PCA Implementation on Stroke Classification

COGS118B Final Project

Steve Kuk, Kevin Pham,Garvin Mo Zhen,
Brendan Wong, Rifu Chen

Spring 2021

1 Introduction and Motivation:

The goal of our project is to compare data that has and has not been processed with PCA. We will compare pre and post PCA data performance with multiple supervised learning algorithms. Additionally, we want to reduce the number of features from our original dataset, and observe if our reduced number of features can yield a similar or better performance. We will be using the Stroke Prediction Dataset from Kaggle as the dataset of choice and our algorithms will classify each row as a stroke or no-stroke. Each algorithm will be compared in both runtime and accuracy on both normal data and PCA data. From this, we will be able to garner an understanding of what algorithms and scenarios in which PCA can have a significant enough impact on to implement. Not only that, we want to see if we are able to accurately predict strokes using a reduced number of features.

2 Related work:

In order to perform our supervised learning methods, we referred to the paper "An Empirical Comparison of Supervised Learning Algorithms" by Rich Caruana and Alexandru Niculescu-Mizil. In this paper, the two researchers compared multiple supervised learning algorithms across multiple metrics such as accuracy, f1 score, and Lift and ranked them by performance across means. Similarly, we utilized this idea to run the supervised learning algorithms on the data pre-PCA and post-PCA and used the accuracy metric to judge which method was more accurate.

In a similar literature paper by Annie George, titled "Anomaly Detection based on Machine Learning: Dimensionality Reduction using PCA and Classification using SVM", we see a similar approach in their attempt to use PCA on a supervised machine learning algorithm. The paper focused on a classification problem (anomaly detection) using pre and post-PCA through SVM on the KDD99 dataset. The conclusion of this paper was that PCA for dimensionality reduction ultimately reduced the execution time and boosted metric scores for SVM. This is similar to our approach to use pre and post-PCA on different supervised machine learning algorithms (including SVM) to compare its performance through accuracy and execution time.

3. Methods:

In order to perform algorithms on our dataset, we had to first clean up our data. We loaded the data in and after checking for null values and looking over potential negative cases, we dropped the null rows. We then separated the target column, strokes, into a separate column and dropped the original column and the id from the dataset. Afterwards, we one scaled our data and one hot encoded them and then put them into a new dataframe called scaled_columns.

We then began performing a manual Principal Component Analysis. We first wrote a function that would sort the eigenvalues and eigenvectors from largest to smallest by referencing previous homeworks. We set up the target column and stored it as a variable and then began calculating the covariance matrix of the scale data. We did so by calculating the mean, converting it to an array, subtracting the mean to get the centered data and then using that to get the covariance matrix. Using the covariance matrix, we were able to perform eigendecomposition and acquire the sorted eigenvectors and eigenvalues. We removed the complex parts for both and then checked the variances to see which components had the highest variance. We used a barchart to graph the PCA components by highest variance percentage to lowest and then chose our two highest components to work with. We assigned the two components and our target variable and then used that transformed data to plot it in one dimension and two dimensional space. Lastly, we created a heatmap for the first two components to demonstrate the correspondence of the components with the features of the data.

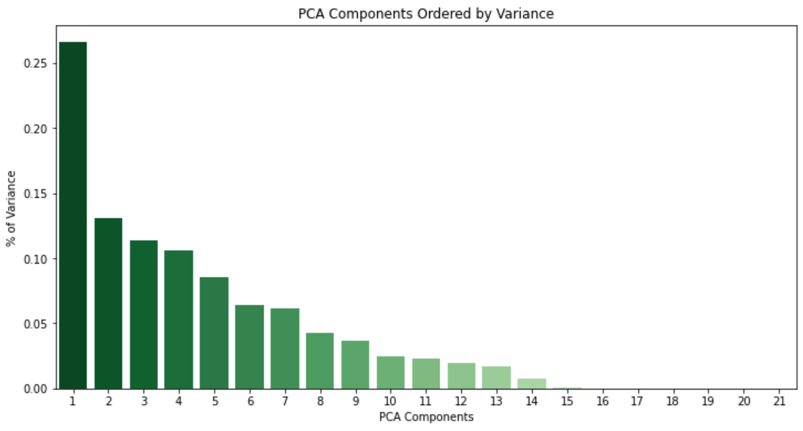
We also performed a Principal Component Analysis Algorithm using a premade package through sklearn. Here we first took all 21 components of the data and fitted it and then transformed the data. Following this, we turned the transformed data into a dataframe and then plotted the data in two dimensional spaces. We created two heat maps: the first heat map focused on the first two components, and the second heat map focused on all the components.

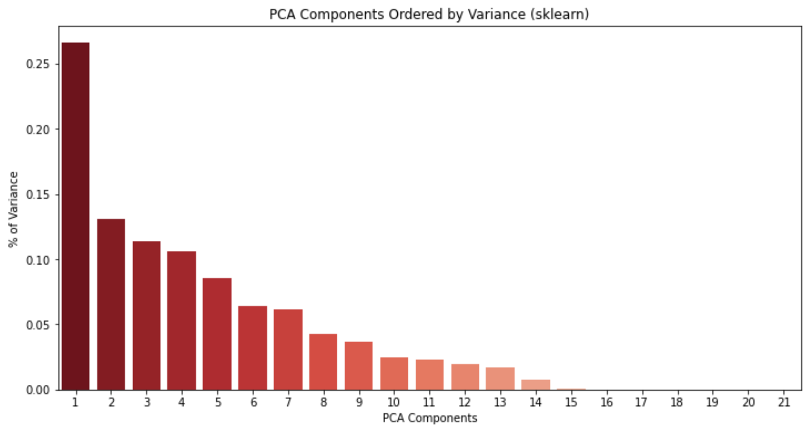
With both the manual and sklearn implemented PCA, we compared the results to make sure PCA was performing correctly. By comparing the PCA components ordered by variance from sklearn and our manual implementation, we see very similar results. We also compared the plots of PCA in 2-D space and they also seemed to be the same. Finally, we compared the heat maps, which also turned out to be the same. From this, we concluded that our manual implementation of PCA was working correctly.

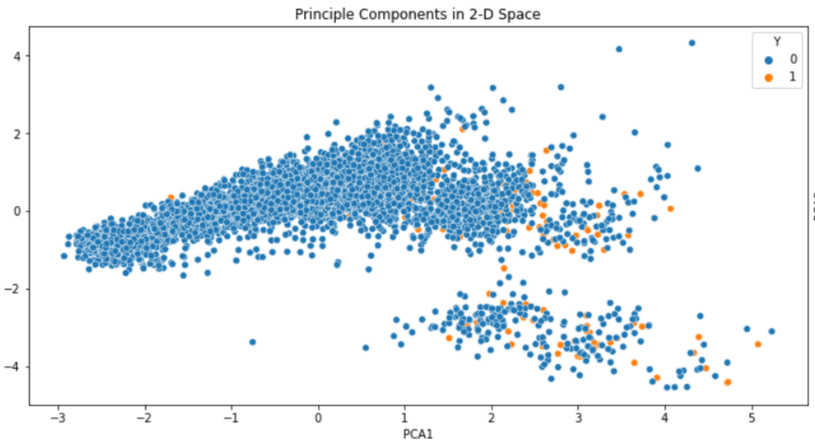
The next step is to compare our classification results. We will compare pre-PCA, post-PCA with 2 principal components, and post-PCA with 9 principal components using supervised machine learning algorithms. To implement these algorithms, we used the sklearn library. We first created a dataframe with the first nine principal components which accounted for about 90% of the variance in our data, meanwhile the first two principal components accounted for about 40% of the variance. We then separated the dataset into training on the raw data, 2 principal components, and 9 principal components. Using train_test_split, we split the data into the train and testing sets with test_size equal to 0.35. To run these supervised algorithms, we used the training data to fit the algorithm and then used the testing set to predict its classification. We ran Logistic Regression under the 3 different training sets (pre-PCA, post-PCA 2 components, and post-PCA 9 components) then noted the accuracy values and time it took for the algorithm to run. Similarly, we then ran Decision Trees under the 3 different training sets and noted the metrics. Finally, for our last supervised algorithm, we ran linear SVM under the 3 training sets and also noted its metrics. From these steps, we obtained the pre and post-PCA (2 and 9 principal components) classification results from the supervised algorithms and can compare the times and accuracy. To simplify our analysis, we created multiple dataframes with the recorded metrics and their respected supervised algorithms for comparison.

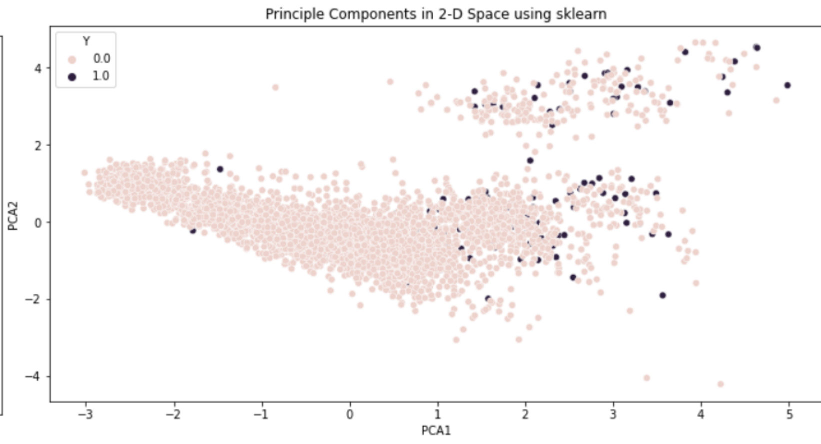
4 Results:

We compared our manual PCA with our sklearn PCA and were pleased to find that there was barely a distinction between the two. We examined the two-dimensional scatterplot and found both plots to be similar, proving that our manual code worked well with the data. In addition, when checking for which components had the highest variance to determine which components we would use, we found that both plots were identical, adding to the validity of our manual PCA. The following images show the manual PCA components on the left and the sklearn PCA components on the right along with the two dimensional plot for manual on the left and the other on the right.

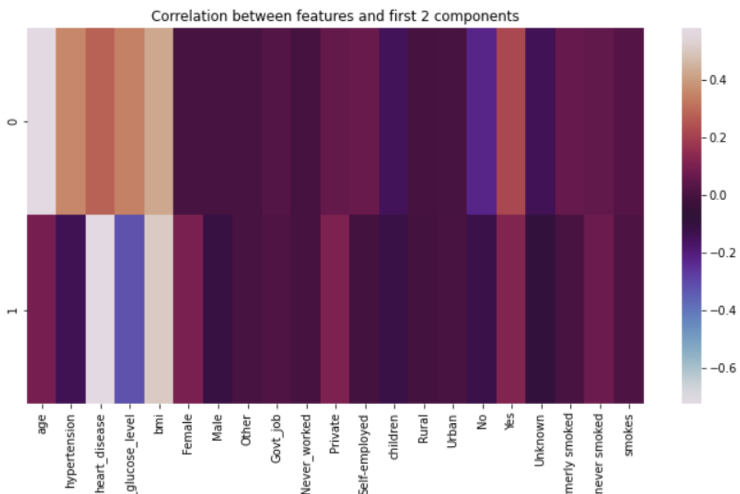


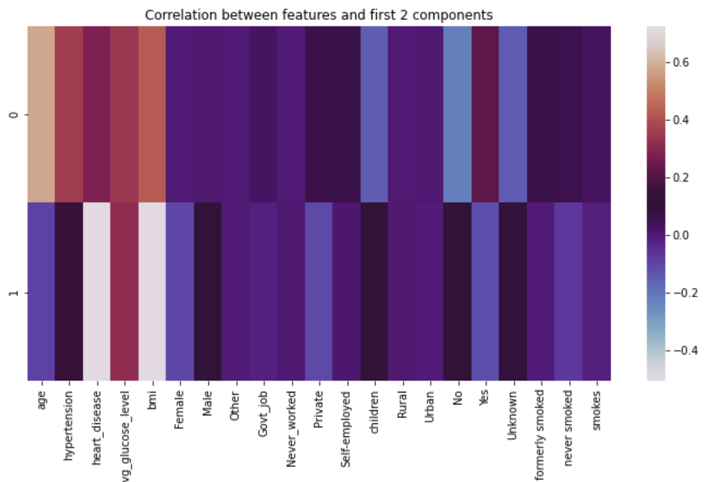




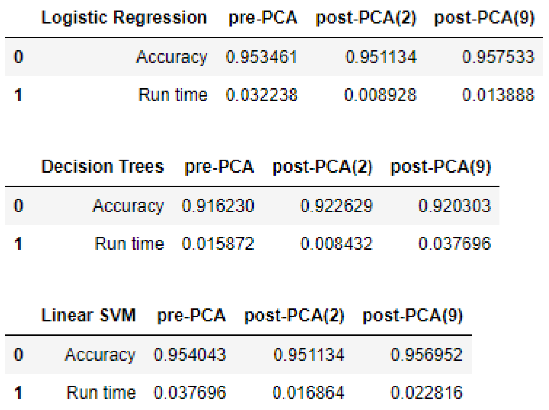


In addition to the plots above, we also carried out heat maps to see the correlation between the features within the components and the likelihood of a stroke. Both heat maps gave similar results except the scale for the manual PCA was from -0.6 to 0.4 whereas the sklearn PCA was -0.4 to 0.6. The heat maps below show the manual PCA to the left and the sklearn to the right.





We discovered that using our dimensionally reduced data to perform classification, resulted in a similar and better performance. Overall, it did have a positive effect on both runtime and accuracy. We can see that, in all three supervised learning algorithms that we tested on, our PCA data was able to outperform the original dataset slightly. Post-PCA using the first 9 components was able to outperform the original dataset in Logistic Regression and SVM learning algorithm. On the other hand, post-PCA using the first 2 components performed the best in the Decision Trees algorithm. Overall, reducing the features to either 2 or 9 components gave us a noticeable reduction in training time. Results for classification can be seen in the table on the right.



	Logistic Regression	pre-PCA	post-PCA(2)	post-PCA(9)
0	Accuracy	0.953461	0.951134	0.957533
1	Run time	0.032238	0.008928	0.013888

	Decision Trees	pre-PCA	post-PCA(2)	post-PCA(9)
0	Accuracy	0.916230	0.922629	0.920303
1	Run time	0.015872	0.008432	0.037696

	Linear SVM	pre-PCA	post-PCA(2)	post-PCA(9)
0	Accuracy	0.954043	0.951134	0.956952
1	Run time	0.037696	0.016864	0.022816

However, we only trained and performed classification on each set of data for just one iteration. If we were to rerun our notebook, we would get similar results but sometimes the other datasets can outperform the other slightly. It was interesting that in some cases, post-PCA with 9 components underperformed post-PCA with 2 components because the first 2 components only accounted for 40% of the variance in our data.

5 Discussion:

From our results, we learned that by applying PCA, we are able to reduce our dataset to a minimum of 2 features. With our reduced dataset we were able to achieve similar and sometimes better accuracy, and also greatly decrease the amount of fitting time in all algorithms used. If possible, we think that PCA should be implemented in any scenario where the resources permit, as there were only upsides.

A possible extension to our project would be to test with more datasets that contain more features. In terms of big data, the Stroke Prediction Dataset was relatively small, at 5110 rows and around 4900 after cleansing, and only had about 21 features after being processed. If we were to retest our conclusion more extensively, greater variation in size of the datasets would be useful, as it would give us a more complete picture of how PCA performs when datasets grow larger. We can also test out how many features we can reduce in datasets with a large number of features. In addition, restructuring the methodology so that comparison times began at the beginning of data processing all the way through PCA and fitting, would give us a bigger picture into when implementing PCA could be more effective in terms of effort and time.

Another extension could also be to include unsupervised clustering algorithms. This project, we limited comparing the results of pre and post-PCA on supervised machine learning algorithms. Including unsupervised algorithms in our project would work better because we would have a larger pool to compare results with and it also allows us to learn about the effects of PCA on unsupervised algorithms.

Not only that, for our classification part of our project we only perform classification on the default parameters and only on one iteration. As a result, we are not too sure if the post-PCA was outperforming the raw dataset by chance or whether it was consistent. We also trained the dataset on the default parameters, and did not perform any sort of hypertuning to the parameters.

I think in the near future, we can implement a better way to compare the classification results by performing more iterations and hypertuning the parameters to get the best performing models.

We think most of our projects worked perfectly. The only thing that we had a major problem with was manually implementing PCA. Next time if we had more time, we could definitely look into PCA more and understand the errors we were having. We could do a deeper dive of research into PCA implementation.

6 Contributions:

Steve: Related Work, Methods, Results, Discussion, Slides
Brendan: Introduction, Motivation, Results, Discussion, Slides
Garvin: Methods, Results, Code Implementation, Slides
Kevin: Related Works, Code Implementation, Methods, Results, Slides
Rifu: Code Implementation, Slides

7 Code:

Source code can be found here: https://github.com/garvingit/COGS118B_FINAL

8 References:

R. Caruana and A. Niculescu-Mizil. "An Empirical Comparison of Supervised Learning Algorithms." In Proceedings of the 23rd International conference on Machine Learning. 2006. https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf

A. George. "Anomaly Detecting based on Machine Learning: Dimensionality Reduction using PCA and Classification using SVM." International Journal of Computer Applications (0975 – 8887) Volume 47– No.21, 2012. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.6954&rep=rep1&type=pdf

Choudhary, A. (2019, September 29). Principal Component Analysis (PCA) with Python. DataScience+. https://datascienceplus.com/principal-component-analysis-pca-with-python

Jaadi, Z. (n.d.). *A Step-by-Step Explanation of Principal Component Analysis (PCA)*. Built In. https://builtin.com/data-science/step-step-explanation-principal-component-analysis

What are good metrics to assess the quality of a PCA fit, in order to select the number of components? https://stats.stackexchange.com/questions/100143/what-are-good-metrics-to-assess-the-quality-of-a-pca-fit-in-order-to-select-the

Brownlee, J. (2019, August 9). *How to Calculate Principal Component Analysis (PCA) from Scratch in Python*. Machine Learning Mastery. https://machinelearningmastery.com/calculate-principal-component-analysis-scratch-python/