Question 1

Load data and get the position of the project and using the dynamic parameter to get the data file

```
data1 = pd.read_csv(os.path.join(os.path.dirname(__file__), 'specs/SensorData_question1.csv'))
```

SensorData_question1

Input1	Input2	Input3	Input4	Input5	Input6	Input7	Input8	Input9	Input10	Input11	Input12
1.473	2.311	3.179	2.666	0.2795	0.2771	0.2234	0.1855	0.2539	1.138	1.111	4.712
1.46	2.377	3.214	2.92	0.2527	0.3064	0.02563	0.1965	0.3027	1.213	1.027	5.463
1.552	2.164	3.064	2.745	0.282	0.21	0.1721	0.1929	0.21	1.221	1.058	5.332
1.605	2.228	3.149	2.834	0.2917	0.3613	0.2087	0.1294	0.2734	1.144	1.062	4.829
1.534	2.114	3.309	2.976	0.21	0.2502	0.2258	0.177	0.2039	1.254	1.112	5.734
1.796	2.262	3.18	2.888	0.2966	0.1477	0.1624	0.282	0.2039	1.172	1.03	4.861
1.566	2.323	3.469	2.711	0.2417	0.05371	0.2112	-0.02686	0.2197	1.158	0.9924	4.78
1.425	2.152	3.287	2.781	0.2991	0.2075	0.1038	0.1147	0.2698	1.271	1.115	5.662
1.595	2.271	3.323	2.743	0.1733	0.1965	0.1685	0.05859	0.2051	1.29	1.033	5.145
1.628	2.211	3.176	2.71	0.08423	0.1892	0.2783	0.1685	0.3491	1.155	1.008	5.613
1.583	2.253	3.278	2.854	0.1489	0.1099	0.2283	0.07324	0.2173	1.135	1.112	5.549
1.454	2.277	3.322	2.866	0.3015	0.2832	0.2124	0.1562	0.2637	1.14	1.229	4.929
1.617	2.183	3.226	2.71	0.1221	0.2563	0.01587	0.1404	0.2856	1.255	1.033	5.376
1.482	2.344	3.181	2.894	0.2344	0.1782	0.2307	0.2051	0.2637	0.9558	1.086	5.157
1.443	2.26	3.143	2.705	0.2808	0.09644	0.1929	0.2197	0.2344	1.133	1.083	5.645
1.477	2.349	2.998	2.994	0.2466	0.1953	0.1794	0.09277	0.0415	1.227	1.075	5.686
1.527	2.206	3.322	2.892	0.21	0.1465	0.09521	0.1648	0.1929	1.222	1.071	5.2
1.599	2.296	3.439	2.662	0.2014	0.1941	0.2271	0.1343	0.08789	1.183	1.118	5.364
1.587	2.308	3.364	2.866	0.2625	0.1624	0.1025	0.2246	0.2991	1.218	1.158	4.751
1.521	2.366	3.203	2.986	0.2173	0.271	0.177	0.1306	0.2844	1.243	1.006	5.056
1.562	2.258	3.313	2.654	0.1648	0.1575	0.1807	0.177	0.376	1.117	1.086	5.443
1.438	2.57	3.169	2.77	0.1282	0.09277	0.1599	0.06836	0.2795	1.345	1.056	5.347

As can be seen from the picture, there are 12 features in the data.

Next, add two columns ‘Original Input3’ 'and ‘Original Input12’ to the original data and assign the value of the columns ‘Input3’ and ‘Input12’ to the columns ‘Original Input3’ 'and ‘Original Input12’

```
data1['Original Input3'] = round(data1['Input3'], 5)
data1['Original Input12'] = round(data1['Input12'], 5)
```

Using function round( ) to handle the precision problem of floating point arithmetic.

Input10	Input11	Input12	Original Input3	Original Input12
1.13800	1.11100	4.71200	3.17900	4.71200
1.21300	1.02700	5.46300	3.21400	5.46300
1.22100	1.05800	5.33200	3.06400	5.33200
1.14400	1.06200	4.82900	3.14900	4.82900
1.25400	1.11200	5.73400	3.30900	5.73400
1.17200	1.03000	4.86100	3.18000	4.86100
1.15800	0.99240	4.78000	3.46900	4.78000
1.27100	1.11500	5.66200	3.28700	5.66200
1.29000	1.03300	5.14500	3.32300	5.14500
1.15500	1.00800	5.61300	3.17600	5.61300
1.13500	1.11200	5.54900	3.27800	5.54900
1.14000	1.22900	4.92900	3.32200	4.92900
1.25500	1.03300	5.37600	3.22600	5.37600
0.95580	1.08600	5.15700	3.18100	5.15700
1.13300	1.08300	5.64500	3.14300	5.64500
1.22700	1.07500	5.68600	2.99800	5.68600
1.22200	1.07100	5.20000	3.32200	5.20000
1.18300	1.11800	5.36400	3.43900	5.36400
1.21800	1.15800	4.75100	3.36400	4.75100

Next, apply z-score normalisation on the columns 'Input3', $z = \frac{x-u}{s}$, where u is the mean of the data, and s is the standard deviation of the data.

```
data1['Input3'] = (data1['Input3'] - data1['Input3'].mean()) / data1['Input3'].std()
```

Apply [0-1] normalisation on the columns 'Input12'

$x_{new} = \frac{x-x_{min}}{x_{mx}-x_{min}}$, where x_{min} is the mean of the minimum of x, and x_{max} is the mean of the maximum of x.

```
data1['Input12'] = (data1['Input12'] - data1['Input12'].min()) / \
                   (data1['Input12'].max() - data1['Input12'].min())
```

Input2	Input3	Input4	Input5	Input6	Input7	Input8	Input9	Input10	Input11	Input12	Original
2.31100	-1.10789	2.66600	0.27950	0.27710	0.22340	0.18550	0.25390	1.13800	1.11100	0.02297	3.17900
2.37700	-1.07844	2.92000	0.25270	0.30640	0.02563	0.19650	0.30270	1.21300	1.02700	0.68640	3.21400
2.16400	-1.20466	2.74500	0.28200	0.21000	0.17210	0.19290	0.21000	1.22100	1.05800	0.57067	3.06400
2.22800	-1.13314	2.83400	0.29170	0.36130	0.20870	0.12940	0.27340	1.14400	1.06200	0.12633	3.14900
2.11400	-0.99851	2.97600	0.21000	0.25020	0.22580	0.17700	0.20390	1.25400	1.11200	0.92580	3.30900
2.26200	-1.10705	2.88800	0.29660	0.14770	0.16240	0.28200	0.20390	1.17200	1.03000	0.15459	3.18000
2.32300	-0.86389	2.71100	0.24170	0.05371	0.21120	-0.02686	0.21970	1.15800	0.99240	0.08304	3.46900
2.15200	-1.01702	2.78100	0.29910	0.20750	0.10380	0.11470	0.26980	1.27100	1.11500	0.86219	3.28700
2.27100	-0.98673	2.74300	0.17330	0.19650	0.16850	0.05859	0.20510	1.29000	1.03300	0.40548	3.32300
2.21100	-1.11042	2.71000	0.08423	0.18920	0.27830	0.16850	0.34910	1.15500	1.00800	0.81890	3.17600
2.25300	-1.02459	2.85400	0.14890	0.10990	0.22830	0.07324	0.21730	1.13500	1.11200	0.76237	3.27800
2.27700	-0.98757	2.86600	0.30150	0.28320	0.21240	0.15620	0.26370	1.14000	1.22900	0.21466	3.32200
2.18300	-1.06835	2.71000	0.12210	0.25630	0.01587	0.14040	0.28560	1.25500	1.03300	0.60954	3.22600
2.34400	-1.10621	2.89400	0.23440	0.17820	0.23070	0.20510	0.26370	0.95580	1.08600	0.41608	3.18100
2.26000	-1.13818	2.70500	0.28080	0.09644	0.19290	0.21970	0.23440	1.13300	1.08300	0.84717	3.14300
2.34900	-1.26019	2.99400	0.24660	0.19530	0.17940	0.09277	0.04150	1.22700	1.07500	0.88339	2.99800
2.20600	-0.98757	2.89200	0.21000	0.14650	0.09521	0.16480	0.19290	1.22200	1.07100	0.45406	3.32200
2.29600	-0.88913	2.66200	0.20140	0.19410	0.22710	0.13430	0.08789	1.18300	1.11800	0.59894	3.43900
2.30800	-0.95223	2.86600	0.26250	0.16240	0.10250	0.22460	0.29910	1.21800	1.15800	0.05742	3.36400
2.36600	-1.08770	2.98600	0.21730	0.27100	0.17700	0.13060	0.28440	1.24300	1.00600	0.32686	3.20300

Check if the 'output' folder exists, create if it does not and export the data to csv.

```
if not os.path.exists(os.path.join(os.path.dirname(__file__), 'output')):
    os.makedirs(os.path.join(os.path.dirname(__file__), 'output'))
    data1.to_csv(os.path.join(os.path.dirname(__file__), 'output/question1_out.csv'), index=False)
data1.to_csv(os.path.join(os.path.dirname(__file__), 'output/question1_out.csv'), index=False)
```

Question 2

Using the tool sklearn to do dimensionality reduction. First import the library 'from sklearn.decomposition import PCA'. Converting a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. n_components=0.95 means that select 95% principal component to represent the all features.

```
pca = PCA(n_components=0.95)
new_data = pca.fit_transform(data2)
```

Next, segmenting data values into 10 bins of equal width

```
for idx in range(new_data.shape[1]):
    data2['pca' + str(idx) + '_width'] = pd.cut(new_data[:, idx], 10)
```

Segment data values into 10 bins of frequency width

```
for idx in range(new_data.shape[1]):
    data2['pca' + str(idx) + '_freq'] = pd.qcut(new_data[:, idx], 10)
```

pca18_width	pca19_width	pca20_width	pca21_width	pca0_freq	pca1_freq	pca2_freq	pca3_freq
-0.904, -0.446]	(-0.263, 0.241]	(-0.563, -0.117]	(0.736, 1.212]	(-6.259, -5.312]	(3.841, 7.945]	(1.523, 2.097]	(-1.14, -0.623]
).931, 1.39]	(-0.263, 0.241]	(-1.009, -0.563]	(-2.125, -1.644]	(-6.259, -5.312]	(3.841, 7.945]	(-0.321, 0.468]	(-1.567, -1.14]
-1.363, -0.904]	(-0.768, -0.263]	(-1.455, -1.009]	(0.26, 0.736]	(-4.164, -3.893]	(3.841, 7.945]	(-0.321, 0.468]	(-1.14, -0.623]
-0.446, 0.0132]	(-1.272, -0.768]	(-0.117, 0.329]	(0.26, 0.736]	(-3.893, -2.289]	(1.992, 3.841]	(2.097, 5.897]	(-1.567, -1.14]
2.307, 2.766]	(0.241, 0.745]	(0.329, 0.775]	(0.736, 1.212]	(-6.259, -5.312]	(3.841, 7.945]	(1.523, 2.097]	(0.661, 1.261]
).0132, 0.472]	(-1.272, -0.768]	(0.329, 0.775]	(1.212, 1.688]	(-8.295, -6.259]	(7.945, 13.626]	(2.097, 5.897]	(0.0284, 0.661]
-1.363, -0.904]	(0.745, 1.25]	(-0.563, -0.117]	(-2.125, -1.644]	(-6.259, -5.312]	(3.841, 7.945]	(1.523, 2.097]	(-0.623, 0.0284]
).472, 0.931]	(-0.768, -0.263]	(-0.563, -0.117]	(-0.692, -0.216]	(-8.295, -6.259]	(7.945, 13.626]	(0.468, 1.523]	(-0.623, 0.0284]
-0.446, 0.0132]	(-0.768, -0.263]	(0.329, 0.775]	(-0.216, 0.26]	(-8.295, -6.259]	(7.945, 13.626]	(1.523, 2.097]	(0.0284, 0.661]
-0.904, -0.446]	(-1.272, -0.768]	(-0.563, -0.117]	(-0.692, -0.216]	(2.433, 5.334]	(1.992, 3.841]	(5.897, 12.503]	(5.455, 10.496]
).0132, 0.472]	(-0.263, 0.241]	(-1.905, -1.455]	(0.26, 0.736]	(-6.259, -5.312]	(7.945, 13.626]	(1.523, 2.097]	(0.661, 1.261]
-1.363, -0.904]	(0.241, 0.745]	(1.221, 1.667]	(-0.692, -0.216]	(-8.295, -6.259]	(7.945, 13.626]	(1.523, 2.097]	(1.261, 5.455]
l.39, 1.848]	(-0.263, 0.241]	(1.221, 1.667]	(0.736, 1.212]	(-8.295, -6.259]	(7.945, 13.626]	(0.468, 1.523]	(1.261, 5.455]
).0132, 0.472]	(-0.263, 0.241]	(0.329, 0.775]	(0.26, 0.736]	(-6.259, -5.312]	(3.841, 7.945]	(0.468, 1.523]	(-2.639, -1.567]
l.848, 2.307]	(1.25, 1.754]	(-0.563, -0.117]	(-1.168, -0.692]	(-5.312, -4.47]	(1.992, 3.841]	(-1.442, -0.849]	(-2.639, -1.567]
).0132, 0.472]	(0.745, 1.25]	(0.329, 0.775]	(-0.216, 0.26]	(-5.312, -4.47]	(0.841, 1.992]	(-0.849, -0.321]	(-4.752000000000001, -3.066]
).472, 0.931]	(-0.263, 0.241]	(-0.563, -0.117]	(0.26, 0.736]	(-5.312, -4.47]	(0.841, 1.992]	(-0.849, -0.321]	(-3.066, -2.639]
-0.446, 0.0132]	(0.241, 0.745]	(-0.117, 0.329]	(0.26, 0.736]	(-6.259, -5.312]	(1.992, 3.841]	(-0.849, -0.321]	(-2.639, -1.567]
-0.446, 0.0132]	(0.241, 0.745]	(-0.117, 0.329]	(0.736, 1.212]	(-6.259, -5.312]	(1.992, 3.841]	(-0.321, 0.468]	(-3.066, -2.639]
2.307, 2.766]	(0.745, 1.25]	(-0.563, -0.117]	(-0.692, -0.216]	(-8.295, -6.259]	(3.841, 7.945]	(-1.442, -0.849]	(-2.639, -1.567]
).931, 1.39]	(1.25, 1.754]	(0.329, 0.775]	(-1.168, -0.692]	(-6.259, -5.312]	(1.992, 3.841]	(-2.86, -1.442]	(-3.066, -2.639]