ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

CS498
Applied Machine Learning
Assignment #3

Students:
nidiaib2, Nidia Bucarelli
sunnyk2, Sunny Katiyar
wangx2, Wang Xiang

March 2, 2020
Spring 2020

PROBLEM 5.10

Part A.

From the plotted graphics below, it can be noticed:

- As the noise that's added to the dataset gets larger in terms of magnitude, the mean-squared error between the original dataset and the expansion of the noisy data onto 4 PCs gets larger.
- For smaller noise values (standard deviation 0.1, 0.2), we see that the MSE of the data represented using 4 PCs is lower than that of the data formed using fewer PCs. Whereas, for larger noise values (standard deviation 0.5, 0.1), we see that the MSE of the data represented using the first few PCs is lower than that of the data formed using more PCs.
- Therefore, as the noise gets larger, using fewer principal components gives a more accurate estimate of the original dataset.

Figure-1: MSE between original dataset and expansion onto 4 PCs using noisy data (SD=0.1)

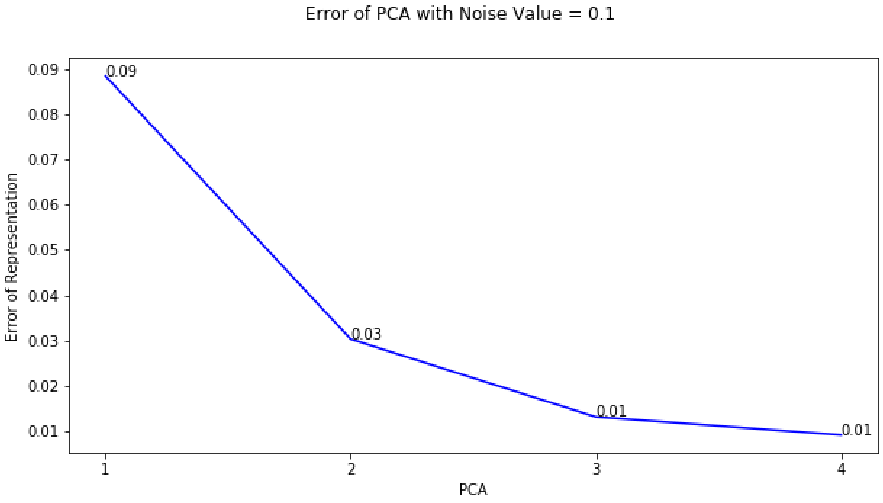


Figure-2: MSE between original dataset and expansion onto 4 PCs using noisy data (SD=0.2)

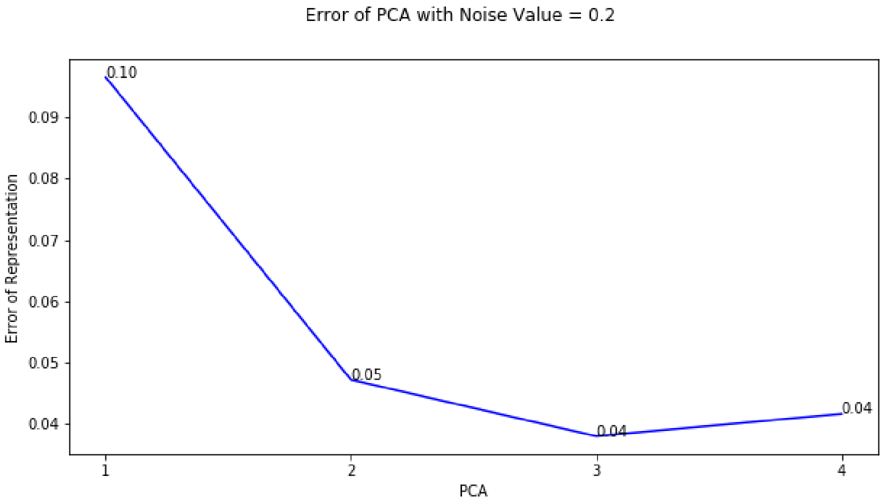


Figure-3: MSE between original dataset and expansion onto 4 PCs using noisy data (SD=0.5)

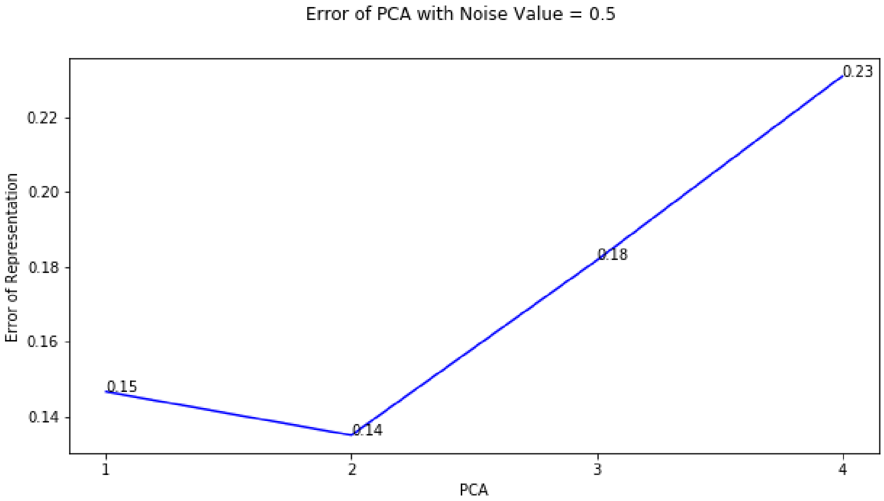


Figure-4: MSE between original dataset and expansion onto 4 PCs using noisy data (SD=1)

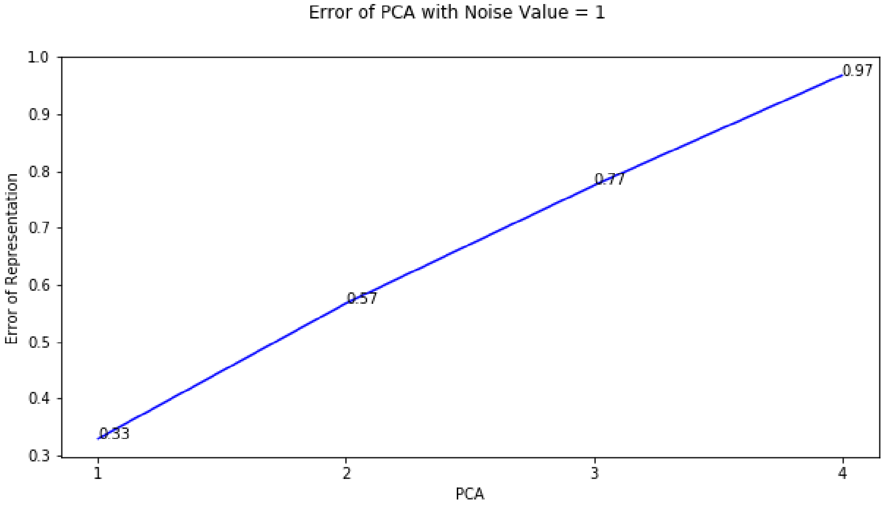
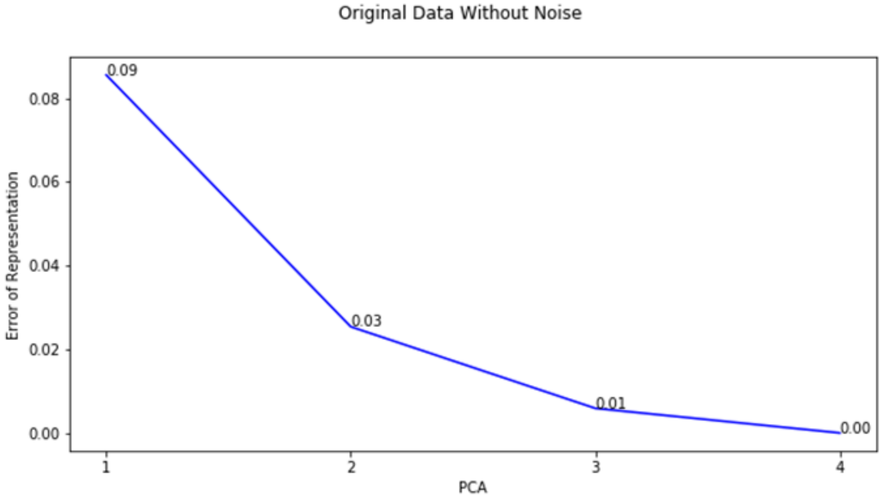


Figure-5 MSE between original dataset and expansion using 1-2-3-4 PCs without any added noise



Part B.

From the plotted graphic, it can be noticed:

- As the noise (W) gets larger, the error in the representation of the original dataset using PCAs gets larger.
- The error of representing the original dataset when the noise (W) is introduced gets larger as the noised dataset is expanded into more PCAs. That is, the lower number of PCA gives a better representation of the original dataset when nose (W) is introduced.
- The noise when W=10 is already too large for representing the dataset.

Figure-6 MSE between original dataset and expansion onto 4 PCs using noisy data

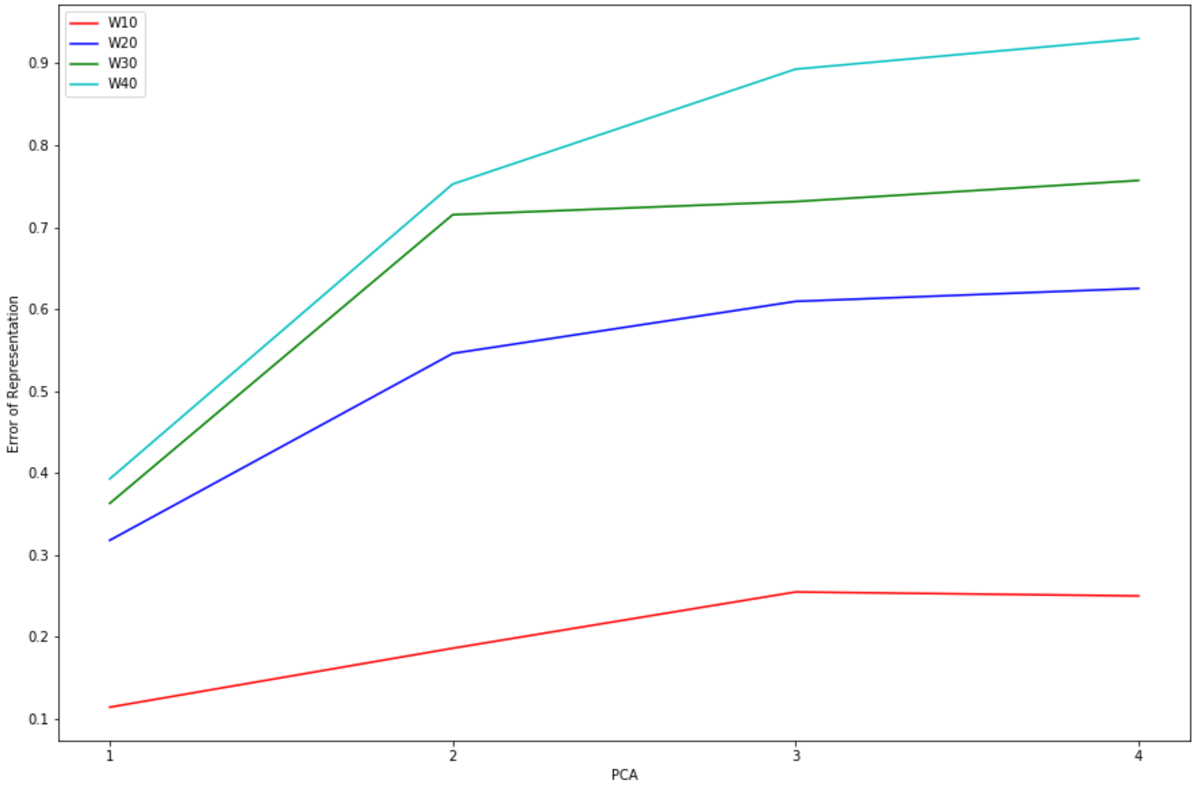


Figure-7 MSE between original dataset and expansion onto 4 PCs using noisy data (W=10)

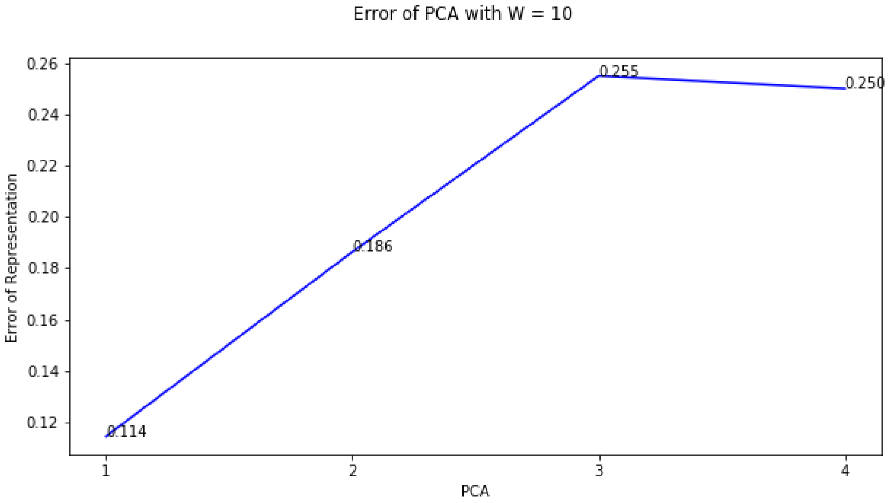


Figure-8 MSE between original dataset and expansion onto 4 PCs using noisy data (W=20)

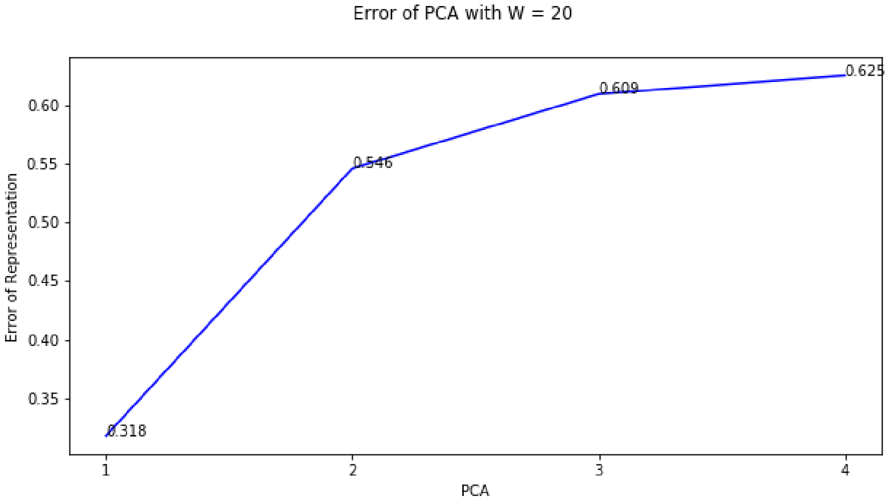


Figure-9 MSE between original dataset and expansion onto 4 PCs using noisy data (W=30)

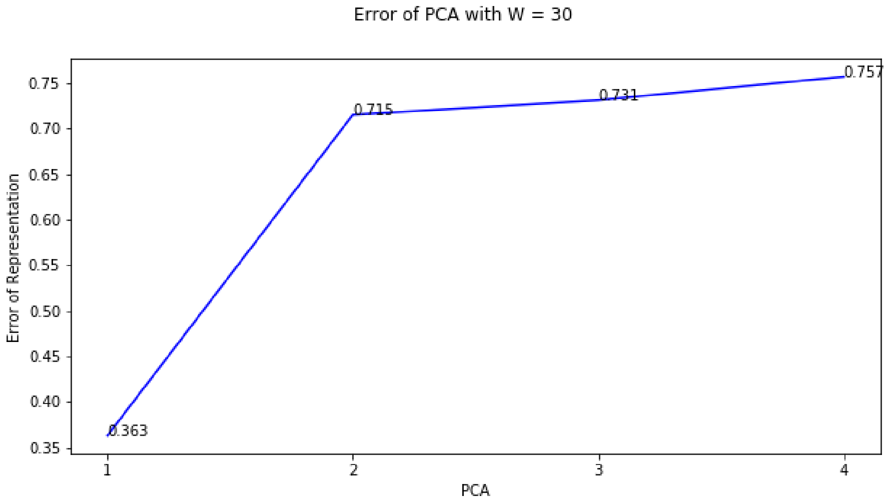
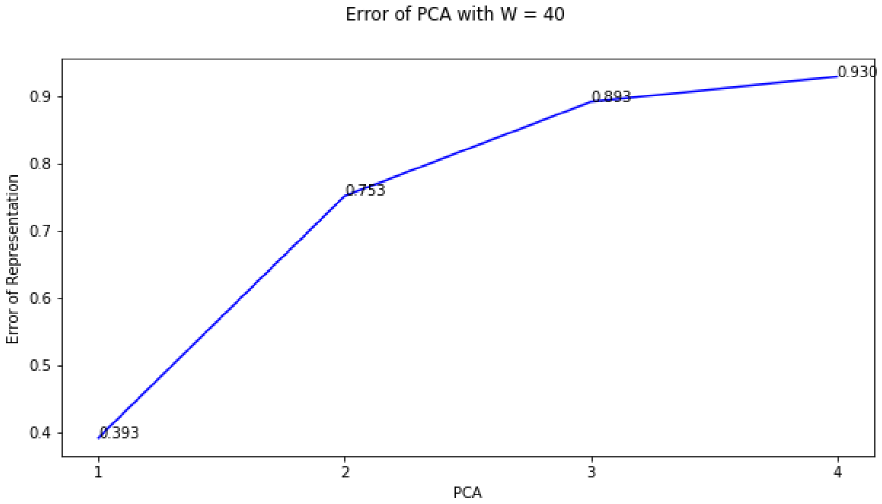


Figure-10 MSE between original dataset and expansion onto 4 PCs using noisy data (W=40)



Why do we see this trend? i.e. why do fewer principal components give a more accurate estimate of the original dataset as the noise gets larger?

The average error in the representation of the original dataset using PCAs will be the sum of error in components that are preserved and the error in components that are zeroed.

$$Total\ Error = s\sigma^2 + \sum_{j=s+1}^{d} \underline{\lambda}_u$$

Here, s is the number of components that are preserved.

When the noise variance is small, $s\sigma^2$ will be small and the total error will be dominated by the components that are zeroed. Hence, more PCs are required to bring this total error down.

However, when the noise variance is high, $s\sigma^2$ will be high and it will dominate the overall error. Therefore, only a few PCs will be needed to keep the total error low.

That is, as the noise gets larger, using fewer principal components gives a more accurate estimate of the original dataset.

Appendix: pdf copy of Python code attached below

CS498_HW3_Final Submission

March 2, 2020

Importing libraries and loading the dataset

```
In [36]: import pandas as pd
         import numpy as np
         import io
         import random
         import matplotlib.pyplot as plt
         from sklearn.decomposition import PCA
         from scipy.spatial import distance
         from sklearn import preprocessing
         from sklearn.metrics import mean_squared_error

         data= []
         with io.open('iris.data','r',encoding='utf-8') as f:
             lines = f.readlines()

         count=0
         for line in lines:
             count= count+ 1
             line= list(line)
             line.remove('\n')
             line= "".join(line)
             line= line.split(',')
             line= line[0:4]
             line= [np.float(i) for i in line]
             data.append(line)
             if count>=150:
                 break;
```

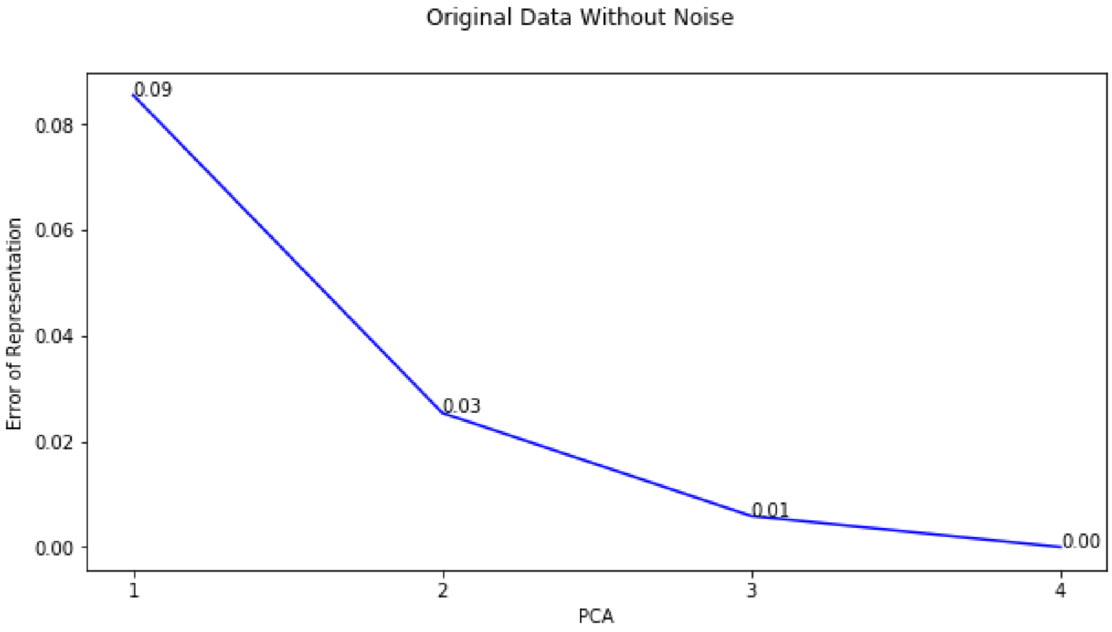
```
In [37]: data_new= np.asarray(data)
         mean= np.mean(data_new, axis=0)
```

Reconstructing the dataset using PCs without any noise added

```
In [38]: ##PRINCIPAL COMPONENT ANALYSIS
         data= data_new
         pca = PCA(n_components=4)
         pca.fit(data)
         PCA_r = pca.fit_transform(data)
         test=pca.inverse_transform(PCA_r)
         vect_u= pca.components_
         eigen= pca.singular_values_
         x_new= {}

         #RECONSTRUCTION OF NEW DATA
         for n in range(1,5):
             PCA_p= np.zeros((150, 4))
             PCA_p[:,0:n]= PCA_r[:,0:n]
             x_rec= np.dot(PCA_p,vect_u) + mean
             x_new[n]= x_rec

         ##ERROR

         err_PCA= []
         for i in range(1,5):
             err= mean_squared_error(x_new[i], data_new)
             err_PCA.append(err)
```

```
In [39]: fig= plt.figure(figsize=(10,5))
         x= [1,2,3,4]
         y=err_PCA
         plt.plot(x,y,'b')
         plt.xticks(list(range(1,max(x)+1)),[str(i) for i in range(1,max(x)+1)])
         for a,b in zip(x, y):
             plt.text(a, b, str("%.2f" % b))
         plt.xlabel('PCA')
         plt.ylabel('Error of Representation')

         plt.suptitle('Original Data Without Noise')
         plt.savefig('RawData.png')
         plt.figure()
```

```
Out[39]: <Figure size 432x288 with 0 Axes>
```



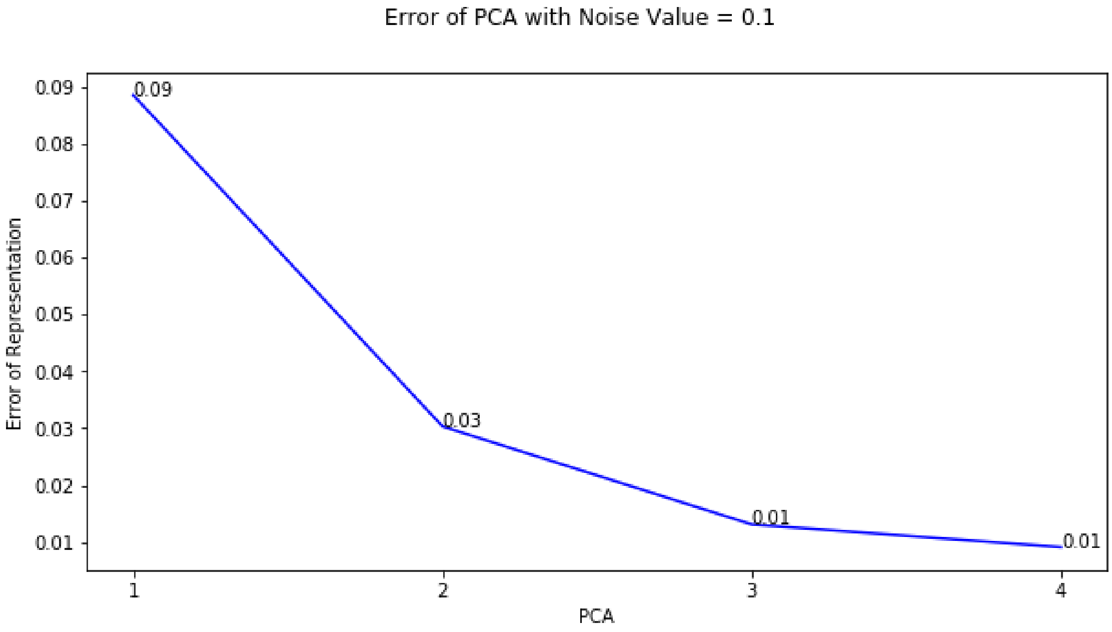
```
<Figure size 432x288 with 0 Axes>
```

0.1 PART A

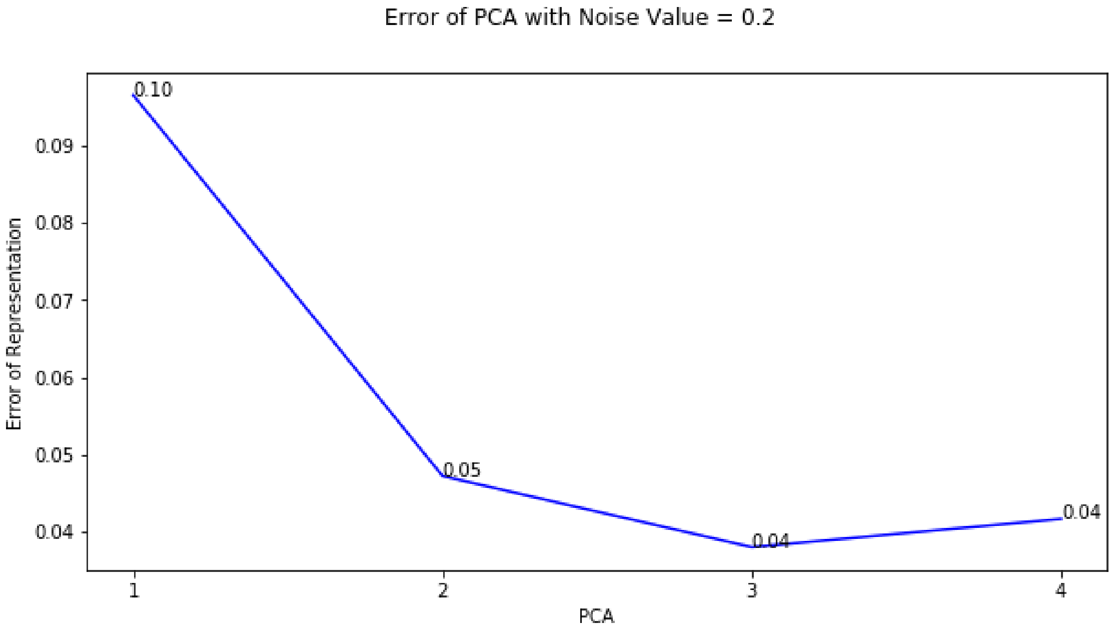
```
In [42]: std= [0.1,0.2,0.5,1]
         # mu= np.mean(data_new, axis=0)
         mu=0
         dataset= {}
         for value in range(0,len(std)):
             data_new2= np.zeros((150, 4))
             for i in range(0,4):
                 sample= np.random.normal(mu, std[value], 150)
                 data_new2[:,i]= data_new[:,i] + sample
             dataset[value]= data_new2

In [43]: for value in range(0,len(dataset)):
         ##PRINCIPAL COMPONENT ANALYSIS
             data= dataset[value]
             mean= np.mean(data, axis=0)
             pca = PCA(n_components=4)
             pca.fit(data)
             PCA_r = pca.fit_transform(data)
             vect_u= pca.components_
             eige= pca.singular_values_
             x_new= {}
```

```
#RECONSTRUCTION OF NEW DATA
for n in range(1,5):
    PCA_p= np.zeros((150, 4))
    PCA_p[:,0:n]= PCA_r[:,0:n]
    x_rec= np.dot(PCA_p,vect_u) + mean
    x_new[n]= x_rec

##ERROR
err_PCA= []

for i in range(1,5):
    err= mean_squared_error(x_new[i], data_new)
    err_PCA.append(err)

print (err_PCA)

##PLOT FIGURE
fig= plt.figure(figsize=(10,5))
x= [1,2,3,4]
y=err_PCA
plt.plot(x,y,'b')
plt.xticks(list(range(1,max(x)+1)),[str(i) for i in range(1,max(x)+1)])
for a,b in zip(x, y):
    plt.text(a, b, str("%.2f" % b))
plt.xlabel('PCA')
plt.ylabel('Error of Representation')

plt.suptitle('Error of PCA with Noise Value = ' + str(std[value]))
plt.savefig('PCA_std_' + str(std[value]) + '.png')
plt.figure()
```

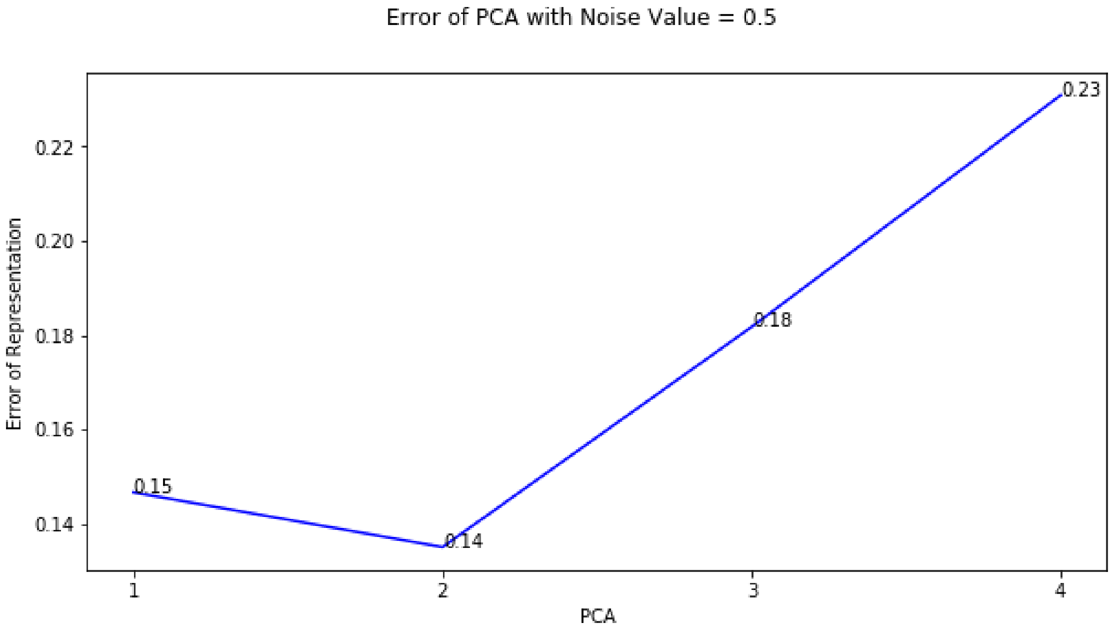
```
[0.08847859511180517, 0.030319826909533373, 0.013117506884252568, 0.009142604889811556]
[0.09652721882069906, 0.0472074672963243, 0.03801212059881355, 0.04164954816413982]
[0.14663017379506946, 0.1350217130240375, 0.18174034584569665, 0.23091012833713345]
[0.32932132763207733, 0.5672866123680635, 0.7747124825153555, 0.9685889329427407]
```


Error of PCA with Noise Value = 0.1
Error of Representation
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.09
0.03
0.01
0.01
1
2
3
4
PCA

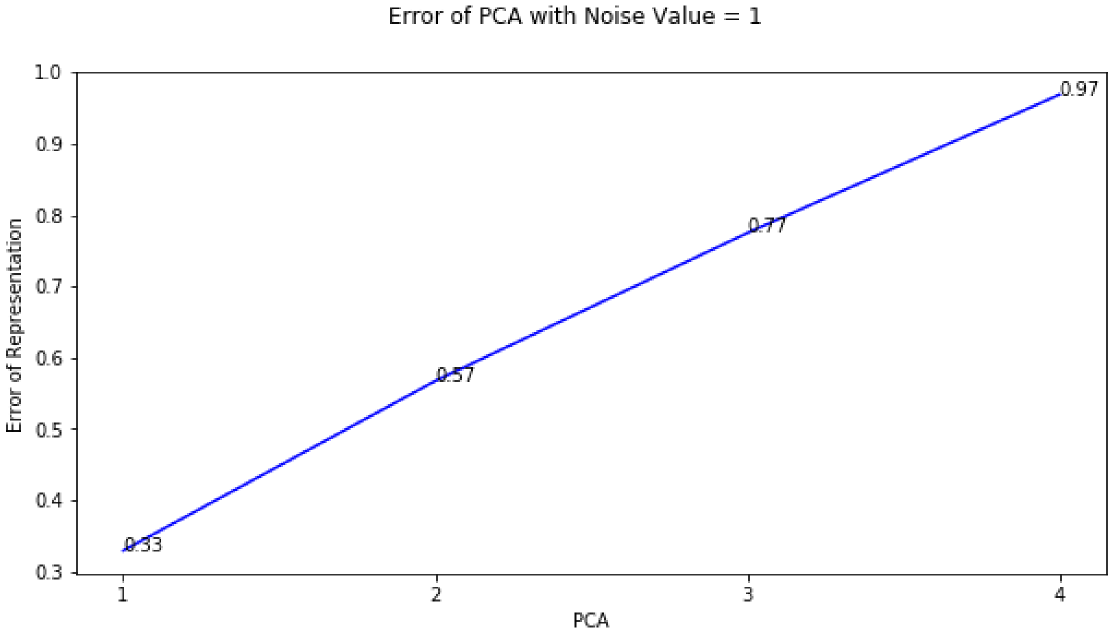
```
<Figure size 432x288 with 0 Axes>
```


Error of PCA with Noise Value = 0.2
Error of Representation
0.09
0.08
0.07
0.06
0.05
0.04
0.10
0.05
0.04
0.04
1
2
3
4
PCA

```
<Figure size 432x288 with 0 Axes>
```


Error of PCA with Noise Value = 0.5
Error of Representation
0.22
0.20
0.18
0.16
0.14
PCA
1
2
3
4
0.15
0.14
0.18
0.23

```
<Figure size 432x288 with 0 Axes>
```


Error of PCA with Noise Value = 1
Error of Representation
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
PCA
1
2
3
4
0.33
0.57
0.77
0.97

```
<Figure size 432x288 with 0 Axes>
```

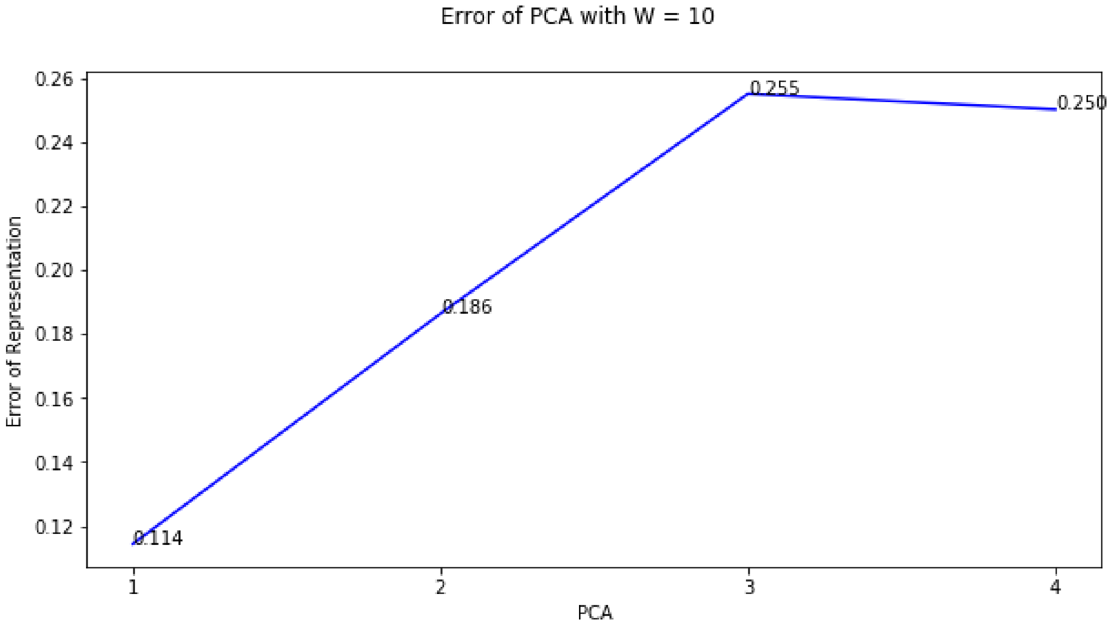
0.2 PART B

```
In [44]: W= [10,20,30,40]
         mu=0
         n=1
         dataset= {}
         for i in range(0,4):
             p= 1- (W[i]/600.0)
             mask= np.random.binomial(n, p, (150,4))
             data_new2= data_new*mask
             dataset[W[i]]= data_new2
```

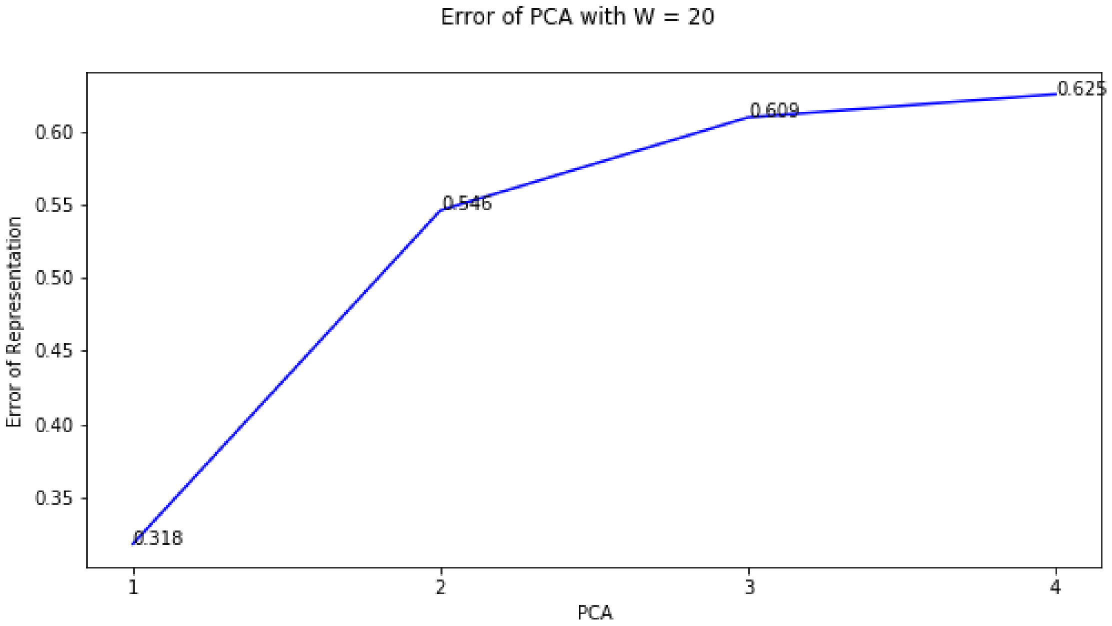
```
In [45]: pcaall=[]

         for value in range(0, len(dataset)):
         ##PRINCIPAL COMPONENT ANALYSIS
             data= dataset[W[value]]
             mean= np.mean(data, axis=0)
             pca = PCA(n_components=4)
             pca.fit(data)
             PCA_r = pca.fit_transform(data)
             vect_u= pca.components_
             eigen= pca.singular_values_
             x_new= {}

             #RECONSTRUCTION OF NEW DATA
             for n in range(1,5):
                 PCA_p= np.zeros((150, 4))
                 PCA_p[:,0:n]= PCA_r[:,0:n]
                 x_rec= np.dot(PCA_p,vect_u) + mean
                 x_new[n]= x_rec

             ##ERROR
             err_PCA=[]
             for i in range(1,5):
                 err= mean_squared_error(x_new[i], data_new)
                 err_PCA.append(err)

             print (err_PCA)

             ##PLOT FIGURE
             fig= plt.figure(figsize=(10,5))
             x= [1,2,3,4]
             y=err_PCA
             pcaall.append(y)
             plt.plot(x,y,'b')
             plt.xticks(list(range(1,max(x)+1)),[str(i) for i in range(1,max(x)+1)])
             for a,b in zip(x, y):
                 plt.text(a, b, str("%.3f" % b))
```

```
            plt.xlabel('PCA')
            plt.ylabel('Error of Representation')

            plt.suptitle('Error of PCA with W = ' + str(W[value]))
            plt.savefig('PCA_W_' + str(W[value]) + '.png')
            plt.figure()
```

```
[0.11441602688481442, 0.18633202239743923, 0.2549783933790244, 0.2501333333333336]
[0.31801445756042557, 0.545942916305165, 0.60945642221582, 0.6252499999999995]
[0.36307327859871547, 0.7152063634602678, 0.7312519503885865, 0.7569833333333323]
[0.3928875505238239, 0.7525750471281176, 0.892789780515326, 0.9301333333333324]
```

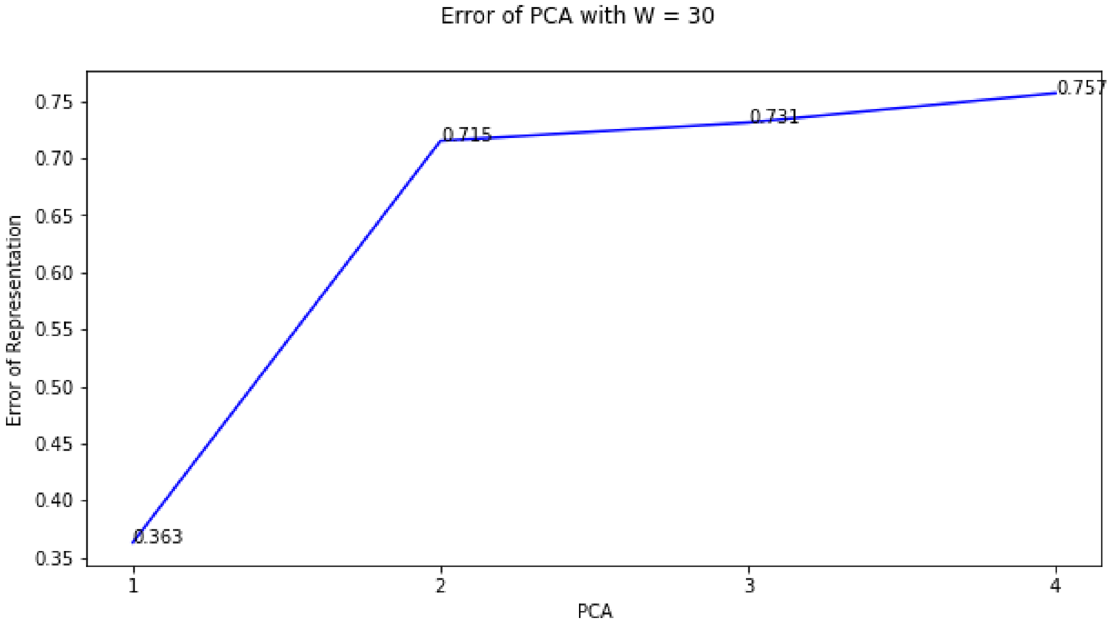


```
<Figure size 432x288 with 0 Axes>
```



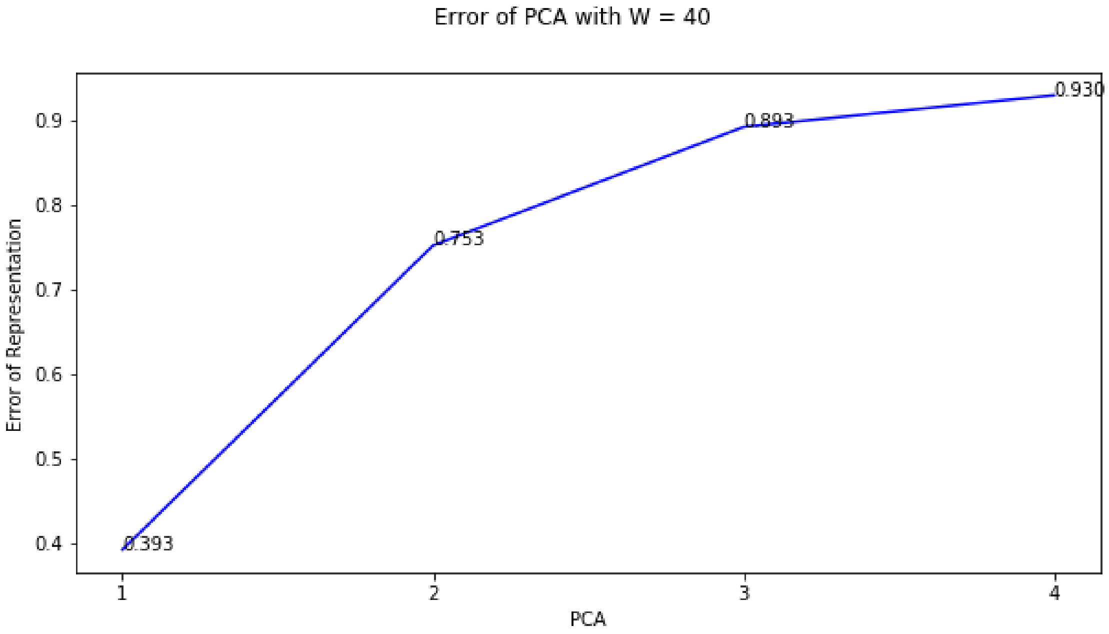
Error of PCA with W = 20

```
<Figure size 432x288 with 0 Axes>
```



Error of PCA with W = 30

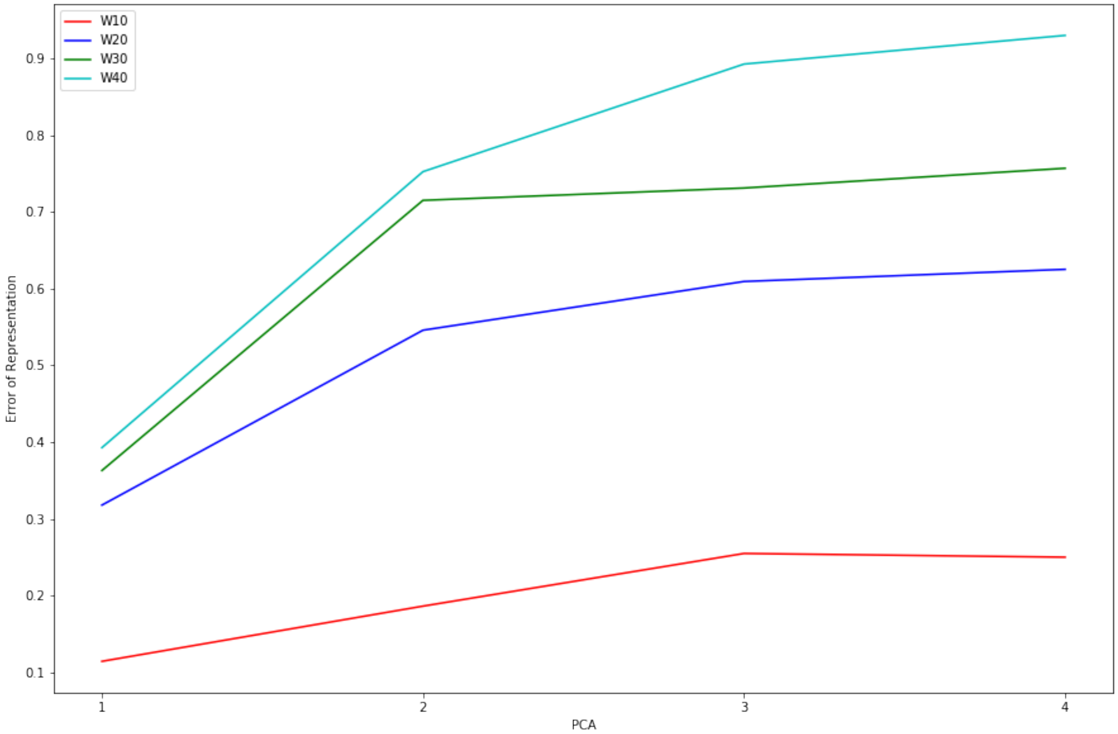
```
<Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```

```
In [52]: import matplotlib.pyplot as plt
         fig= plt.figure(figsize=(15,10))
         x= [1,2,3,4]
         plt.xticks(list(range(1,max(x)+1)),[str(i) for i in range(1,max(x)+1)])
         plt.plot(x,pcaall[0], 'r', label= 'W10')
         plt.plot(x,pcaall[1], 'b', label= 'W20')
         plt.plot(x,pcaall[2], 'g', label= 'W30')
         plt.plot(x,pcaall[3], 'c', label= 'W40')

         plt.xlabel('PCA')
         plt.ylabel('Error of Representation')


         plt.legend()
         plt.savefig('PCA_WALL.png')
         plt.figure()
```

```
Out[52]: <Figure size 432x288 with 0 Axes>
```


W10
W20
W30
W40
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Error of Representation
1
2
3
4
PCA

```
<Figure size 432x288 with 0 Axes>
```