**Q1. Data Transformation.**

**The file SensorData question1.csv contains data obtained from a sensory**

**system. Some of the attributes in the file need to be normalised, but you do**

**not want to loose the original values.**

**1. Generate a new attribute called Original Input3 which is a copy of the**

**attribute Input3. Do the same with the attribute Input12 and copy it into**

**Original Input12.**

**2. Normalise the attribute Input3 using the z-score transformation method.**

**3. Normalise the attribute Input12 in the range [0.0, 1.0].**

**4. Generate a new attribute called Average Input, which is the average of all**

**the attributes from Input1 to Input12. This average should include the**

**normalised attributes values but not the copies that were made of these.**

**5. Save the newly generated dataset to ./output/question1 out.csv.**

1. Firstly we have to read the data from file using pandas and preserve the column ***Input3*** and***Input12*** by generating a copy of these into new column ***Original Input3***and ***Original Input12****.*
2. Applied Z-Score Normalization on column ***Input3***. Z-Score can be calculated on column using the formula **df[col] = (df[col] - df[col].mean())/df[col].std()**.
3. Applied MinMax Scaling on column ***Input12****.* Calculated max and min value of the column and used the formula to scale the range between 0-1. **df[col] = (df[col] - min) / (max - min).**
4. Generated a new column ***Average Input*** by taking mean of columns from Input1 to Input12. We had to make sure that axis=1 while calculating mean so that the means are calculated row-wise .
5. Saved the newly generated csv file using ***to\_csv*** function and removed the index by keeping the value it as ***False.***

**Q2. Data Reduction and Discretisation.**

**The files DNAData question2 a.csv contains biological data arranged into multiple columns. We need to compress the information contained in the data.**

**1. Reduce the number of attributes using Principal Component Analysis (PCA), making sure at least 95% of all the variance is explained.**

**2. Discretise the PCA-generated attribute subset into 10 bins, using bins of equal width. For each component X that you discretise, generate a new column in the original dataset named pcaX width. For example, the first discretised principal component will correspond to a new column called**

**pca1 width.**

**3. Discretise PCA-generated attribute subset into 10 bins, using bins of equal frequency (they should all contain the same number of points). For each component X that you discretise, generate a new column in the original dataset named pcaX freq. For example, the first discretised principal component will correspond to a new column called pca1 width**

1. Read the file DNAData\_question2\_a.csv and stored it into a dataframe. Imported Principal Component Analysis(PCA) class from ***sklearn.decomposition*** package. Visualized the data using the matplotlib to get an idea, how many n\_components or dimension we need to keep the variance above 95%. Once that done, created an object of PCA and fitted the data into it. Set the ***n\_components=0.95*** so as to keep the variance above 95%. Stored the transformed data into a variable for further use.
2. Discretised the PCA generated attributes into 10 bins using ***sklearn.preprocessing.KBinsDiscretizer***. Set the attributes ***n\_bins =10*** for distributing it into 10 bins, ***encode=’ordinal’*** for numerical data and ***strategy=’uniform’*** because we had to split the data into equal width. Then fit and transformed the data and generated the new columns pca1\_width to pca22\_width to original data file.
3. Similarly we can also do binning based on frequency. We only have to change the ***strategy=’quantile’*** in ***sklearn.preprocessing.KBinsDiscretizer.***Generated the new columns pca1\_freq to pca22\_freq in the original dataset.
4. Saved the updated original data file into the directory.