Lab Assignment 2

In Lab Assignment 1, you applied PCA to a dataset generated by 3D-scanning an actual sculpture. Real life 3D objects are a good segue to PCA, since it's **fun** to see its effects on a dataset we can see and touch. Another benefit is that all three spatial dimensions, **x**, **y**, and **z**, each measure the same unit-length relative to one another, so no extra consideration need be made to account for PCA's weakness of requiring feature scaling.

But now the **fun** is over. Gaining some practical experience with real-world datasets, which rarely allot you the luxury of having features all on the same scale, will help you see how critical feature scaling is to PCA. In this lab, you're going to experiment with a subset of UCI's Chronic Kidney Disease data set, a collection of samples taken from patients in India over a two month period, some of whom were in the early stages of the disease. The starter code over at /Module4/**assignment2.py**.

1. Start by looking through the attribute information on the dataset website. Whenever you're given a dataset, the first thing you should do is find out as much about it as possible, both by reading up on any metadata, as well as by prodding through the actual data. Particularly, pay attention to what the docs say about these three variables: **bgr**, **rc**, and **wc**.
2. Load up the **kidney_disease.csv** dataset from the /Module4/Datasets/ directory, and drop **all rows** that have *any* nans. You're probably already a pro at doing that by now. In addition to getting rid of nans, did you know that the `.dropna()` method (upon completion) also automatically re-checks your features and assigns them an appropriate inferred data types?
3. Use an appropriate indexer command to select *only* the following columns: **bgr**, **rc**, and **wc**. Or

Dive Deeper

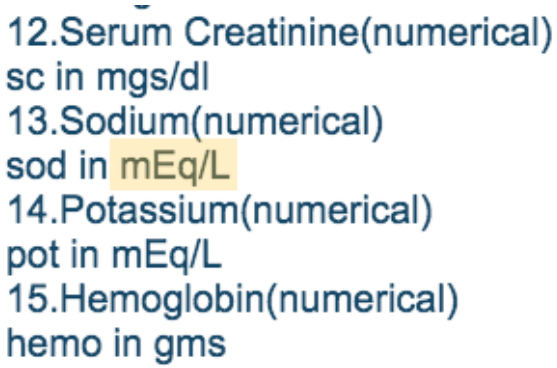
- 5. Data Modeling

alternatively, you can drop every other column, but it's probably easier to just use an indexer to select the one's you wish to keep.

4. Do a check of your dataframe's dtypes. Anything that didn't make it to the right type, you may want to investigate. Look through the data and identify *why* the conversion failed. These types of problems often arise when you aren't in control of how your data is organized. Luckily the issue isn't too bad so once you've identified it, you can fix it through simple numeric coercion.
5. Print the variance of your dataset, as well as a `.describe()` printout.
6. Reduce your dataset to two principal components by run it through PCA, then check out the resulting visualization.

Lab Questions

(5/5 points)



Having reviewed the dataset metadata on its website, what are the units of the **rc**, Red Blood Cell Count feature? An example of where units are defined is shown above. NOTE: In case the UCI site is down, here is a mirror.

- mm/Hg

- mgs/dl
- mEq/L
- cells/cumm ✔
- gms

Why did the .dropna() method fail to convert all of the columns to an appropriate numeric format?

- It actually *did* successfully convert them
- There were a few erroneous leading tab / whitespace characters ✔
- There were a few erroneous trailing tab / whitespace characters
- The dataset comma offset was incorrect in a few rows causing nans to move into the next column

Sort the features below from the **largest** to the **smallest** variance amount.

wc, bgr, rc ✔

As you know, the first thing PCA does is center your dataset about its mean by subtracting the mean from each sample. Looking at the .describe() output of your dataset, particularly the min, max, and mean readings per feature, which feature do you think dominates your X axis? How about the Y axis?

- (●) **wc** dominates the X axis, and **bgr** dominates the Y axis ✔
- ( ) **wc** dominates the X axis, and **rc** dominates the Y axis
- ( ) **rc** dominates the X axis, and **bgr** dominates the Y axis
- ( ) **bgr** dominates the X axis, and **wc** dominates the Y axis

According to your labeling, red plots correspond to chronic kidney disease, and green plots are non-CKD patients. Looking at the scatter plot, are the two classes completely separable, or are there multiple records mixed together?

No, a few records are mixed together ✔

You have used 2 of 2 submissions

You're almost there! The last thing you have to do, and the purpose of this lab really, is to see how feature scaling alters your PCA results.

1. Make a backup of your **assignment2.py** file for safe keeping.
2. Change the line that reads:

```
    scaleFeatures = False
```

   So that is now reads:

```
    scaleFeatures = True
```

3. Also take a look inside of assignment2_helper.py. There are some *important* notes in there about what SKLearn's *transform() methods do, and why they do it. You will need to know this information for future labs!
4. Re-run your assignment and then answer the questions below:

Lab Questions (Continued)

(2/2 points)
Did scaling your features affect their variances at all?

- (●) Yes ✔
- ( ) No

After scaling your features, are the green patients without chronic kidney disease more cleanly separable from the red patients with chronic kidney disease?

- They are less separable
- There isn't much change
- They are more separable ✔

EXPLANATION

To add feature scaling, you're going to need to use SciKit-Learn's StandardScaler: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler

After scaling your features, the variance of all features should be identical. However it should still have varying standard deviations.

You have used 1 of 2 submissions