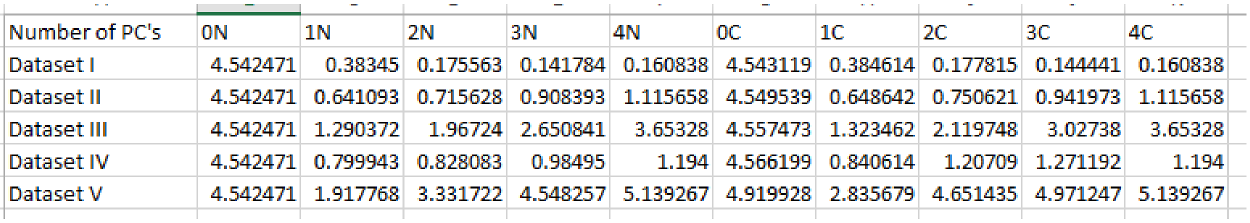
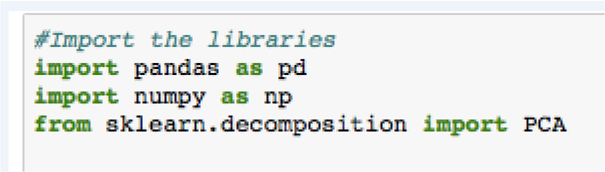
The table showing numbers filled in a table set out as below, where "N" columns represents the components calculated via the noiseless dataset and the "c" columns the noisy datasets.:

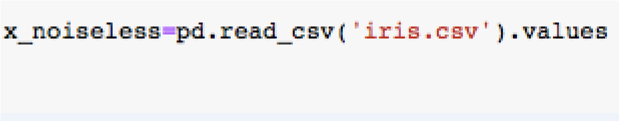


Number of PC's	0N	1N	2N	3N	4N	0C	1C	2C	3C	4C
Dataset I	4.542471	0.38345	0.175563	0.141784	0.160838	4.543119	0.384614	0.177815	0.144441	0.160838
Dataset II	4.542471	0.641093	0.715628	0.908393	1.115658	4.549539	0.648642	0.750621	0.941973	1.115658
Dataset III	4.542471	1.290372	1.96724	2.650841	3.65328	4.557473	1.323462	2.119748	3.02738	3.65328
Dataset IV	4.542471	0.799943	0.828083	0.98495	1.194	4.566199	0.840614	1.20709	1.271192	1.194
Dataset V	4.542471	1.917768	3.331722	4.548257	5.139267	4.919928	2.835679	4.651435	4.971247	5.139267

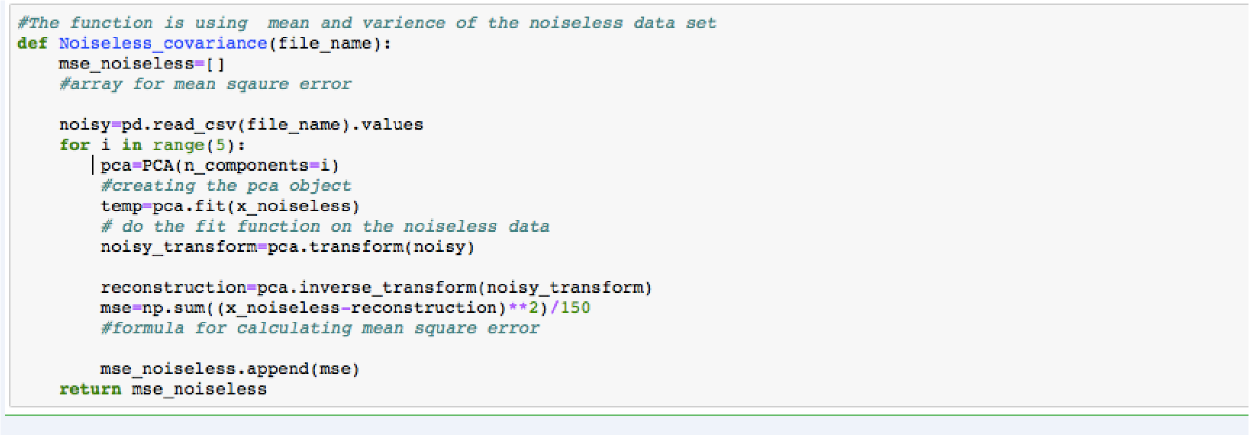
Code :



```
#Import the libraries
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
```



```
x_noiseless=pd.read_csv('iris.csv').values
```

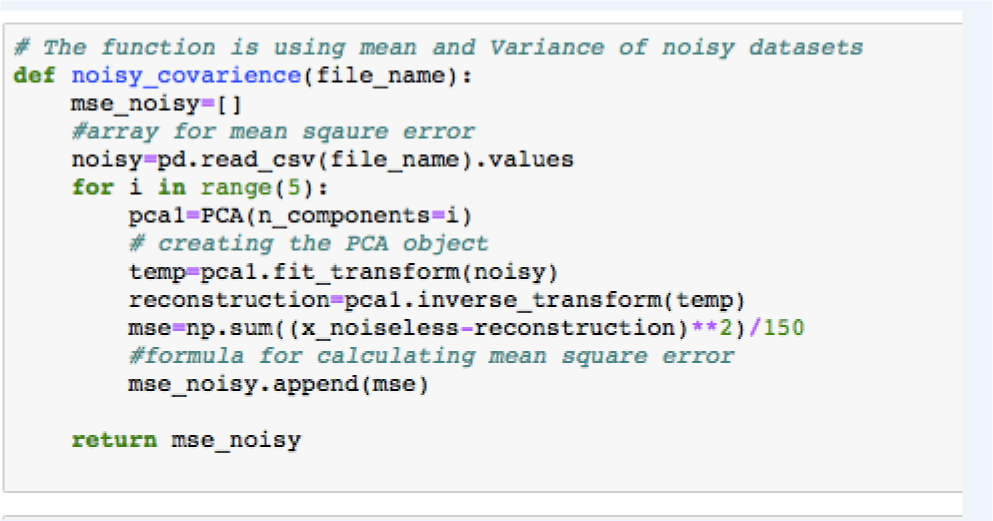


```
#The function is using  mean and varience of the noiseless data set
def Noiseless_covariance(file_name):
    mse_noiseless=[]
    #array for mean sqaure error

    noisy=pd.read_csv(file_name).values
    for i in range(5):
        pca=PCA(n_components=i)
        #creating the pca object
        temp=pca.fit(x_noiseless)
        # do the fit function on the noiseless data
        noisy_transform=pca.transform(noisy)

        reconstruction=pca.inverse_transform(noisy_transform)
        mse=np.sum((x_noiseless-reconstruction)**2)/150
        #formula for calculating mean square error

        mse_noiseless.append(mse)
    return mse_noiseless
```



```
# The function is using mean and Variance of noisy datasets
def noisy_covarience(file_name):
    mse_noisy=[]
    #array for mean sqaure error
    noisy=pd.read_csv(file_name).values
    for i in range(5):
        pca1=PCA(n_components=i)
        # creating the PCA object
        temp=pca1.fit_transform(noisy)
        reconstruction=pca1.inverse_transform(temp)
        mse=np.sum((x_noiseless-reconstruction)**2)/150
        #formula for calculating mean square error
        mse_noisy.append(mse)

    return mse_noisy
```