COMP40370 Practical 2

Ryan Jennings 19205824

Question 1: Data Transformation

1. For question 1.1 a simple addition of two new columns that are just copies of the columns “Input3” and “Input12”.
   ```
   df_1['Original Input3'] = df_1['Input3']
   df_1['Original Input12'] = df_1['Input12']
   ```
2. This part involves normalising the Input3 column using the z-score transformation method. This normalises the data by the data’s standard deviation from the mean. There are a number of methods to do this. Two of them are:
   ```
   df_1['Input3'] = (df_1['Input3'] -
   df_1['Input3'].mean())/df_1['Input3'].std(ddof=0)
   ```
   and
   ```
   from sklearn.preprocessing import StandardScaler
   df_1['Input3'] = StandardScaler().fit_transform(df_1.loc[:,
   ['Input3']])
   ```
   But the test_practical2.py file has a different value from the above two methods, the values from test_practical2.py can be obtained with
   ```
   df_1['Input3'] = (df_1['Input3'] -
   df_1['Input3'].mean())/df_1['Input3'].std()
   ```
   without the `ddof=0`
3. The third part is a normalisation but this time instead of using a z-score transformation but between 0 and 1. The formula for this method is:
   ```
   df_1['Input12'] = (df_1['Input12'] - df_1['Input12'].min()) /
   (df_1['Input12'].max() - df_1['Input12'].min())
   ```
4. 1.4 is another python oneliner where you sum up the values in each row if the column is of the format InputX where X ∈ [1,12].
   ```
   df_1['Average Input'] = df_1[[col for col in df_1.columns if
   col.startswith('Input')]].mean(axis=1)
   ```
5. Finally for question 1 just write the dataframe to a csv file, ensuring to state the floating point format so avoid floating point errors when checking in the test class.
   ```
   df_1.to_csv('./output/question1_out.csv', index=False,
   float_format='%g')
   ```

Question 2: Data Reduction and Discretisation

- PCA for 95% explained_variance_ratio. From 59 columns to 22
- Kbins width (test starting with 0 rather than 1 in the doc issue)
- Kbins freq
- (not clear what the end dataset should be issue, went by tests)

1. The first part of question 2 involves applying Principal Component Analysis to reduce the amount of columns in the dataset but retaining at least 95% of the information. 22 components ended up being enough to convey the 95% of data.

```
pca_dna = PCA(n_components=22)
pc_arr = pca_dna.fit_transform(df_2)
df_pc = pd.DataFrame(data=pc_arr, columns=['p' + str(i+1) for
i in range(22)])
```

   Then we can verify that the variance is above 95%

```
pca_dna.explained_variance_ratio_.sum() > 0.95
```

2. I was able to make use of sklearn's KBinsDiscretizer method to divide the DNA dataset into bins of equal width.

```
width_discretizer = KBinsDiscretizer(n_bins=10,
encode='ordinal', strategy='uniform')
w_des = width_discretizer.fit_transform(df_pc)
df_w_des = pd.DataFrame(data=w_des,
                        columns=['pca'+str(i)+'_width' for i
in range(len(df_pc.columns))])
```

   I found though that enough though the practical2.pdf stated that the pcaX_width columns and pcaX_freq columns should start at 1, the test_practical2.py file tested that the values started at 0 so I made that change to pass the tests.

3. Similar to 2.2 I used KBinsDiscretizer to divide the dataset again for bins of equal frequency.

```
freq_discretizer = KBinsDiscretizer(n_bins=10,
                                    encode='ordinal',
                                    strategy='quantile')
f_des = freq_discretizer.fit_transform(df_pc)
df_f_des = pd.DataFrame(data=f_des,
                        columns=['pca'+str(i)+'_freq' for i
in range(len(df_pc.columns))])
```

4. Concatenate the different dataframes. I wasn't sure what dataframes had to be included as it wasn't stated so I based it off of the shape tests in the test file.

```
df_out = pd.concat([df_2, df_w_des, df_f_des], sort=False,
axis=1)
```

   Write the final data into an output file

```
df_out.to_csv('./output/question2_out.csv', index=False,
float_format='%g')
```