1. 50 MSE numbers filled in a table:

0N	1N	2N	3N	4N
4.54247066666666	0.383450311504985	0.175563000244339	0.14178364800457	0.160838361807638
4.54247066666666	0.641093184900985	0.715628487504957	0.908392907398274	1.11565785784931
4.54247066666666	1.2903724507598	1.96724039237987	2.65084113513274	3.65327973251111
4.54247066666666	0.799942743733825	0.828082554706742	0.984949768240668	1.194
4.54247066666666	1.91776774994606	3.33172210394033	4.54825719724983	5.13926666666667

0c	1c	2c	3c	4c
4.54311902907455	0.384613533957618	0.177815282669626	0.144440506031377	0.160838361807638
4.54953899271544	0.648642108410852	0.750621128999984	0.941972819285057	1.11565785784931
4.55747296393055	1.32346214804188	2.11974804928195	3.02737991997533	3.65327973251111
4.56619866666667	0.840614157257198	1.20708979682591	1.27119196718607	1.194
4.919928	2.83567942802643	4.65143450271709	4.97124727152559	5.13926666666666

2. Code Screenshots:

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
# from sklearn.metrics import mean_squared_error
from scipy.spatial import distance

component_list = [0, 1, 2, 3, 4]

df_1 = pd.read_csv('dataI.csv')
df_2 = pd.read_csv('dataII.csv')
df_3 = pd.read_csv('dataIII.csv')
df_4 = pd.read_csv('dataIV.csv')
df_5 = pd.read_csv('dataV.csv')
df_og = pd.read_csv('iris.csv')

data1 = df_1.values
data2 = df_2.values
data3 = df_3.values
data4 = df_4.values
data5 = df_5.values
og = df_og.values

all_data = {1: data1, 2: data2, 3: data3, 4: data4, 5: data5}
all_avg = {1: df_1.mean().values, 2: df_2.mean().values, 3: df_3.mean().values,
           4: df_4.mean().values, 5: df_5.mean().values}
og_avg = df_og.mean().values
```

```
def get_mse_list (data_set, og_data, data_avg, og_avg):
    n_list = []
    c_list = []
    for each in component_list:
        pca_c = PCA(n_components=each, svd_solver='full')
        pca_n = PCA(n_components=each, svd_solver='full')

        lower_c = pca_c.fit_transform(data_set)
        pca_n.fit(og_data)
        lower_n = pca_n.transform(data_set)
        result_c = pca_c.inverse_transform(lower_c)
        result_n = pca_n.inverse_transform(lower_n)

        # Calculating MSE using what is suggested by Piazza post
        n_list.append(get_dist(og_data, result_n))
        c_list.append(get_dist(og_data, result_c))
        del pca_c, pca_n
    return n_list + c_list
```

```
def get_dist (arrayA, arrayB):
    summation = 0.
    for eachA, eachB in zip(arrayA, arrayB):
        summation += distance.sqeuclidean(eachA, eachB)
    return summation / len(arrayA)
```

```
result_list = []
for key, data in all_data.items():
    result_list.append(get_mse_list(data, og, all_avg[key], og_avg))
```