Study on Data Reduction Techniques using Principal Component Analysis

Research Practice
Submitted in partial fulfillment of requirements of
BITS G540 Research Practice

By
Madhusmita Oke
ID. 2018H1030194G

Under the supervision of
Dr. Sanjay K. Sahay
Assistant Professor

Department of Computer Science and Information Systems

April 2019

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
K K BIRLA GOA CAMPUS

CERTIFICATE

This is to certify that the Research Practice entitled, '**Study on Data Reduction Techniques using Principal Component Analysis**' and submitted by **Madhusmita Oke**, ID No. **2018H1030194G**, in partial fulfilment of the requirement of BITS G540, Research Practice embodies the work done by her under my supervision.

Date:

Place:

(Signature of Mentor)
Dr. Sanjay K. Sahay
Assistant Professor
Department of CS & IS
BITS Pilani K K Birla Goa Campus

ACKNOWLEDGEMENT

This research study would not have been successful without the help and support of **Dr. Sanjay K. Sahay** , who offered invaluable guidance and assistance to me to fully understand and finish this study. The frequent suggestions and reviews provided by him were critical for the fulfillment of this work. He has imparted sufficient knowledge for me to learn.

I would also like to extend my gratitude to **Dr. Ashwin Srinivasan** and **ARD Division**, BITS Pilani KK Birla Goa Campus , for providing this opportunity and all the facilities required to complete this study.

ABSTRACT

Astronomical researches involve working with data that is high dimensional. It has been observed that over the years the data available for analysis would be of the order of 60 PB. The task of analysing such an enormous amount of data is quite challenging. In such a scenario, a virtual observatory must provide meaningful data to astronomers for analysis instead of raw data. This step of mining data into a meaningful data that can be transformed to original form or used in the same form, leads to the motivation for the study of data reduction techniques, namely, Principal Component Analysis. This study aims at studying data reduction technique Principal Component Analysis that is used in fields mentioned above.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The increasing amount of data available for analysis in research fields like Astronomy, Image processing, Biomedical studies there is need for mechanism to reduce the amount of data to a meaningful form that can provide insights and be used for further analysis. Principal Component Analysis is a data reduction technique that is applied on quantitative data which can be used in these fields. Data reduction is a crucial step in data mining since it can reduce computation costs and make data at hand meaningful to use for further analysis.

When the attributes in data (columns/dimensions) are correlated there is information that is redundant which increases the dataset's noise. This redundant information impacts negatively in terms of performance and computation. As a result, dimensionality reduction methods have paramount importance. It is a useful in reducing complexity and avoid overfitting. In a nutshell, Principal Component Analysis (PCA) is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, Principal Component Analysis is a powerful tool for analysing data.

Consider the examples of image processing in this regard.An image consists of pixels , in a square grid of n cells we have data contained in n^2 positions. For an image with good quality, we have huge data along with specific dimensional (attributes) information for each cell. In order to reduce the computation costs for processing the image, image compression applies PCA. This reduces the cost of further processing and simplifies the data. In addition, the compressed image preserves most characteristics of the original image. The original image can be reconstructed using the compressed form reducing computational cost to a great extent.

CHAPTER 2

METHODOLOGY

1. Study and Review on Principal Component Analysis
2. Application of Principal Component Analysis on 4 column data
3. Reconstruct the reduced data with minimum possible error
4. Generalization of Principal Component Analysis for n columns
5. Reconstruction of the reduced dataset with reasonable accuracy

CHAPTER 3

STUDY ON PRINCIPAL COMPONENT ANALYSIS

Principal component analysis (PCA) is a method to reduce the dimensions of the dataset of possibly correlated variables into a smaller number of uncorrelated variables. The new data so obtained is smaller in space and still preserves the characteristics of original data. Principal Components are directions in data that maximize variance ,that is, minimize information loss. The principal component conveys higher information when it represents maximum variance possible. As the data is present is compressed form, it becomes easier to use this data for data visualization in addition to the advantage of reduced cost of computation. The steps involved in PCA and significance of each step is elaborated as follows:

Steps in Principal Component Analysis:

1. **Mean Adjustment:**
   Mean is subtracted from each of the dimensions , that is, X becomes X- X_{mean}. This step is essential to center the data values. This produces a dataset whose mean is zero.

2. **Calculate Covariance:**
   Covariance matrix is calculated for each of x-y values. Covariance helps in understanding the similarity or dissimilarity between the data points. If it's positive, both values increase simultaneously. If it is negative , one of the variables decreases as the other increases. Covariance zero signifies that both x and y are independent of each other. It can be calculated as:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

3. **Calculate Eigen vectors and Eigen values:**
   Covariance matrix is a square matrix, one can find eigen vectors and eigen values for this matrix. The eigen vectors should have length as one. An important characteristic of eigen vectors is that they are orthogonal to each other. That means, we have computed representative dimensions which are independent to each other. This step helps us find patterns in dataset after plotting them alongside the original values.

4. **Choose Components and form feature vectors:**
   On careful look at the eigen vectors /values, we can notice that one of the vectors best represents majority of dataset called the principal component. In this step, we order the eigen values from highest to lowest , that is, the order of significance. We can now ignore insignificant eigen values. This results into some loss of data but since the eigen value is small the loss is not considerable. Now feature vector is constructed by taking the eigenvectors that one wants to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns. Given as:

$$FeatureVector = (eig_1\ eig_2\ eig_3\ ....\ eig_n)$$

5. **Derive new data:**
   Now we can derive final data using the selected feature vector by simply taking transpose of the vector and multiplying it on the left of the original data set,transposed. Given as:

$$FinalData = RowFeatureVector \times RowDataAdjust$$

   Here, RowFeatureVector is the matrix with the eigen vectors in the columns transposed so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and RowDataAdjust is mean adjusted data transposed, that is, the data items are in each column, with each row holding a separate dimension.

6. **Get the original data back:**

   It is crucial to be able to get the original data back while using PCA. Since few eigen vectors are dropped one cannot get the exact original data. But as many components are taken the accuracy goes on increasing with it. The formula can be given as:

$$RowDataAdjust = RowFeatureVector^T \times FinalData$$

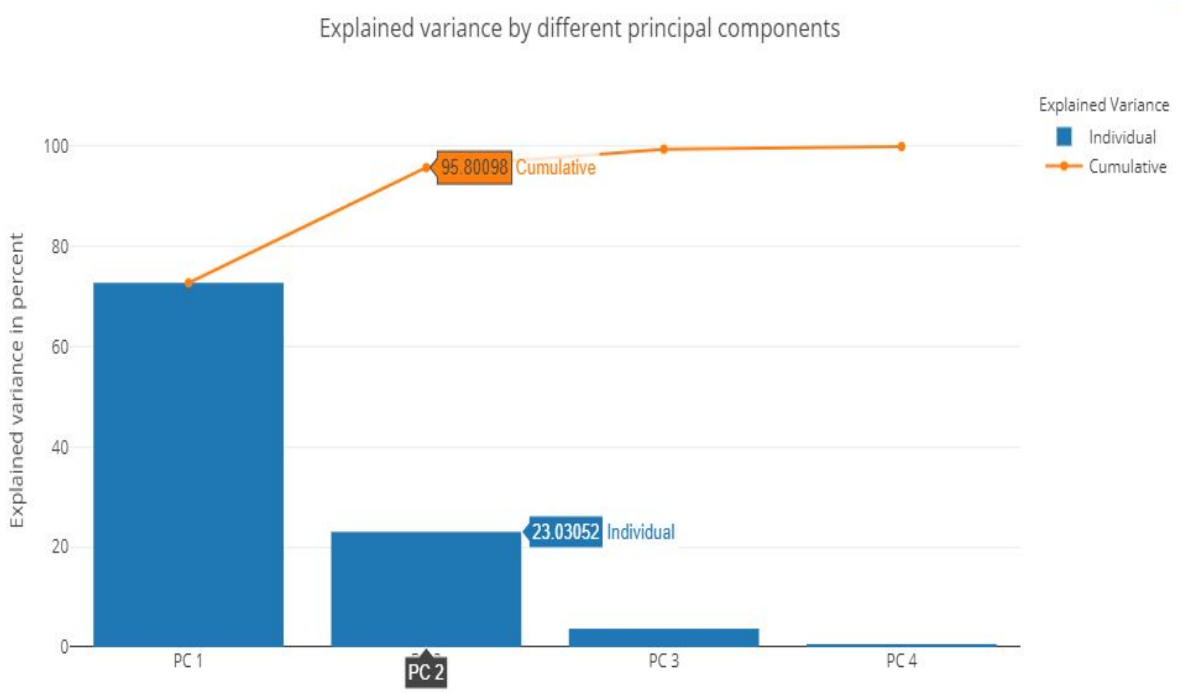
   Mean can be added to the above result in order to get the original data. There will be certain losses but if the major dimensions are appropriately represented by the principal components the accuracy is reasonable.

CHAPTER 4

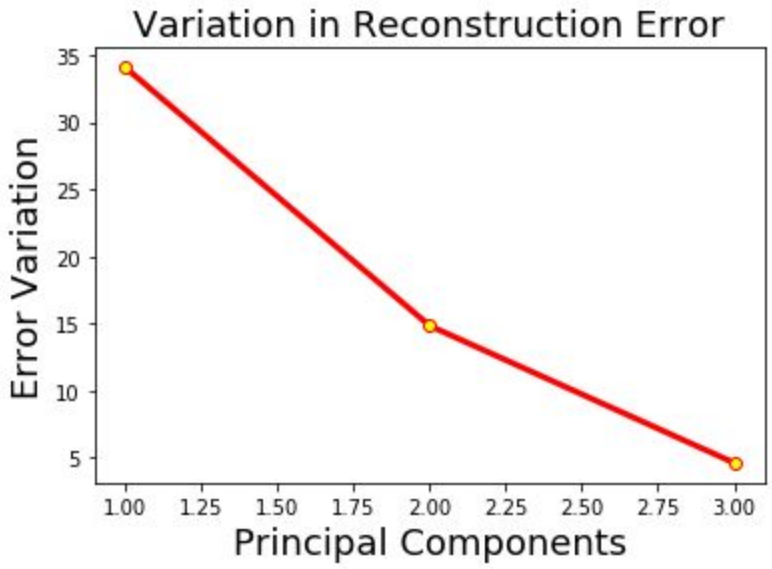
EXPERIMENTAL ANALYSIS AND RESULTS

1. Principal Component Analysis on 4 column data
   Dataset: Iris Dataset
   150 instances , 4 attributes reduced to 2 components

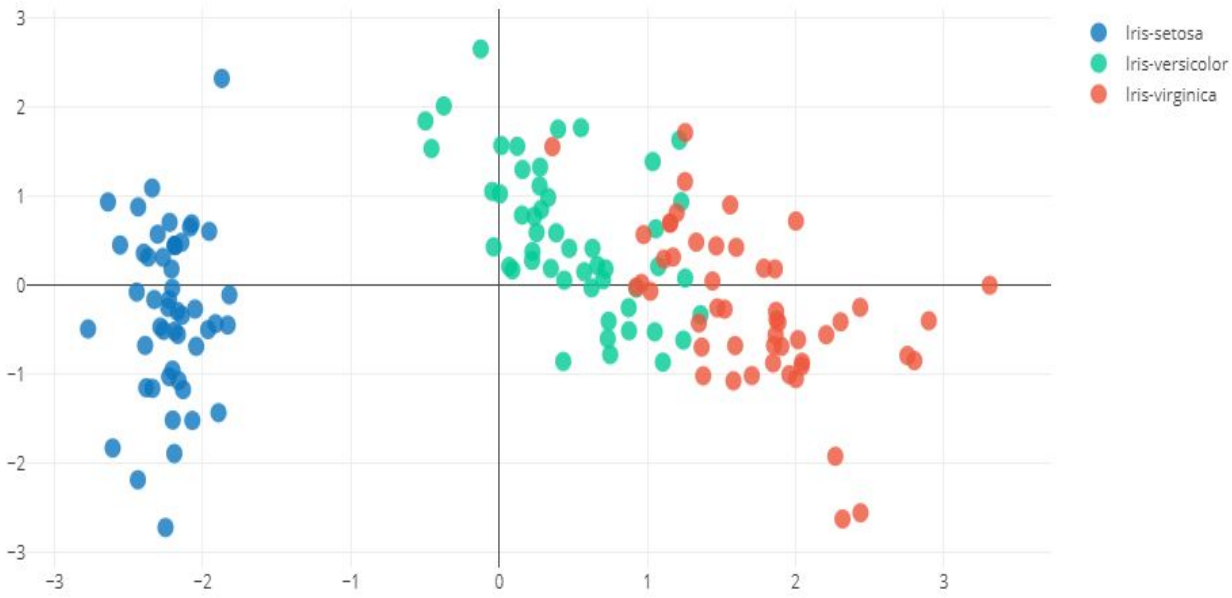
   Explained variance by different principal components



Reconstruction Error Variation



Visualization of Reconstructed Data

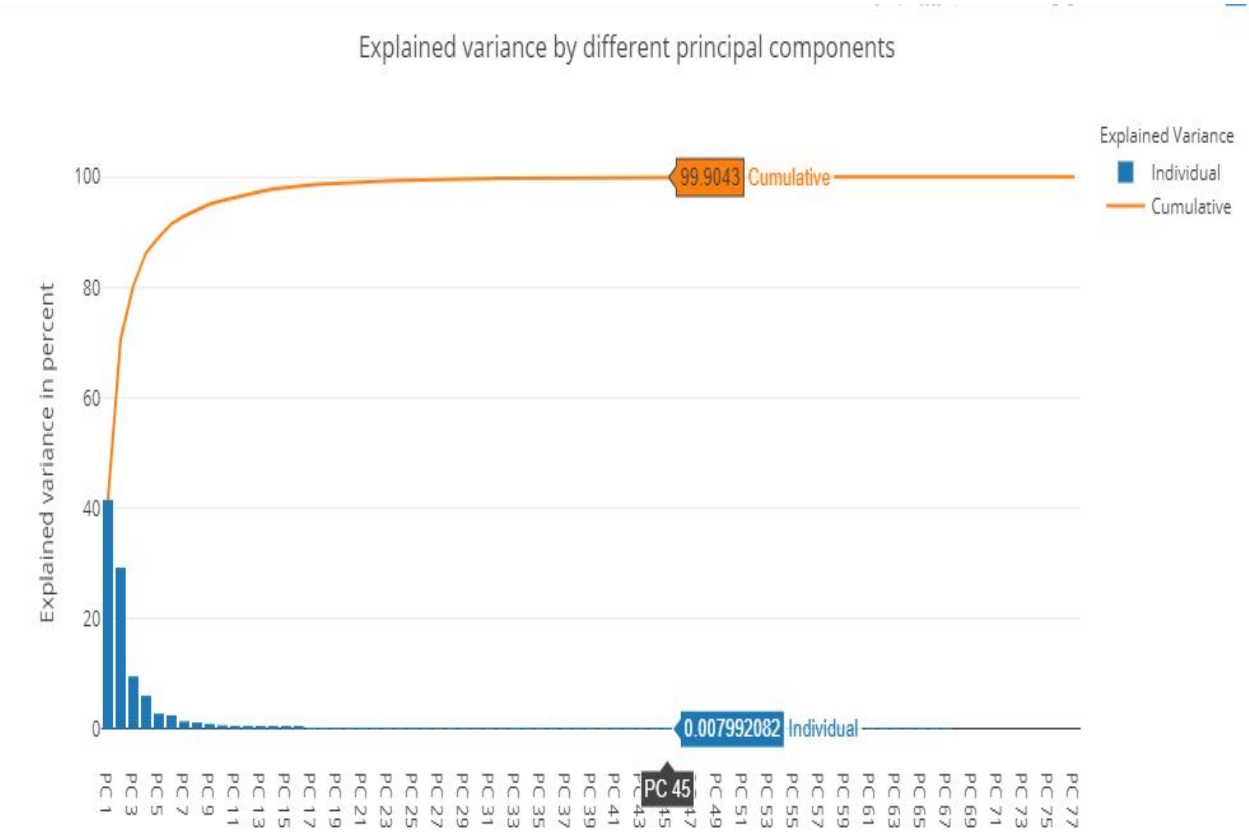


2. Principal Component Analysis on data with more attributes:

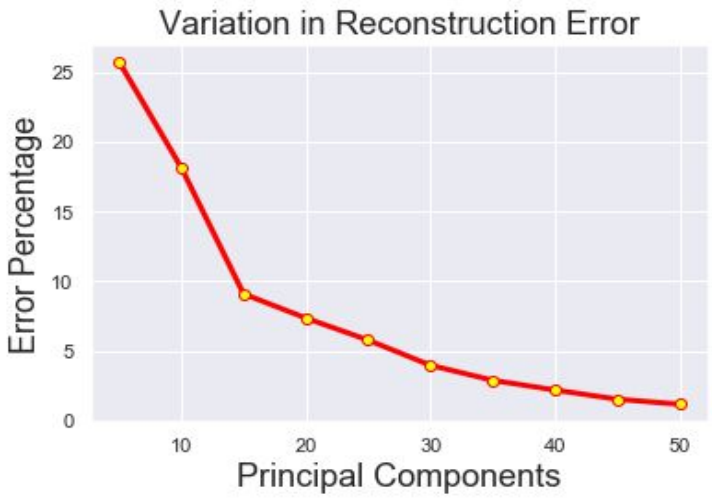
Dataset: Mice Protein Expression Dataset

1080 instances , 82 attributes reduced to 45 components

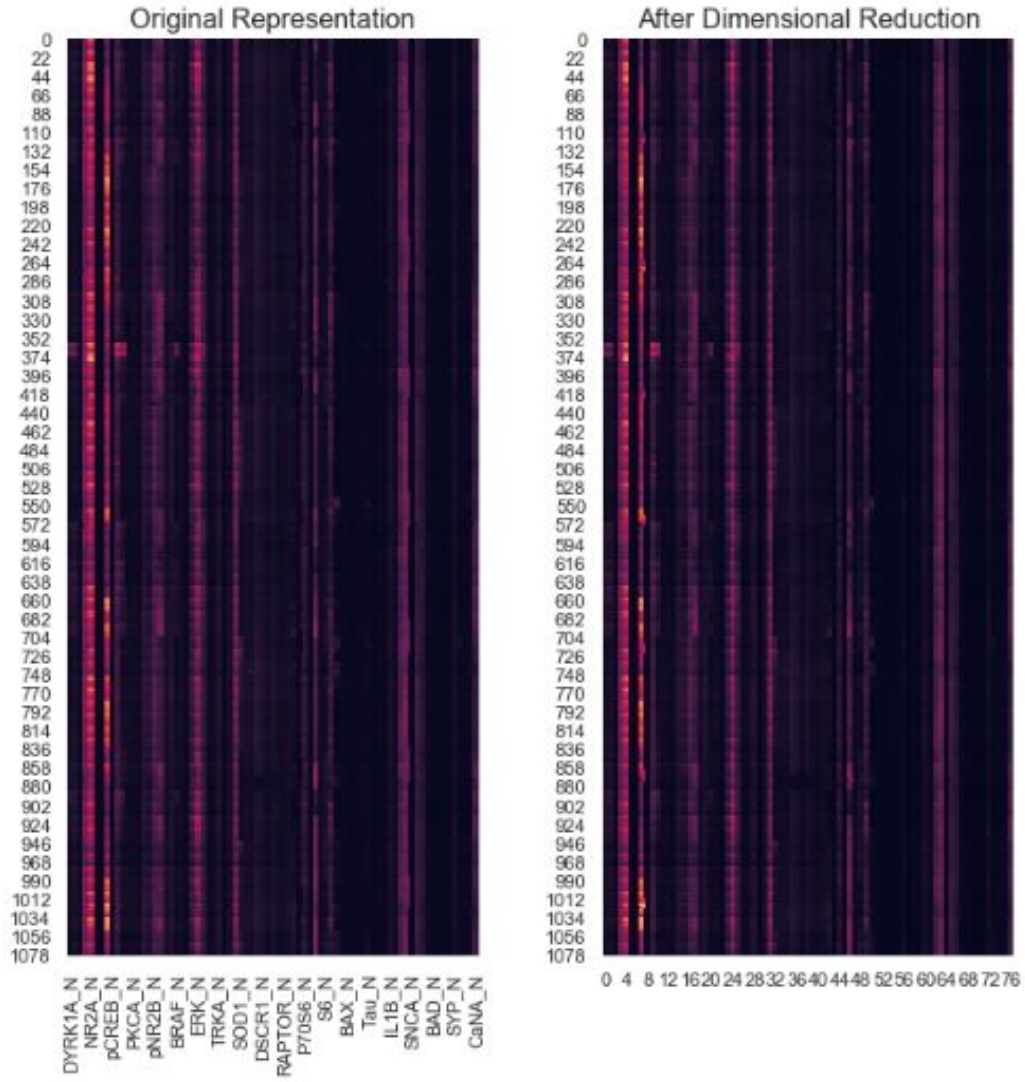
Explained variance by different principal components



Reconstruction Error Variation



Visualization of Reconstructed Data



Following are the insights of the above graphs :

- Explained variance is maximum for first two principal components of the graph , after which the decrease is very high (further principal components contribute to very less variance)
- Reconstruction Error goes on decreasing as we increase the number of attributes used as principal components (the more the number of components considered lesser is the error and hence more is the accuracy)
- Inspite of loss of some amount of data due to principal component analysis, the visualization of data shows that reconstructed data has less variation as compared to original one. This can be attributed from the fact that most significant variation was covered by top principal components that are selected for the compressed / reduced data.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTION

Principal Component Analysis (PCA) applied to high dimensional data results into reduced data which has fewer dimensions in a new feature space. The components that represent maximum variance from the original data set (principal components) are considered whereas others are discarded. As a result, there would be some error in the reconstructed data owing to loss of few features. The accuracy is reasonable if the major components are considered which reducing the data using PCA.

This technique , as mentioned earlier, is very useful in fields like Astronomy where the end users may need to analyze the data that is meaningful and not just raw data. The future scope would entail formulating a scalable technique of PCA to be applied in a distributed environment where local principal components are calculated and shared to compute global principal components.The factors affecting the performance, namely, network latency, bandwidth, accuracy may be considered to derive the efficiency metric of the technique.

BIBLIOGRAPHY

1. "A tutorial on Principal Component Analysis". http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf
2. "Introduction to Data Mining", Pang-Ning-Tan, Pearson Edition
3. "A Communication efficient distributed data mining for Astronomical data", Aruna Govada, Sanjay K. Sahay, Elsivier- Volume 16
4. "A covariance free iterative algorithm for distributed principal component analysis", Yue-Fei Guo, Pattern Recognition Volume 3,2012.

APPENDIX

Generalized PCA python script: GeneralizedPCA1.ipynb

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.plotly as py
import seaborn as sns

df1=pd.read_csv('mfeat-fac', header=None, delimiter=r"\s+")
#df1 = pd.read_excel('Data_Cortex_Nuclear.xls')

df1.head()

cols =len(df1.columns)
print(cols)

#replace the null values with the mean
v1_null=df1.iloc[:, 0:cols-1]
mean= v1_null.mean(axis=0)

v1_null.isnull().sum()
modified_v1 = []
modified_v1 = v1_null.apply(lambda v1 : v1.fillna(v1.mean()),axis=0)


#finding the covariance matrix
covariance_v1 = np.cov(modified_v1.transpose())

#finding the eigen values and eigen vectors
eigen_value, eigen_vector = np.linalg.eig(covariance_v1)
```

```
eigen_value.shape
len(eigen_value)
len(eigen_value.T)

tot = sum(eigen_value)
var_exp = [(i / tot)*100 for i in sorted(eigen_value, reverse=True)]

cum_var_exp = np.cumsum(var_exp)


k = 1; cum_var_exp1 = 0;
for j in sorted(eigen_value, reverse=True):
    cum_var_exp1 = cum_var_exp1 +(j / tot)*100
    if cum_var_exp1 >= 99.9 :
       print(cum_var_exp1)
       break
    else:
        k = k+1

comps = k
print(comps)


t1 = dict(
    type='bar',
    x=['PC %s' %i for i in range(0,cols-1)],
    y=var_exp,
    name='Individual'
)
```

```
t2 = dict(
    type='scatter',
    x=['PC %s' %i for i in range(0,cols-1)],
    y=cum_var_exp,
    name='Cumulative'
)

data = [t1, t2]

layout=dict(
    title='Explained variance by different principal components',
    yaxis=dict(
        title='Explained variance in percent'
    ),
    annotations=list([
        dict(
            x=1.16,
            y=1.05,
            xref='paper',
            yref='paper',
            text='Explained Variance',
            showarrow=False,
        )
    ])
)

fig = dict(data=data, layout=layout)
py.iplot(fig, filename='selecting-principal-components')


#finding the two principal components
eig_1 = []
eig_1 = eigen_vector[:,:k]
print(eig_1)
```

```
np.shape(eig_1)


eig_2 = eig_1.transpose()
np.shape(eig_2)
#eig_2 here is rowFeatureVector


z1=modified_v1.transpose()
np.shape(z1)


#finding the matrix multiplication of the principal components with the original dataset
rows = df1.shape[0]

z2 = []
for i in range(rows):
    z2 = np.dot(eig_2,z1)
    z3 = z2.transpose()
np.shape(z2)
#z2 here is final data

np.shape(z3)

zTrans = eig_2.transpose()
np.shape(zTrans)

for k in range(rows):
   oriData = np.dot(zTrans,z2)

np.shape(oriData)
```

```
oriDataTrans = oriData.transpose()
np.shape(oriDataTrans)

#Visualizing the Data with Heap Maps
sns.set()
sns.despine(left=True)
figure1, axes = plt.subplots(1, 2, figsize=(15, 15), gridspec_kw=dict(horispace=.4,
widspace=.3))
title_settings={'fontsize':16}
ax = sns.heatmap(modified_v1, cbar=False, ax=axes[0])
ax.set_title('Original Representation', **title_settings)

ax = sns.heatmap(oriDataTrans, cbar=False, ax=axes[1])
ax.set_title('After Dimensional Reduction', **title_settings)
print('Compare okay')

print(comps)
rem=comps%5
print(rem)
val=comps-rem+5
print(val)



errorValues=[]
for m in range(5,val+1,5) :

    eigen_1 = eigen_vector[:,:m]
    #print(eigen_1)
    np.shape(eigen_1)

    eigen_2 = eigen_1.transpose()
    np.shape(eigen_2)
    #eigen_2 here is rowFeatureVector
```

```
zErr=modified_v1.transpose()
np.shape(zErr)

 #finding the matrix multiplication of the principal components with the original dataset
zErr2=[]
for i in range(rows):
    zErr2=np.dot(eigen_2,zErr)
    z3=zErr2.transpose()
np.shape(zErr2)
#zErr2 here is final data

zTrans1=eigen_2.transpose()

for k in range(rows):
   oriData1 = np.dot(zTrans1,zErr2)



np.shape(oriData1)

oriData1Trans =oriData1.transpose()
np.shape(oriData1Trans)

sumErrorCol=0
for i in range(cols-1):
   for j in range(rows):
      sumErrorCol=sumErrorCol+abs(modified_v1.iloc[j][i]-oriData1Trans[j][i])

#print(cols)
avgError=sumErrorCol/cols-1
print(avgError)
avgError=avgError/rows
print(avgError)
print(avgError*100)
```

```
    errorValues.append(avgError)

#np.shape(oriData1Trans)
np.shape(modified_v1)

xPC=[]
for p in range(5,val+1,5):
    xPC.append(p)

#Error Variation graph

plt.plot(xPC, errorValues, color='red',marker='o',markerfacecolor='yellow',
linewidth=3)
title_settings2={'fontsize':18}
plt.xlabel('Principal Components',**title_settings2)
plt.ylabel('Error Percentage',**title_settings2)
plt.title('Variation in Reconstruction Error',**title_settings2)
plt.figure(figsize=(30,30),facecolor='white')
plt.show()
print('PCA Error in reconstruction okay!')
```