**CS550\_Final\_Project\_#\_#**

**TEAM MEMBERS**

**Shraddha Gajjar(15992)**

**Hardik Mistry(16100)**

**Project Definition**

This project is focused on using R language to calculate a medical scoring system based on some given data. This technique is currently used in few disciplines of medicine including baby’s health determination within 5 minutes of birth (Apgar score) or a similar measure to Apgar score for determination of health of patience in ICU after surgery. The goal is to generate a health score as a multivariate linear combination of multiple factors. These factors are health metrics that determine the health level of an individual in that category. For example one factor could be the blood pressure of the subject. If the blood pressure is within the normal range then the score will have a value of 2, if the blood pressure is above or below the normal range but still within an acceptable range the score for blood pressure will be 1 and if it is above or below the border line. The blood pressure score will be equal to zero if it is out of normal range. Each parameter or score belongs to a vital sign category. Each score also has a factor or a multiplier that shows the degree of the contribution of that parameter in the health score.

There are two datasets given for this problem. Two matrices, of size 51 by 1000 elements are given in a zip file. The first 50 rows shows the actual scores generated based on some criterion and last row (row 51) shows the health score corresponding to each column of the data (50 scores) so the size of health score corresponds to a vector of 1x1000.

You have to use the correct machine learning technique to figure out the following:

1. Factors for all 50 scores based on collected data.
2. If you find that some of these score/parameters are correlated then there is a potential to combine them into a common parameter. (Hint: since you have both the output, i.e. health score and the each individual score you probably have to use partial least squares rather than PCA).
3. Try your algorithm with the second data set. What is the difference between results for the two data sets?
4. You need to write the R code for doing this project and also a report explaining how you used the data and what techniques you used and why these methods are suitable for this kind of data analysis.

**Theory :**

PCA is a dimensionality reduction method in which a covariance analysis between factors takes place. The original data is remapped into a new coordinate system based on the variance within the data. PCA applies a mathematical procedure for transforming a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

PCA is useful when there is data on a large number of variables, and (possibly) there is some redundancy in those variables. In this case, redundancy means that some of the variables are correlated with one another. And because of this redundancy, PCA can be used to reduce the observed variables into a smaller number of principal components that will account for most of the variance in the observed variables.

PCA is recommended as an exploratory tool to uncover unknown trends in the data. The technique has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension.

The PCA algorithm consists of 5 main steps:

1. Subtract the mean: subtract the mean from each of the data dimensions. The mean subtracted is the average across each dimension. This produces a data set whose mean is zero.
2. Calculate the covariance matrix:

**C^{n \times n}=(c_{i,j},c_{i,j} = cov(Dim_{i},Dim_{j}))**

where C^{n \times n} is a matrix which each entry is the result of calculating the covariance between two separate dimensions.

1. Calculate the eigenvectors and Eigen values of the covariance matrix.
2. Choose components and form a feature vector: once eigenvectors are found from the covariance matrix, the next step is to order them by Eigen value, highest to lowest. So that the components are sorted in order of significance. The number of

eigenvectors that you choose will be the number of dimensions of the new data set. The objective of this step is construct a feature vector (matrix of vectors). From the list of eigenvectors take the eigenvectors selected and form a matrix with them in the columns:

**Feature Vector = (eig\_1, eig\_2, ..., eig\_n)**

1. Derive the new data set. Take the transpose of the Feature Vector and multiply it on the left of the original data set, transposed:

**FinalData = RowFeatureVector x Row Data Adjusted**

where RowFeatureVector is the matrix with the eigenvectors in the columns transposed (the eigenvectors are now in the rows and the most significant are in the top) and Row Data Adjusted is the mean-adjusted data transposed (the data items are in each column, with each row holding a separate dimension).

We choose princomp method from stats package for this tutorial.

* R package: **stats**
* Method: **princomp**
* Documentation: [princomp](http://stat.ethz.ch/R-manual/R-patched/library/stats/html/princomp.html" \t "_blank)

**princomp** is a generic method with "**formula**" and "**default**" methods from stats package which performs a principal components analysis on the given numeric data matrix and returns the results as an object of class **princomp**.

**Principal component analysis** (**PCA**) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called **principal components**. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is sensitive to the relative scaling of the original variables.

PCA is mostly used as a tool in exploratory data analysis and for making predictive models. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering (and normalizing or using Z-scores) the data matrix for each attribute. The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a particular data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score).

PCA is the simplest of the true eigenvector-based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualized as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its (in some sense; see below) most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced.

PCA is closely related to factor analysis. Factor analysis typically incorporates more domain specific assumptions about the underlying structure and solves eigenvectors of a slightly different matrix.

**Solution :**

From the given data it was analyzed that data of last row 51 shows the health score based on the all 50 rows scores. The data of the last row is the response of all the corresponding data of all other rows. This is somewhat like one block of response Y in terms of another block of predicted variables X. The behavior of this type of quantitative response in terms of set of quantitative predictors X could be model using linear model shown below:

E(y) = x1b1 + x2b2 + x3b3 + … + xpbp

The main goal is to find all coefficients bj (j = 1, 2... p), so that it could help to predict Y values by combining with predicted X variables. This model could be solve using Partial Least Square Regression (PLS-R).

**R-CODE:**

library(plsdepot)

predictors = read.table("C:\\Users\\Beastfury\\Desktop\\HardikMistry\_16100\_Finalproject\\hScore.dat", nrows = 50)

response = read.table("C:\\Users\\Beastfury\\Desktop\\HardikMistry\_16100\_Finalproject\\hScore.dat", nrows = 1, skip = 50)

predictors\_final = t(predictors)

response\_final = t(response)

predictors\_final

regcoefs\_final

stdcoefs\_final

predictors

response

plsreg\_final = plsreg1(predictors\_final,response\_final, comps = 2, crosval = FALSE)

plot(plsreg\_final)

stdcoefs\_final = t(plsreg\_final$std.coefs)

regcoefs\_final = t(plsreg\_final$reg.coefs)

R2\_final = t(plsreg\_final$R2)

final\_predictor <- factor(predictors\_final)

final\_levels = t(levels(final\_predictor))

#\_\_\_\_\_\_\_\_Analyzing second data set\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

predictors = read.table("C:\\Users\\Beastfury\\Desktop\\HardikMistry\_16100\_Finalproject\\hScore1.dat", nrows = 50)

response = read.table("C:\\Users\\Beastfury\\Desktop\\HardikMistry\_16100\_Finalproject\\hScore1.dat", nrows = 1, skip = 50)

predictors\_final = t(predictors)

response\_final = t(response)

response\_match(response$code)

final\_levels(mylevels)

predictors\_final

R2\_final

regcoefs\_final

stdcoefs\_final

predictors

response

plsreg\_final = plsreg1(predictors\_final, response\_final, comps = 2, crosval = FALSE)

plot(plsreg\_final)

stdcoefs\_final = t(plsreg\_final$std.coefs)

regcoefs\_final = t(plsreg\_final$reg.coefs)

R2\_final = t(plsreg\_final$R2)

final\_predictor <- factor(predictors\_final)

final\_levels = t(levels(final\_predictor))













