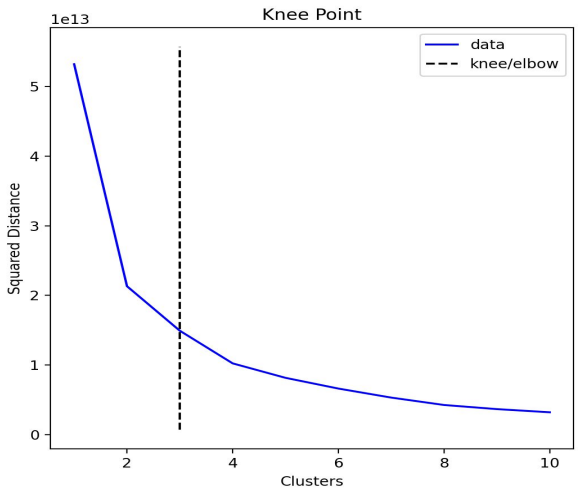
Mini-Project 2
By: Sabrina Margetic

This project is written in django, with backend processing done in python files: Task1.py and Task2.py, and with frontend processing done in files: index.js, index.css, Home.html, MDSScatterPlots.html, PCADataProjection.html, ScatterPlotMatrix.html, and ScreenPlots.html.

Beginning with backend information, Task1.py mostly handles data preprocessing. It gathers the attributes: “Change in Rank,” “Revenue,” “Revenue Change,” “Profit,” “Profit Change,” “Assets,” “Market Value,” “Employees,” “Years on Fortune 500 List,” and “State” from the csv file provided. From there, it removes unnecessary characters and deals with nan values by estimating its value from other features. This is done in the following code snippet:

```
# get rid of NaN data
X = np.array(newData)
imp = IterativeImputer(max_iter=10, random_state=0, sample_posterior=True)
imp.fit(X)
```

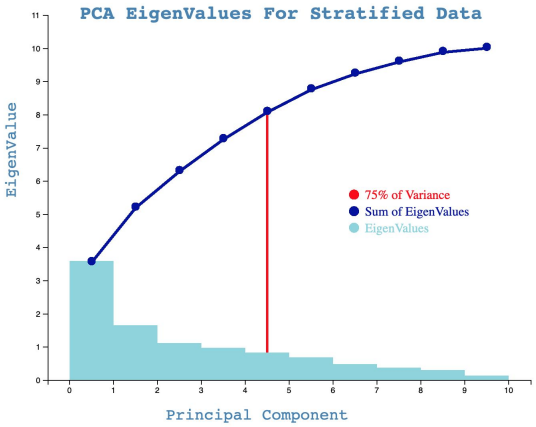
After that, I find out the correct number of clusters through the elbow method. I did this by going through 1-10 number of clusters. I applied the “KMeans()” function in order to appropriately cluster the data based on those numbers. In each cluster, for all data points, I calculated the squared distance between each attribute and the respective attribute of its cluster’s center. Ultimately, I was able to produce the following graph:



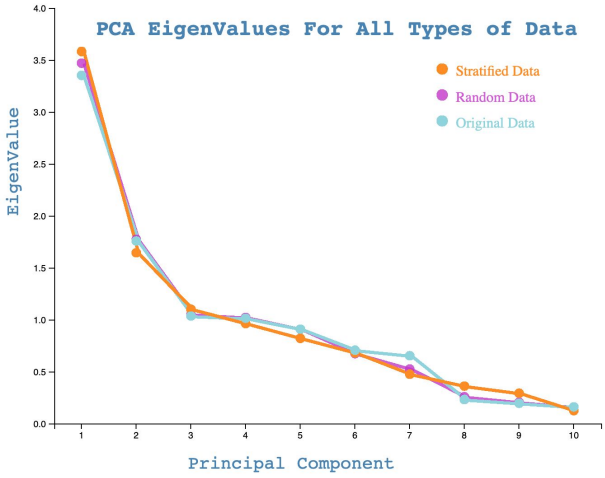
I was able to find the elbow point through the “KneeLocator()” function in python. Based on this number of clusters, I once again used KMeans() and assigned data points to a cluster. I then counted how many data points fit into each cluster and reduced the amount by 75%, to account for my stratified sampling.

When randomly sampling, I simply used a random number generator in order to add data points, until the quota of 25% of my original data was fulfilled.

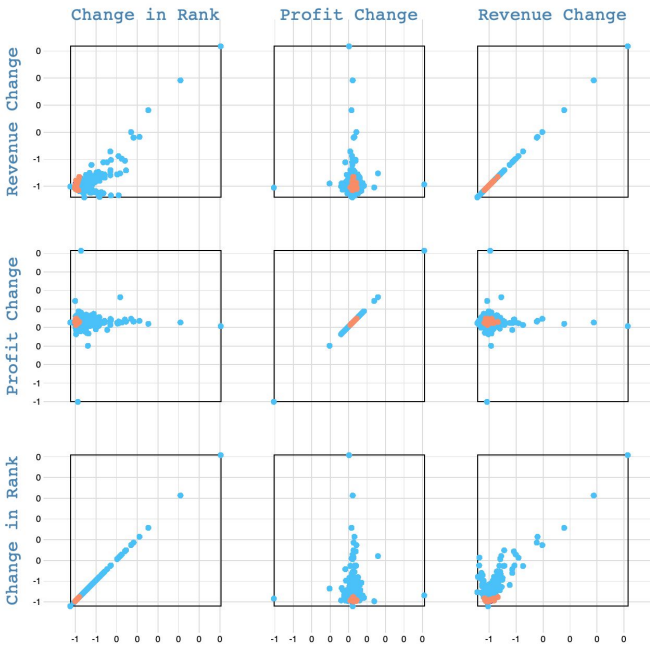
In Task2.py I begin to deal with PCA analysis. The sklearn library of python has a built in PCA function which I used for this part of the assignment. I fit my data onto an instance of the pca function and was able to produce an array of eigenvalues. I used the eigenvalues in order to calculate variance (eigenvalue/sum) for each eigenvalue. The intrinsic dimensionality was calculated to be when 75% of the total variance occurred. An example graph of this is as shown below.



As you can see, the intrinsic dimensionality of this example is 4. Additionally, another graph I chose to produce was the following:



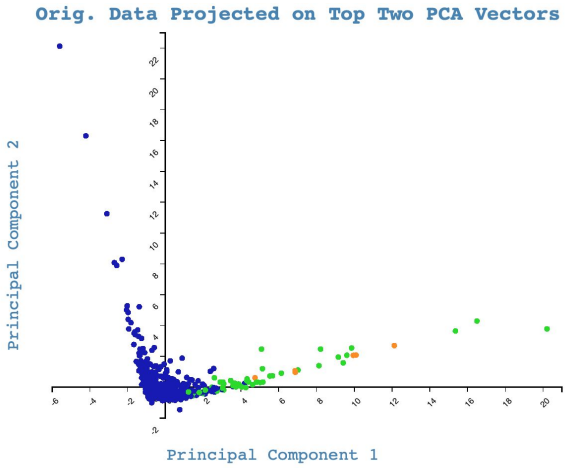
In this, I plot the eigenvalues for all three types of data. As you can see, there is some divergence amongst the eigenvalues, especially as the principal component numbers increase. However, as stated previously, the intrinsic dimensionality is around 4, and for this, the eigenvalues are roughly the same amongst the three types of data sets.



Additionally, based on the intrinsic number of dimensions, I calculated the PCA load factors. Each attribute produced a different load factor. From there, I calculated the three attributes with the highest load factors (by summing their values amongst the intrinsic number of dimensions). I then got the data points associated with these for each type of data set. This is what I used in order to plot the scatter plot matrix. Two examples of such are shown above, the left being from original data, and the right from stratified data.

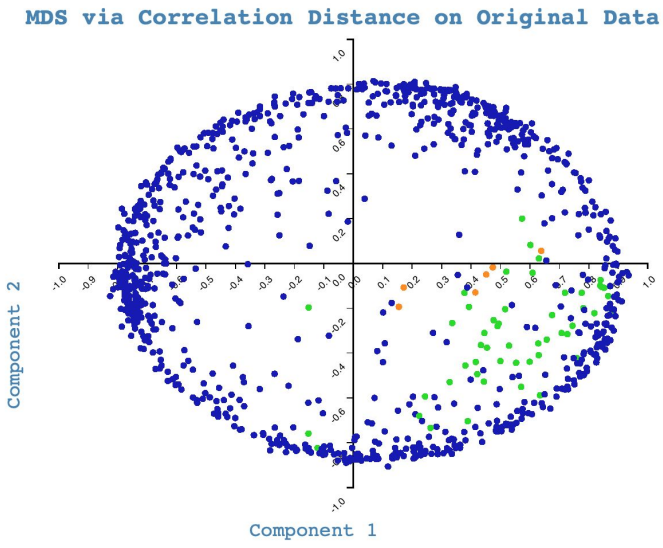
One notable thing about the scatter plot matrix is that for random and stratified data, the attributes used sometimes change. This is due to the fact that the sampling has an element of randomness to it. Based on these data points, an attribute has either more or less significance to a pca component. With this, it is important to recognize that the attributes are rarely the same between random sampling and the original data. This means that our random sampling does not precisely depict the data.

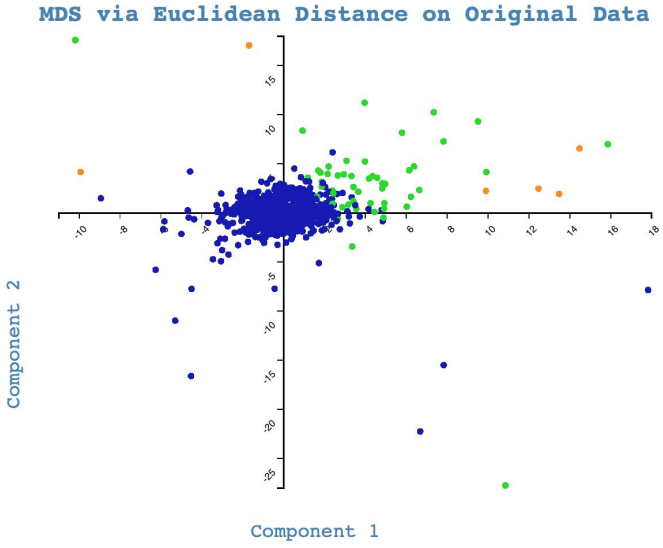
Another important point would be to look at the scatter plot's produced. If we look at the intersection for Change in Rank versus Revenue, we can see that the original data has a y=x trend, with some amount of divergence. Similarly, we can see this trend in our stratified sample, with significantly less data points. Meaning that our clustering and sampling (for stratified data) was performed correctly and we were able to get the same information with far less data.



After this was completed, I used PCA() once again in order to specify the number of dimensions, in this case, two. I fit my three types of data onto this, and then plot them. An example of such is shown to the right.

After this, I move on to obtain the graphs produced through MDS. Similar to PCA(), sklearn has a MDS() function, which I use to reduce my dimensions to two. I use the pairwise_distances() function to either specify if Euclidean or correlation distances are used. Examples of both for original data are shown below.





From these graphs, you can see that the data presented on the PCA graph resembles that of a check mark, the data on the MDS graph via Euclidean distance is elliptically clustered, and the data on the MDS graph via Correlation resembles an ellipsoid. Since these types of analysis try to reduce dimensions and convey the same amount of information, it seems favorable to have a wider spread with less clustering, less overlap. Therefore, in my analysis, it seems best to use MDS analysis via Correlation Distance.