor has an even vote. It also allows you to specify 'distance', where the votes are scaled inversely porportionally to their distance from the sample being classified (1/d). Lastly, it allows you to specify a user defineable function.

The problem is, the UDF takes in as parameters only a vector of distances and expects an equally sized vector of weights. This doesn't allow you to take advantage of using a different metric on a per class basis to properly weigh your samples to address the undesireability of false negatives over false positives, as it's WAY more important to errantly classify a benign tumor as malignant and have it removed, than to incorrectly leave a malignant tumor, believing it to be benign, and then having the patient progress to full blown in cancer.

One work around for this would be to program your own KNeighbors classifier. Another would be to "bake" the information into your dataset by taking advantage of the fact that KNeighbors is sensitive to the distribution of your variables. For example, randomly reducing the ratio of benign samples compared to malignant samples in your training set.

Between the two provided SciKit-Learn options for weighing, which one performed better on this dataset overall, given the many 'K' permutations you experimented with?