Queen Mary University of London

Department of Electrical Engineering & Computer Science

ECS766P Data Mining – Assignment 1

Hasan Emre Erdemoglu

27 October 2020

Table of Contents:

Part 1: Oct 6, 2020

Question 1.1:

In this question the following sales data is given:

$$[2, 15, 20, 5, 1, 4, 7, 9, 10\ 3, 14, 8]$$

The question asks to do apply equal frequency binning; then to smooth these bins by two different methods; by bin means and bin boundaries. While the scale of the question is easy to calculate by hand; for scalability, these operations are coded in Python. As there are 12 data points; there can be 1, 2, 3, 4, 6 and 12 binning options available; as question does not indicate how many bins are desired; it is assumed that the number of bins that is desired is 4. The code below can work with different bin levels given that BIN_WIDTH option returns to be integer, i.e. the data can be separated equally.

First, this data is created in a list and sorted using sort method of list class. After that, bin widths are found, and data is separated into bins. For smoothing via bin mean, mean of the bin is found and everything is set to this value. For smoothing by bin boundaries, the difference of the value to the bin boundary is calculated and the value is assigned to the closest boundary. If distances are equal, larger number was favored.

The output can be found below where the code generating this output can be found in the appendix.

```
Given data:  [2, 15, 20, 5, 1, 4, 7, 9, 10, 3, 14, 8]
Sorted data:  [1, 2, 3, 4, 5, 7, 8, 9, 10, 14, 15, 20]

Data Divided into 4 Bins:
[[ 1.  2.  3.]
 [ 4.  5.  7.]
 [ 8.  9. 10.]
 [14. 15. 20.]]

Smoothing by bin means:
[[ 2.          2.          2.        ]
 [ 5.33333333  5.33333333  5.33333333]
 [ 9.          9.          9.        ]
 [16.33333333 16.33333333 16.33333333]]
[ 2.          2.          2.          5.33333333  5.33333333  5.33333333
  9.          9.          9.         16.33333333 16.33333333 16.33333333]

Smoothing by bin boundaries:
[[ 1.  3.  3.]
 [ 4.  4.  7.]
 [ 8. 10. 10.]
 [14. 14. 20.]]
[ 1.  3.  3.  4.  4.  7.  8. 10. 10. 14. 14. 20.]
```

Figure 1. Answers to Question 1

Question 1.2:

The question gives data points $[10, 20, 35, 70, 100]$ and asks for the implementation of following normalization techniques:

Min-Max Normalization (with min=0 & max=1):

For this normalization, the following code is used. It implements the formula also given below:

```
# Min-max normalization with min = 0, max = 1:
MIN_Q2 = min(q2_data)
MAX_Q2 = max(q2_data)
NEW_MIN = 0
NEW_MAX = 1
min_max_data = []
for idx,value in enumerate(q2_data):
    res = (value-MIN_Q2)/(MAX_Q2-MIN_Q2) * (NEW_MAX-NEW_MIN) + NEW_MIN
    min_max_data.append(res)
print('Min-Max Normalization Scores:')
print(min_max_data)
print()
```

$$v_{normalized} = \frac{v_{current} - v_{min}}{v_{max} - v_{min}} * (\max_{new} - min_{new}) + \ min_{new}$$

This code segment prints out the following score:

```
Min-Max Normalization Scores:
[0.0, 0.1111111111111111, 0.2777777777777778, 0.6666666666666666, 1.0]
```

Figure 2. Min-Max Normalization

As expected, all values between 10 and 100 are mapped between 0 and 1.

Z-Score Normalization:

Z-score normalization removes the mean of the data and divides by the standard deviation of the data. The code and the formula are given below:

```
MEAN_Q2 = np.mean(q2_data)
STD_Q2 = np.std(q2_data)
z_score_data = []
for idx,value in enumerate(q2_data):
    res = (value-MEAN_Q2)/STD_Q2
    z_score_data.append(res)
print('Z-Scores:')
print(z_score_data)
```

$$z = \frac{(v - \mu_v)}{\sigma_v}$$

This code segment prints out the following score:

```
Z-Scores:
[-1.1075660291651117, -0.808223859120487, -0.35921060405354976, 0.6884869911026371,
1.5865135012365115]
```

Figure 3. Z-Score Normalization

Normalization with Decimal Scaling:

Decimal scaling normalization follows the following formula:

$$v_{norm} = \frac{v}{10^j}$$

For this question scaling factor j is equal to 2. Normally j can be picked automatically to satisfy the smallest integer which makes the maximum value of the data less or equal to 1. The code and the output can be found below:

```
SCALE_FACTOR = 2

dec_data = []

for idx,value in enumerate(q2_data):

    res = value / 10 ** SCALE_FACTOR

    dec_data.append(res)

print('Decimal Scaling Normalization Scores:')

print(dec_data)

print()
```

```
Decimal Scaling Normalization Scores:
[0.1, 0.2, 0.35, 0.7, 1.0]
```

Figure 4. Decimal Scaling Normalization

Question 1.3:

Given the data, the question requires to calculate the correlation between a person's 'Age' and 'BMI'. Correlation coefficient is calculated with the following formula:

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \mu_A)(b_i - \mu_B)}{n\sigma_A\sigma_B}$$

The following method defines this equation. Then, given data is fed through this method.

```
def calc_correlation(data):
    mean_age = np.mean(data[:,0])
    mean_bmi = np.mean(data[:,1])
    std_age = np.std(data[:,0])
    std_bmi = np.std(data[:,1])
    no_samples = data.shape[0]
    # Calculate & return r:
    return np.sum((data[:,0]-mean_age) * (data[:,1]-mean_bmi)) / (no_samples * std_age * std_bmi)
```

The correlation is found to be 0.891, in three significant figures; indicating a positive correlation between Age and BMI; for completeness, the scatter plot of the data samples is also given. It also indicates positive correlation.

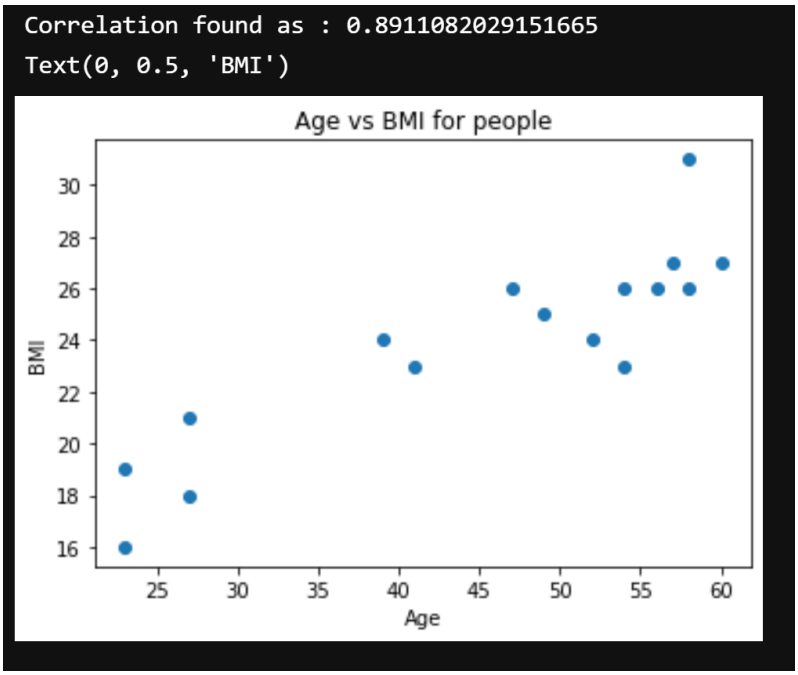


Figure 5. The correlation output for Age & BMI and scatter plot of given samples

Question 1.4:

The question is to understand whether *patient satisfaction is correlated with a specific hospital.* The question asks to use Chi-square test to test this hypothesis assuming a significance level of 0.001 (which corresponds to chi-square value of 10.828). The following table is given as input:

Rating/Hospital	**Hospital A**	**Hospital B**	***Total***
Satisfied	71	129	***200***
Dissatisfied	37	73	***110***
Total	***108***	***202***	***310***

Chi-square test has the following equation, where o and e stand for observed and expected frequency terms respectively:

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

To be able to calculate Chi-square values; expected frequencies must be calculated. Expected frequencies can be calculated by the following formula:

$$e_{ij} = \frac{count(A = a_i)\ count(B = b_j)}{n}$$

From these equations the following result is found. The associated code for this question can be found in the appendix. The code simply calculates expected frequencies and use them to calculate chi squared score.

```
Calculated chi score:  16.861050688402173
The premise -patient satisfaction with hospitals are independent- is False.
```

Figure 6. Calculated Chi Score.

As calculated chi score is greater than the value 10.828; with significance level 0.001; we must reject the hypothesis that patient satisfaction is independent with the hospitals. In fact, if looked closely patient satisfaction is statistically prominent with hospital B (there are a smaller number of dissatisfied patients than expected in hospital B). The following list shows individual scores for chi values, to prove this point:

```
Chi hospital A satisfaction score:  0.025104540023895054
Chi hospital B satisfaction score:  0.013422229319705 98
Chi hospital A dissatisfaction score:  0.045644618225263243
Chi hospital B dissatisfaction score:  16.776879300833308
```

Figure 7. Individual contributions to Chi Score

Question 1.5:

This question asks pre-processing of British Library Books dataset. The following operations are requested:

1. Inspect and describe issues with data
2. Remove fields which are used for internal processes at the British library (Corporate Author, Corporate Contributors, Issuance type, Former Owner, Shelf marks, Engraver)
3. Data cleaning on Data of Publication column.

Upon inspection it is seen that many fields in Edition statement and some in Publisher and Author column are empty. There are typos and unwanted characters or information present in Place of Publication, Data of Publication, Title fields. Corporate Authors, Corporate Contributors, Former owner, and Engraver fields are completely empty. There are 8287 book samples in total in which there might be duplicates which might be not detectable due to possible typos in different attributes of the book. However, identifier field seems to provide a unique key for each book.

For item 2,; pd.drop(columns_to_drop_list, inplace=True) is used to drop necessary columns from the data frame in place. After this operation, 8287 samples will be available and requested columns will be deleted. The code for this section can be found in Appendix with additional explanation. The output is as follows:

Identifier	Edition Statement	Place of Publication	Date of Publication	Publisher	Title	Author	Contributors	Flickr URL
206	NaN	London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	FORBES, Walter.	http://www.flickr.com/photos/britishlibrary/ta...
216	NaN	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	BLAZE DE BURY, Marie Pauline Rose - Baroness	http://www.flickr.com/photos/britishlibrary/ta...
218	NaN	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of "All for Gr...	A., A. A.	BLAZE DE BURY, Marie Pauline Rose - Baroness	http://www.flickr.com/photos/britishlibrary/ta...
472	NaN	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	Appleyard, Ernest Silvanus.	http://www.flickr.com/photos/britishlibrary/ta...
480	A new edition, revised, etc.	London	1857	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	BROOME, John Henry.	http://www.flickr.com/photos/britishlibrary/ta...

Figure 8. Removal of Library Related Attributes

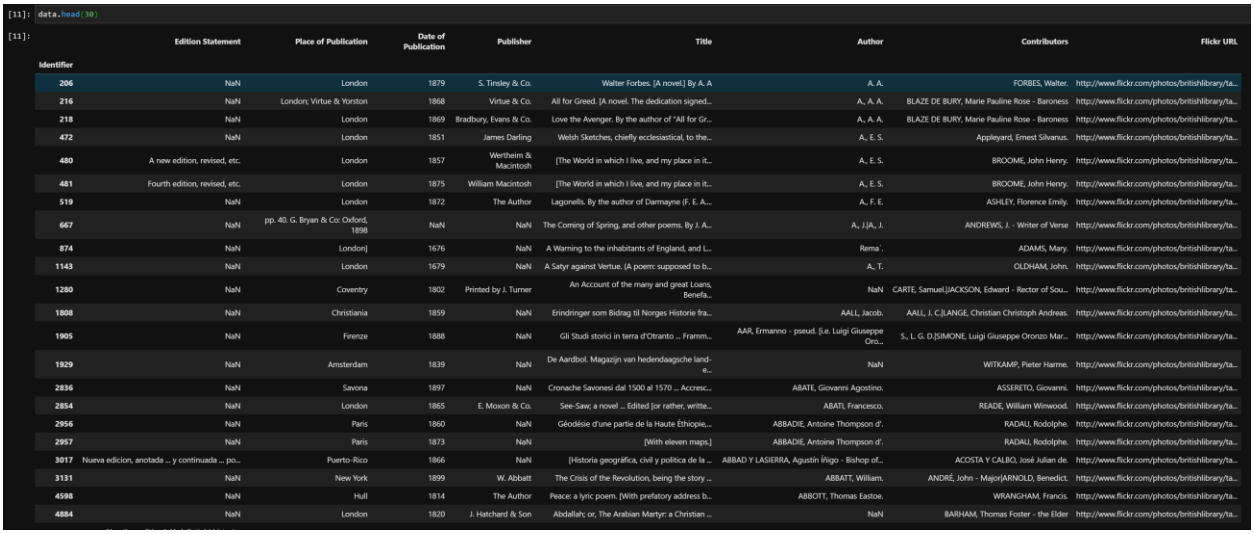
Upon inspection it is seen that there are many NaN values in edition statements attribute. These values are empty in 'BL-books.csv' file. Removing all NaN values, reduces the number of samples to 326 by inspection. Another approach is to drop edition statement column altogether; then using pd.dropna() to eliminate NaN values. Using this there are 3062 samples remaining, down from 8287. As pre-processing on these columns are not requested by the question; these NaN values will be kept as is, although the dataset can make use of further pre-processing and cleaning steps for all of the columns.

For item 3, data cleaning will be applied on Date of Publication column. There are many different cases (e.g. NaN values, dates separated by comma, square brackets, dates with 2 characters, dates separated by dashes, dates with question marks; commas, dots; dates with incomplete brackets or a combination of these) available in the column and there are 2 rules two achieve:

1. Remove dates in the square brackets.
2. Convert dates to their start date wherever present.

A close observation on Date of Publication forum show that other than very rare scenarios such as Identifier=1929; where entry is 1839 38-54 there are no scenarios where the first 4-digit date is later than any other date specified. If this is not ignored a complex series of functions should be written and data must be passed through many times to ensure everything is in order. Assuming these samples do not exist; it is sufficient to use the first 4-digit year value as the Date of Publication.

The code for this question can be found in the Appendix. The following is the result after cleaning 'Date of Publication' attribute.



```
[11]: data.head(30)
```

[11]:

Identifier	Edition Statement	Place of Publication	Date of Publication	Publisher	Title	Author	Contributors	Flickr URL
206	NaN	London	1879	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	FORBES, Walter.	http://www.flickr.com/photos/britishlibrary/ta...
216	NaN	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	BLAZE DE BURY, Marie Pauline Rose - Baroness	http://www.flickr.com/photos/britishlibrary/ta...
218	NaN	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of "All for Gr...	A., A. A.	BLAZE DE BURY, Marie Pauline Rose - Baroness	http://www.flickr.com/photos/britishlibrary/ta...
472	NaN	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	Appleyard, Ernest Silvanus.	http://www.flickr.com/photos/britishlibrary/ta...
480	A new edition, revised, etc.	London	1857	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	BROOME, John Henry.	http://www.flickr.com/photos/britishlibrary/ta...
481	Fourth edition, revised, etc.	London	1875	William Macintosh	[The World in which I live, and my place in it...	A., E. S.	BROOME, John Henry.	http://www.flickr.com/photos/britishlibrary/ta...
519	NaN	London	1872	The Author	Lagonells. By the author of Darmayne (F. E. A....	A., F. E.	ASHLEY, Florence Emily.	http://www.flickr.com/photos/britishlibrary/ta...
667	NaN	pp. 40. G. Bryan & Co: Oxford, 1898	NaN	NaN	The Coming of Spring, and other poems. By J. A...	A., J.[A., J.	ANDREWS, J. - Writer of Verse	http://www.flickr.com/photos/britishlibrary/ta...
874	NaN	London]	1676	NaN	A Warning to the inhabitants of England, and L...	Rema`.	ADAMS, Mary.	http://www.flickr.com/photos/britishlibrary/ta...
1143	NaN	London	1679	NaN	A Satyr against Vertue. [A poem: supposed to b...	A., T.	OLDHAM, John.	http://www.flickr.com/photos/britishlibrary/ta...
1280	NaN	Coventry	1802	Printed by J. Turner	An Account of the many and great Loans, Benefa...	NaN	CARTE, Samuel.]JACKSON, Edward - Rector of Sou...	http://www.flickr.com/photos/britishlibrary/ta...
1808	NaN	Christiania	1859	NaN	Erindringer som Bidrag til Norges Historie fra...	AALL, Jacob.	AALL, J. C.]LANGE, Christian Christoph Andreas.	http://www.flickr.com/photos/britishlibrary/ta...
1905	NaN	Firenze	1888	NaN	Gli Studi storici in terra d'Otranto ... Framm...	AAR, Ermanno - pseud. [i.e. Luigi Giuseppe Oro...	S., L. G. D.]SIMONE, Luigi Giuseppe Oronzo Mar...	http://www.flickr.com/photos/britishlibrary/ta...
1929	NaN	Amsterdam	1839	NaN	De Aardbol. Magazijn van hedendaagsche land-e...	NaN	WITKAMP, Pieter Harme.	http://www.flickr.com/photos/britishlibrary/ta...
2836	NaN	Savona	1897	NaN	Cronache Savonesi dal 1500 al 1570 ... Accresc...	ABATE, Giovanni Agostino.	ASSERETO, Giovanni.	http://www.flickr.com/photos/britishlibrary/ta...
2854	NaN	London	1865	E. Moxon & Co.	See-Saw; a novel ... Edited [or rather, writte...	ABATI, Francesco.	READE, William Winwood.	http://www.flickr.com/photos/britishlibrary/ta...
2956	NaN	Paris	1860	NaN	Géodésie d'une partie de la Haute Éthiopie,...	ABBADIE, Antoine Thompson d'.	RADAU, Rodolphe.	http://www.flickr.com/photos/britishlibrary/ta...
2957	NaN	Paris	1873	NaN	[With eleven maps.]	ABBADIE, Antoine Thompson d'.	RADAU, Rodolphe.	http://www.flickr.com/photos/britishlibrary/ta...
3017	Nueva edicion, anotada ... y continuada ... po...	Puerto-Rico	1866	NaN	[Historia geográfica, civil y política de la ...	ABBAD Y LASIERRA, Agustín Íñigo - Bishop of...	ACOSTA Y CALBO, José Julian de.	http://www.flickr.com/photos/britishlibrary/ta...
3131	NaN	New York	1899	W. Abbatt	The Crisis of the Revolution, being the story ...	ABBATT, William.	ANDRÉ, John - Major]ARNOLD, Benedict.	http://www.flickr.com/photos/britishlibrary/ta...
4598	NaN	Hull	1814	The Author	Peace: a lyric poem. [With prefatory address b...	ABBOTT, Thomas Eastoe.	WRANGHAM, Francis.	http://www.flickr.com/photos/britishlibrary/ta...
4884	NaN	London	1820	J. Hatchard & Son	Abdallah; or, The Arabian Martyr: a Christian ...	NaN	BARHAM, Thomas Foster - the Elder	http://www.flickr.com/photos/britishlibrary/ta...

Figure 9. Data frame after pre-processing

Question 1.6:

For this question Age and Income data must be imputed with mean of the corresponding attribute of the dataset. Later the categorical attributes (‘Online Shopper’ and “Region’) must be converted to numerical attributes.

NaN values can be easily filled by using fillna(…) method of the data frame. As parameter mean() function of the dataframe can be used. The following method fully fills NaN values and outputs the following:

```
data = data.fillna(data.mean())
```

	Region	Age	Income	Online Shopper
0	India	49.000000	86400.000000	No
1	Brazil	32.000000	57600.000000	Yes
2	USA	35.000000	64800.000000	No
3	Brazil	43.000000	73200.000000	No
4	USA	45.000000	76533.333333	Yes
5	India	40.000000	69600.000000	Yes
6	Brazil	43.777778	62400.000000	No
7	India	53.000000	94800.000000	Yes
8	USA	55.000000	99600.000000	No
9	India	42.000000	80400.000000	Yes

Figure 10. Dataset after filling NaN values with mean of the corresponding attributes

Since there are 2 classes only, the following lambda expression can be used to convert categorical attributes to numerical attributes. Note that this is only one way of doing it; for if multiple regions were available other methods might be needed to solve this issue; such as using a dict, etc. The following code is used to achieve these results:

```
data['Online Shopper'] = data['Online Shopper'].apply(lambda val: 0 if val=='No' else 1)

data['Region'] = data['Region'].apply(lambda val: 1 if val=='India' else 2 if val=="USA" else 3)
```

The result is as follows:

	Region	Age	Income	Online Shopper
0	1	49.000000	86400.000000	0
1	3	32.000000	57600.000000	1
2	2	35.000000	64800.000000	0
3	3	43.000000	73200.000000	0
4	2	45.000000	76533.333333	1
5	1	40.000000	69600.000000	1
6	3	43.777778	62400.000000	0
7	1	53.000000	94800.000000	1
8	2	55.000000	99600.000000	0
9	1	42.000000	80400.000000	1

Figure 11. The result after imputation and replacing categorical attributes to numerical attributes.

Question 1.7:

The question asks the following:

1. Plot Separate Scatter Plots of Shoe size vs Height for males and females separately.
2. Compute Pearson's Correlation Coefficient and compare the results with the scatterplots obtained.

After loading data using pandas, the following code segment is used to plot the scatterplots. The outputs are shown below.

```
import matplotlib.pyplot as plt

# Plots are asked to be drawn seperately!

print(data['Size'])

# Plot the scatterplot shoesize vs height - MALE: Filter first, then plot:

data_male = data[data['Gender']=='M']

plt.figure(0)

plt.scatter(data_male['Size'], data_male['Height'])

plt.title("Shoe size vs Height for Males")

plt.xlabel("Shoe size")

plt.ylabel("Height")


# Plot the scatterplot shoesize vs height - FEMALE:

data_female = data[data['Gender']=='F']

plt.figure(1)

plt.scatter(data_female['Size'], data_female['Height'])

plt.title("Shoe size vs Height for Females")

plt.xlabel("Shoe size")

plt.ylabel("Height")
```

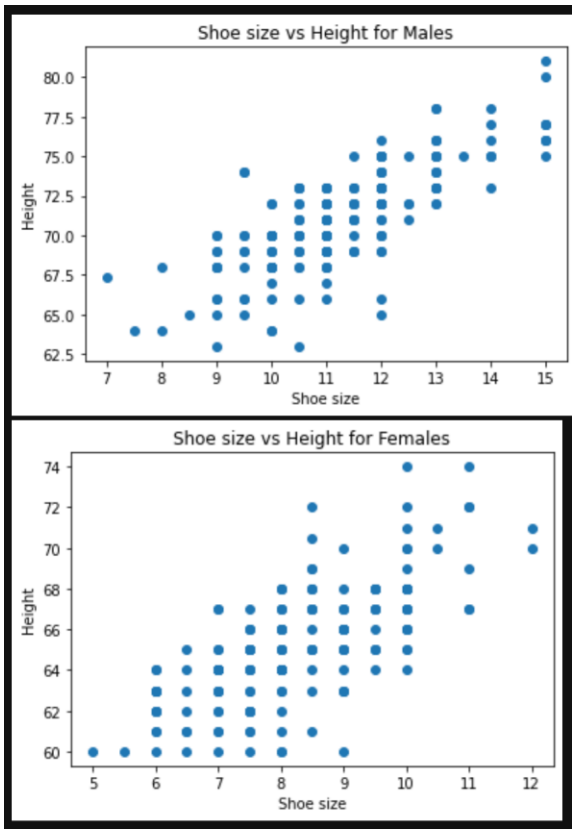


Figure 12. Shoe size vs height scatter plots for Males & Females

The pearson's correlation coefficient can be calculated easily however the question allows use of stats library. The following code and the results are found.

```
# Compute Pearson's Correlation Coefficient:

from scipy import stats

# Male stats:

r_male, p_male = stats.pearsonr(data_male['Size'], data_male['Height'])

# Female Stats:

r_female, p_female = stats.pearsonr(data_female['Size'], data_female['Height'])

# Results:

print('Pearson Corr for Males: ', r_male)

print('Pearson Corr for Females: ', r_female)
```

```
Pearson Corr for Males:  0.7677093547300977
Pearson Corr for Females:  0.707811941714397
```

Figure 13. Pearson Correlation Measurements for Males and Females with respect to Height and Shoe Size

The results indicate that the size of a person and their shoe sizes are positively correlated. The same result is also reflected on the scatter plots. The correlation is strong but not very strong however, implying that the relationship between height and shoe size can vary to some extent.

Question 1.8:

This question uses Breast Cancer Dataset from Section 1 of the laboratory work. The questions ask to calculate PCA with 2 components: computing the explained variance ratio for each component. It also requires doing the scatterplot of all samples along principal components color coded according to the class they belong in.

The process of the solution is in the following order:

1. Data loading and dropping unrelated columns ('Sample code')
2. Replacing '?' terms with NaN, removing samples which don't have data available.
3. Changing 'Bare Nuclei' attribute to numeric to allow computations to happen.
4. Dropping duplicate samples
5. Separating class labels from actual data
6. Z-Normalization on data (Not the classes)
7. Initializing PCA on 2 components
8. Analysis of Results

Note that parts 1-6 follow from Section 1; as the same dataset was used in both example and this question. This report will focus on Part 7 & 8, which constructs the core part of this question.

The code shown below uses sklearn library to utilize PCA. PCA computes the eigenvalues and its corresponding eigenvectors of the covariance matrix of the attributes to calculate variance across all attribute dimensions. As there are 9 dimensions; ideally a 9-dimensional eigenspace will fully contain all the information/variance retained in the dataset. PCA, in this case, picks best 2 dimensions (the eigenvectors that have the highest eigenvalues) out from this 9-dimensional space; projecting other 7 dimension on these two dimensions. However, this projection operation removes other eigen-dimensions from equation; therefore, the variance explained by these eigenvectors are not present in the lower dimensional space. Therefore, PCA reduces dimensionality of a 9-attribute system into two eigenvectors, which can be visualized more efficiently.

The core below first generates a PCA object then fits the transform calculating necessary covariance matrix of the dataset and its corresponding eigenpairs; it also picks the best two eigenvalues. This is done by fit_transform(Z) method. After that these eigenpairs are used to transform 9-dimensional dataset space to a 2-dimensional space which are named as pc1 and pc2. Finally, this transformation is logged in a data frame and the associated class labels are appended to the dataframe.

```
import pandas as pd

from sklearn.decomposition import PCA

# PCA parameters:

PCA_COMP_SIZE = 2

pca = PCA(n_components=PCA_COMP_SIZE)

# Normalize Data - CLASS SHOULD NOT BE USED:

pca.fit_transform(Z)
```

```
projected = pca.transform(Z)

projected = pd.DataFrame(projected,columns=['pc1','pc2'],index=range(1,Z.shape[0]+1))

projected['Class'] = list_class # works but check this probably indices are off.

projected
```

These are some of the results provided by this analysis:

	pc1	pc2	Class
1	-2.227261	-0.013771	2
2	0.453610	-0.450109	2
3	-2.347697	0.004607	2
4	0.557281	-0.312441	2
5	-2.120607	-0.004716	2
...	...	...	...
445	-2.457464	0.597246	2
446	-2.478106	0.279124	2
447	2.681721	0.114452	4
448	1.254999	-0.885602	4
449	1.629231	-0.980696	4

Figure 14. Breast Cancer Dataset after 2D transformation

Later this result is printed on a scatterplot:

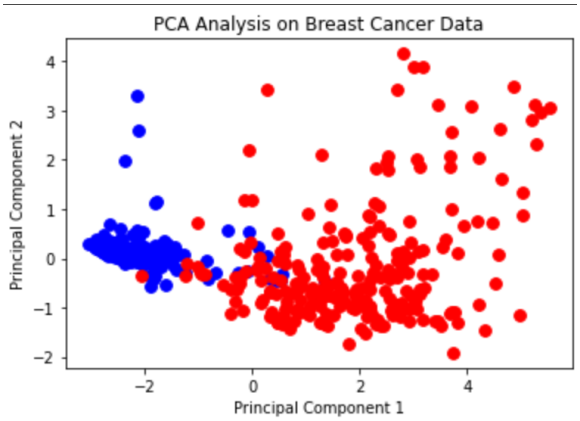


Figure 15. Scatterplot of samples using the principal components

```
Printing eigenvalues:  [5.33372679 0.85045277]
Variance explained by respective eigenvalue:  [0.59263631 0.09449475]
Total Explained Variance Percentage:  68.71310628589626 %.
```

Figure 16. Variance Explained by Eigenvalues

As seen from Figure 15, most of the data is distributed around principal component 1, which corresponds to the eigenvalue 5.33. This eigenvalue can explain 59.26 % of the collective variance of the 9-dimensional dataset by itself. Second principal component holds less samples and has value 0.85 and can explain 9.45 % of the variance by itself. The magnitude of the eigenvalue and the variance explained by that eigenvalue (and its corresponding eigenvector) is correlated.

Since eigenvalues are orthogonal the variance represent by these vectors are independent from each other. In total these components can represent 68.71 % of the knowledge retained by the original dataset, which means that 31.29 % of the variance/information is lost when the original 9-dimensional space is reduced to a 2-dimensional space. Each individual point on the scatter plot originally is described by a nine-dimensional orthogonal vector space, but in this case, they represented by best 2 orthogonal vectors only.

Even though a significant information is lost using this method; approximately 70% of the variance is sufficient to group these two classes from each other as seen in the scatterplot.

The code for this section can be found in the appendix.

Part 2: Oct 13, 2020

Question 2.1:

This question asks the relationship inferred from summary statistics regarding **ACT Composite Score** and **SAT Total Score**. The section 1.3 has the univariate statistics and correlation coefficients.

	ACT composite score	SAT total score	parental income	high school gpa	college gpa	years to graduate
ACT composite score	1.000000	0.885884	0.183879	0.874206	0.507349	-0.129880
SAT total score	0.885884	1.000000	0.247556	0.910425	0.518257	-0.125523
parental income	0.183879	0.247556	1.000000	0.227238	0.460863	-0.239500
high school gpa	0.874206	0.910425	0.227238	1.000000	0.492489	-0.119524
college gpa	0.507349	0.518257	0.460863	0.492489	1.000000	-0.467499
years to graduate	-0.129880	-0.125523	-0.239500	-0.119524	-0.467499	1.000000

Figure 17. Correlation Summary for Graduation Rate Dataset

From Figure 17, it is clearly visible that ACT composite score and SAT total score have a high positive correlation. This implies that having a high score from one test, implies that the other test will be high as well. The scatterplots on section 1.8 best describes the relationship between these scores visually.

The following code first plots the scatterplot which is only outputting the answer for ACT composite score and SAT total score (Figure 18); whereas pairplot() method complements Figure 17 by visualizing correlation for each pair of attributes (Figure 19):

```
sns.scatterplot(x='ACT composite score', y='SAT total score', data=df)

plt.show()

sns.pairplot(df)

plt.show()
```

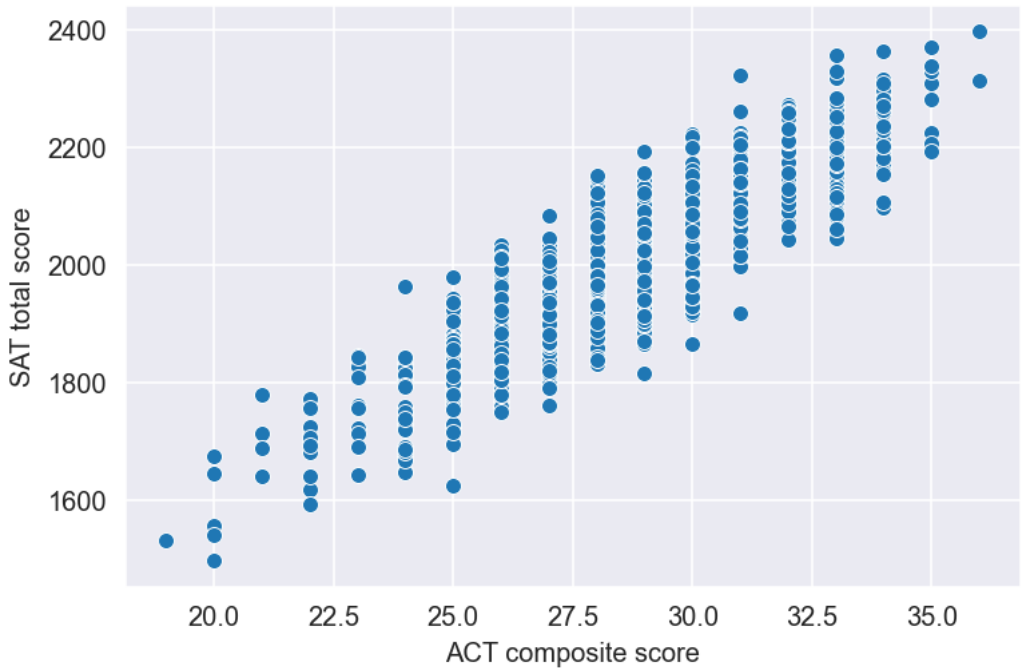


Figure 18. Scatterplot between ACT Composite Score vs SAT Total Score

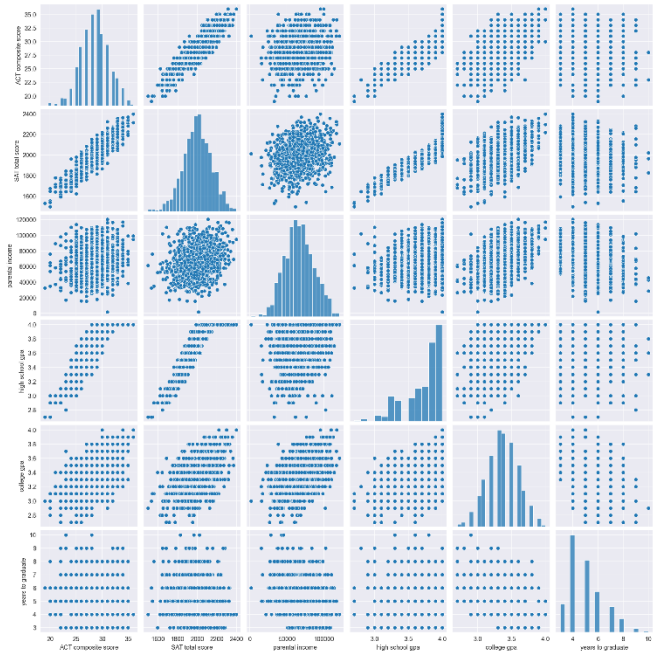


Figure 19. Complementary Graph Visualizing Data on Figure 17.

As expected, the pairs with no correlation such as SAT total score and Parental income does not show a correlation.

Question 2.2:

This question focuses on box plots of Parental level of education and Parental Income. The following is already given in the assignment.

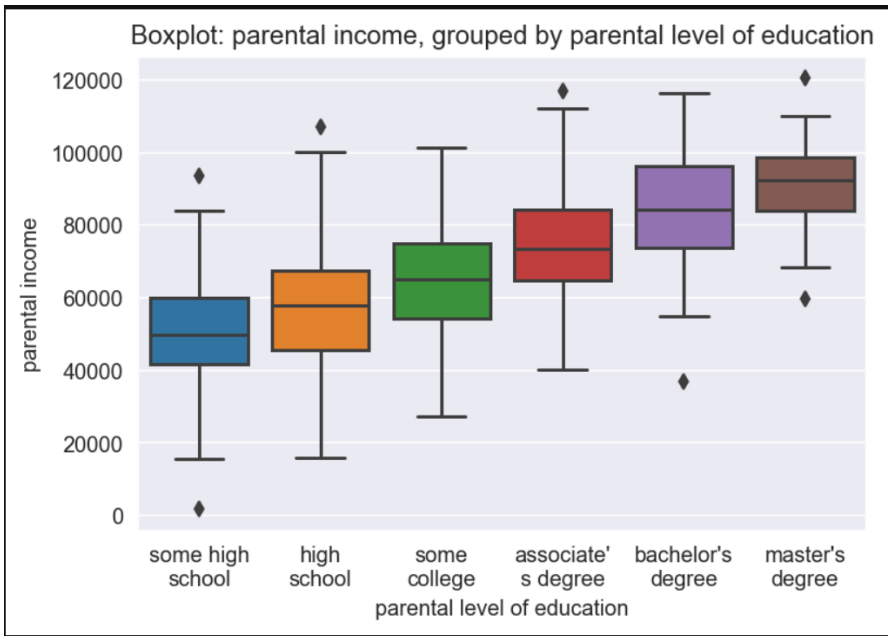


Figure 20. Box plot of Parental Income, grouped by parental level of education

Figure 20 shows that the mean parental income increases as the level of education increases. There is a tendency that the upper and lower quartile values of the income increase as well. Box plots provide, mean, first and third quartiles, interquartile range and maximal and minimal data (in this case it is 1.5 times IQR by default) along with outliers that are above/below maximum and minimum ranges respectively.

Regarding Parental income of students whose parent's have a master's degree; it seems that there are 2 outliers (one below minimum range and one above maximum range). These values can be exposed using table visualization.

First the dataset can be filtered by only showing the entries where Parental Level of Education is "Master's Degree". After that, two approaches can be followed. First, another filter can be applied to Parental Income attribute, which will exclude entries which will fall between the minimum and maximum ranges of the boxplot by the following formula: [Q1-1.5IQR, Q3+1.5IQR]. In the second approach, since its known that there are 2 outliers from the box plot; the Parental Income attribute can be sorted and maximum and minimum values can be extracted.

As first option delves into Outlier Detection which will be covered in Week 8, I will utilize the second approach. The code is given below:

```
dataframe = (df[df['parental level of education'] == "master's degree"].sort_values(by='parental income', ascending=False)).apply(lambda x: x.iloc[[0, -1]])

print('Sorting by parental income of students whose parents have a masters degree:')

display(dataframe)
```

Sorting by parental income of students whose parents have a masters degree:

	ACT composite score	SAT total score	parental level of education	parental income	high school gpa	college gpa	years to graduate
411	31	2108	master's degree	120391	4.0	3.6	4
420	28	2097	master's degree	59724	3.9	3.2	4

Figure 21. Outliers for Parental Income given Parental Level of Education is a MS Degree

Question 2.3:

If the features are not properly scaled, then the feature that has larger magnitudes may affect the distance calculations such that the feature where the distances are small will have less effect. This is not helpful as the distance metric would ignore the features which are small compared to the larger features. For a good inspection, the features should be comparable with each other. Most of the time; the features are normalized to zero mean and unit variance to make sure all features are comparable with respect to each other.

Let us assume the following arbitrary dataset, where feature 1 is small compared to feature 2. Assume that we are inspecting Euclidean distance between the samples. First, the distance matrix will be generated for the datapoints without any normalization. Then normalization will be applied, and the results will be compared.

Table 1. Arbitrary Data for Question 2.3

Sample #	1	2	3	4	5	6	7	8	9	10
Feature 1	**-8**	**-5**	**-2**	**-1**	**0**	**1**	**3**	**4**	**7**	**9**
Feature 2	**-46**	**-30**	**-15**	**-4**	**-1**	**10**	**23**	**36**	**38**	**55**
F1 (normalized)	**-1.78**	**-1.18**	**-0.56**	**-0.36**	**-0.16**	**0.04**	**0.44**	**0.64**	**1.25**	**1.66**
F2 (normalized)	**-1.73**	**-1.20**	**-0.71**	**-0.35**	**-0.25**	**0.11**	**0.54**	**0.96**	**1.06**	**1.58**

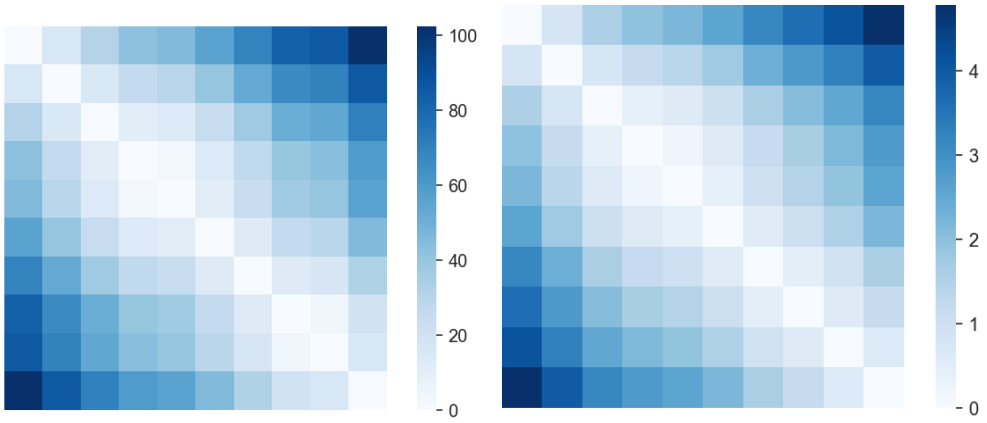


Figure 22. Unscaled Distance map (Left) vs. Scaled Distance Map (Right)

Although the colormap looks the same (as the samples are sorted and ordered, so that the initial data has some bias), when looked at the scale it is seen that the distance to sample 1 to sample 10 is very huge in un-scaled version of the distance matrix, dominated by the distance in feature 2. For scaled version, the distance between these datapoints are smaller, and both feature 1 and feature 2 effect the distance in a comparable way. For more dimensional cases, scaling becomes even more important as large unscaled features may prioritize one feature over the other features in applications such as PCA or clustering.

The code for this section can be found in the appendix.

Question 2.4:

The matrix visualization discussed by the question is given below. The figure shows the distance between a student with respect to another student sorted by their parent's education level, starting from some high school to master's degree. Since there are 1000 student samples available in the dataset, this distance matrix has a size of 1000 by 1000. Also note that the diagonal entries consist of a distance 0, as those entries compare the sample by itself. Therefore, for making conclusions from this matrix, looking either at lower or upper triangle is sufficient.

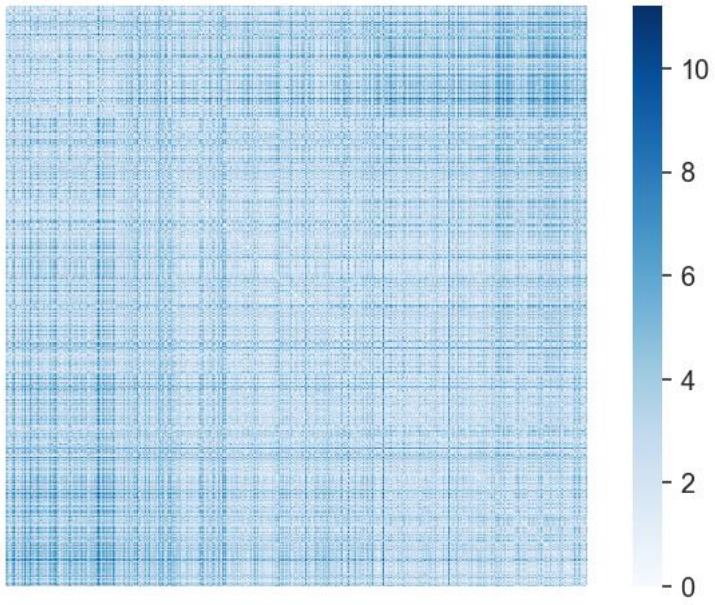


Figure 23. Distance Matrix between students sorted by parent's level of education.

The question states that "average distance between students whose parents only have some high school education and students whose parents have a master's degree is larger than the average distance between students whose parents only have some high school education."

This statement is logical, as intra class attributes would be closer to each other and inter class attributes would be further away from each other. Given that some high school education vs master's degree is in two ends of spectrum it is expected to see that there is a high distance between students of these classes. For students who belong to the same class (parent's education level) the samples would be more similar, hence showing less distance.

This premise is visible using the distance matrix as upper right and lower left corner which corresponds to distance between some high-school education vs master's degree and master's degree vs high-school education, respectively. It is clearly visible that the distance between these two classes (interclass distance) are more than distance between samples which are in the same class (intraclass).

Question 2.5:

This question asks the following:

> *"In Section 2, increase the number of evenly spaced numbers from 10 to 100 for both axes and observe the corresponding heat map created through nearest neighbor interpolation. Read about this interpolation method and explain what you observed."*

When the number of evenly spaced samples are increased to 100 for both axes (by changing the parameter in linspace(…)); there is 100 times increase in the resolution; therefore, when plotted the following effect will be observed:

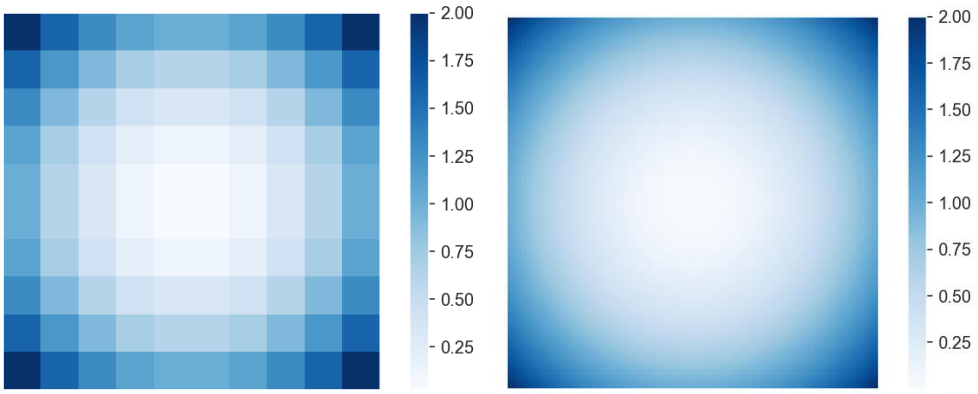


Figure 24. The formula $z = x^2 + y^2$ sampled by 100 times (Left), sampled by 10000 times (Right) – Nearest Interpolation

Nearest interpolation interpolates its proximal region with a single point. In the case of Figure 24, where it is used; nearest neighbor interpolation uses 100 points to approximate the function space. For each of the points, the pixels around that point is assigned to the approximating point's value.

For example for the first sample, (x,y) values change from -1 to -0.77 for both x and y. All (x,y) pairs within this range is approximated by (-1,-1) point. So, for each square tile, the function uses upper left corner coordinates to approximate the function.

As more samples are taken from linear space; a more refined image can be drawn, as seen on Figure 24 with the image on the right. Note that with enough samples, the image on the right looks very similar to Bilinear interpolation.

For the bilinear interpolation, two linear interpolations on x and y directions are made one after the other. As points are simply connected to each other in both axes; more sample point can be generated from bilinear interpolation hence more refined results can be achieved using less sample points. This is proven with the experiments as well.

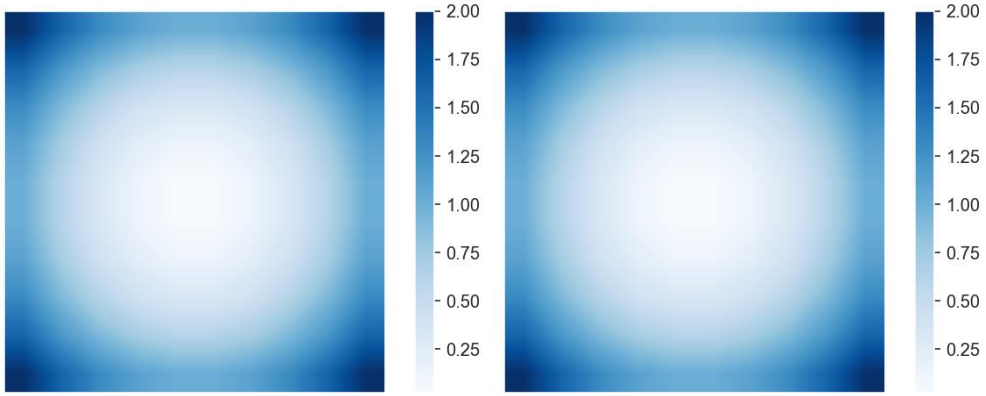


Figure 25. The formula $z = x^2 + y^2$ sampled by 100 times (Left), sampled by 10000 times (Right) – Bilinear Interpolation

Looking at Figure 25, Bilinear Interpolation is able to achieve similar performance with just 100 sample points from the function. A similar result can be achieved using nearest neighbor interpolation using 100 times more samples.

Question 2.6:

Question 2.6.1:

The dataset can be loaded using the following code segment. First 5 entries are shown below as an example. Frequencies for the categorical feature is given. Note that the **target** attribute is categorical, and the remaining attributes are numerical.

```
from sklearn.datasets import load_wine

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# 6.1. Load the wine dataset. Which feature is categorical? Compute the frequency of each value of the categorical feature.

# Load scikit dataset into dataframe - Target is categorical rest is numerical.

wine_data = load_wine()

dq6 = pd.DataFrame(data=wine_data.data, columns=wine_data.feature_names)

dq6['target'] = pd.Series(wine_data.target)

dq6.head()


# Compute the frequency of each value of the categorical feature.

freqs = dq6['target'].value_counts()/len(dq6)

display(freqs)
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	target
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0

```
Categorical Feature Frequencies: (Class - Frequency)
1    0.398876
0    0.331461
2    0.269663
Name: target, dtype: float64
```

Figure 26. First 5 entries of Wine Dataset & Target Frequencies

Question 2.6.2:

The following code segment deals with what this part of the question asks. First it computes univariate summaries for all numerical features. Then multivariate features are printed out and displayed in a graphical way as well. Later observations are grouped by categorical feature and median values are shown in tabular form.

```
# Compute Univariate and multivariate summaries for all numerical features:

print('Univariate summaries:')

display(dq6.loc[:,'alcohol':'proline'].describe())


print('Multivariate summaries')

print("\nCorrelation coefficients:")

display(dq6.corr())


print('Display multivariate summary in matrix form:')

ax = plt.axes()

sns.heatmap(dq6.corr(), ax = ax)


ax.set_title('Multivariate summary in matrix form')

plt.show()


#  Group observations by the categorical feature and compute the corresponding **median** for each remaining numerical feature.

display(dq6.groupby('target').median())
```

Univariate summaries:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	5.058090	0.957449	2.611685	746.893258
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318286	0.228572	0.709990	314.907474
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280000	0.480000	1.270000	278.000000
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220000	0.782500	1.937500	500.500000
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690000	0.965000	2.780000	673.500000
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200000	1.120000	3.170000	985.000000
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000000	1.710000	4.000000	1680.000000

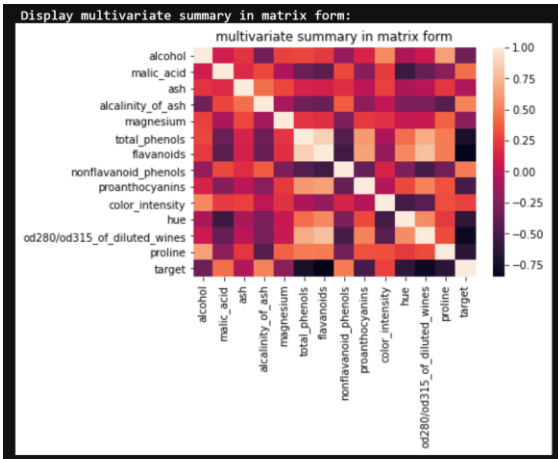
Multivariate summaries

Correlation coefficients:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	target
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.289101	0.236815	-0.155929	0.136698	0.546364	-0.071747	0.072343	0.643720	-0.328222
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	-0.335167	-0.411007	0.292977	-0.220746	0.248985	-0.561296	-0.368710	-0.192011	0.437776
ash	0.211545	0.164045	1.000000	0.443367	0.286587	0.128980	0.115077	0.186230	0.009652	0.258887	-0.074667	0.003911	0.223626	-0.049643
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	-0.321113	-0.351370	0.361922	-0.197327	0.018732	-0.273955	-0.276769	-0.440597	0.517859
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	0.214401	0.195784	-0.256294	0.236441	0.199950	0.055398	0.066004	0.393351	-0.209179
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	1.000000	0.864564	-0.449935	0.612413	-0.055136	0.433681	0.699949	0.498115	-0.719163
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	0.864564	1.000000	-0.537900	0.652692	-0.172379	0.543479	0.787194	0.494193	-0.847498
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	-0.449935	-0.537900	1.000000	-0.365845	0.139057	-0.262640	-0.503270	-0.311385	0.489109
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	0.612413	0.652692	-0.365845	1.000000	-0.025250	0.295544	0.519067	0.330417	-0.499130
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	-0.055136	-0.172379	0.139057	-0.025250	1.000000	-0.521813	-0.428815	0.316100	0.265668
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	0.433681	0.543479	-0.262640	0.295544	-0.521813	1.000000	0.565468	0.236183	-0.617369
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	0.699949	0.787194	-0.503270	0.519067	-0.428815	0.565468	1.000000	0.312761	-0.788230
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	0.498115	0.494193	-0.311385	0.330417	0.316100	0.236183	0.312761	1.000000	-0.633717
target	-0.328222	0.437776	-0.049643	0.517859	-0.209179	-0.719163	-0.847498	0.489109	-0.499130	0.265668	-0.617369	-0.788230	-0.633717	1.000000

Figure 27. Univariate and Multivariate Summaries

Display multivariate summary in matrix form:



target	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline
0	13.750	1.770	2.44	16.8	104.0	2.800	2.980	0.29	1.870	5.40	1.070	3.17	1095.0
1	12.290	1.610	2.24	20.0	88.0	2.200	2.030	0.37	1.610	2.90	1.040	2.83	495.0
2	13.165	3.265	2.38	21.0	97.0	1.635	0.685	0.47	1.105	7.55	0.665	1.66	627.5

Figure 28. Multivariate Summaries in Graphical form & median values of numerical attributes grouped by categorical attribute

Question 2.6.3:

The following code segment shows desired output:

```
# 6.3 Group observations by categorical feature and create one box plot of ``alcohol`` for each group.
ax = sns.boxplot(x='target', y='alcohol', data=dq6)
plt.title('Alcohol groupped by Target')

# Wrap xticks
import textwrap
ax.set_xticklabels([textwrap.fill(t.get_text(), 10)  for t in ax.get_xticklabels()])

plt.show()
```

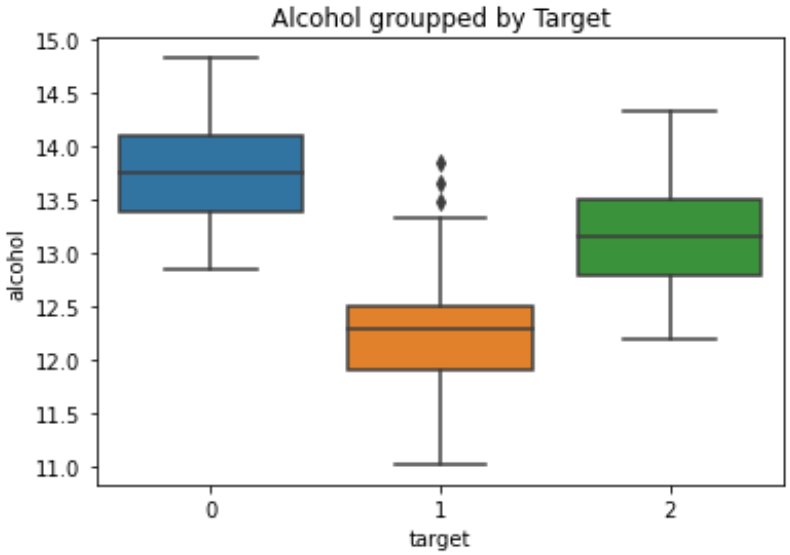


Figure 29. Box plot of Alcohol levels, grouped by Categorical ("Target") attribute

Question 2.6.4:

The question asks to create the scatter plot for the features with the highest correlation, excluding correlation with itself which is naturally equal to 1. The following code achieves the results.

```
# 6.4. Create a scatter plot for the pair of **distinct** numerical features with the highest correlation.

corrs = dq6.loc[:,'alcohol':'proline'].corr()

display(corrs)

# Filter best correlation:

corr_np = corrs.to_numpy() # get numpy version

corr_np = corr_np ** 2 #fix minus terms

corr_np[corr_np == 1] = 0 # exclude non distinct term

max_val = corr_np.argmax() # gives out flattened result

indices = np.unravel_index(max_val, corrs.shape) # unravel flattened result to normal array

print('Max corr located at:', str(indices)) # show max location

col_term = dq6.iloc[:,indices[1]]

row_term = dq6.iloc[:, indices[0]]

#print(col_term.shape)

#print(row_term.shape)

sns.scatterplot(x = row_term, y = col_term)

# display(sns.pairplot(dq6)) # display all features
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.289101	0.236815	-0.155929	0.136698	0.546364	-0.071747	0.072343	0.643720
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	-0.335167	-0.411007	0.292977	-0.220746	0.248985	-0.561296	-0.368710	-0.192011
ash	0.211545	0.164045	1.000000	0.443367	0.286587	0.128980	0.115077	0.186230	0.009652	0.258887	-0.074667	0.003911	0.223626
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	-0.321113	-0.351370	0.361922	-0.197327	0.018732	-0.273955	-0.276769	-0.440597
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	0.214401	0.195784	-0.256294	0.236441	0.199950	0.055398	0.066004	0.393351
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	1.000000	0.864564	-0.449935	0.612413	-0.055136	0.433681	0.699949	0.498115
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	0.864564	1.000000	-0.537900	0.652692	-0.172379	0.543479	0.787194	0.494193
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	-0.449935	-0.537900	1.000000	-0.365845	0.139057	-0.262640	-0.503270	-0.311385
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	0.612413	0.652692	-0.365845	1.000000	-0.025250	0.295544	0.519067	0.330417
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	-0.055136	-0.172379	0.139057	-0.025250	1.000000	-0.521813	-0.428815	0.316100
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	0.433681	0.543479	-0.262640	0.295544	-0.521813	1.000000	0.565468	0.236183
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	0.699949	0.787194	-0.503270	0.519067	-0.428815	0.565468	1.000000	0.312761
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	0.498115	0.494193	-0.311385	0.330417	0.316100	0.236183	0.312761	1.000000

```
Max corr located at: (5, 6)
<AxesSubplot:xlabel='total_phenols', ylabel='flavanoids'>
```

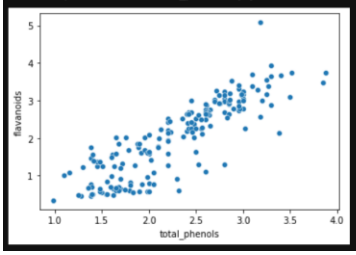


Figure 30. The output of the code segment given above

From the figure, it is indeed that max correlation is found at index (5,6) with value 0.86. The operations to achieve this result is written as in-line comments in the code above.

The final commented line displays scatterplot pairs of all features as an overview. The heatmap of correlations are also given with Figure 28, under Question 2.6.3. It is indeed that the scatterplot at index (5,6) = (total_phenols, flavonoids) has the highest correlation. Same result can also be confirmed with the heatmap given in Figure 28.

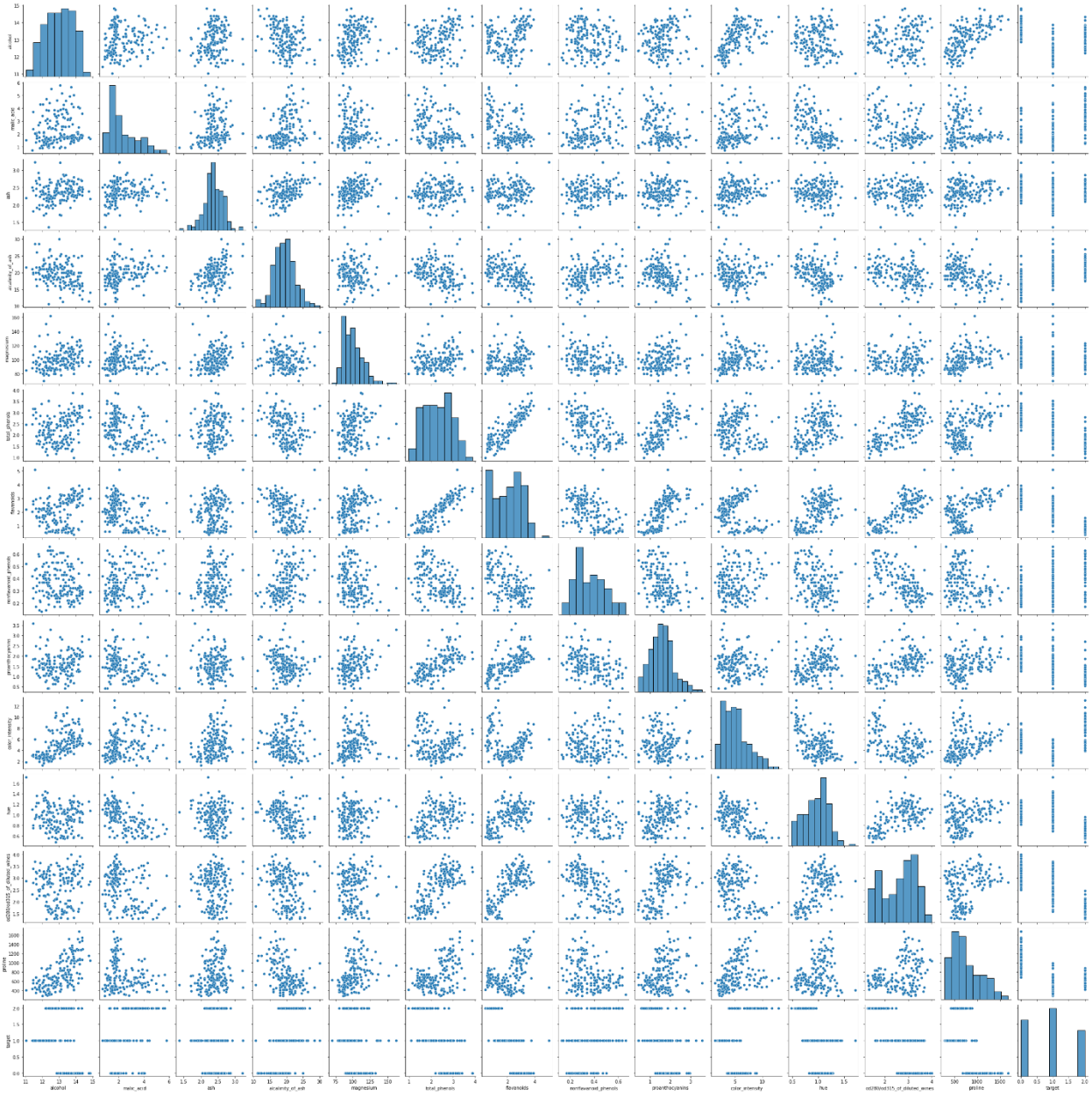


Figure 31. Pair plot for the attributes of Wine Dataset

Question 2.6.5:

The following code describes what is requested by the question. The algorithm is explained within inline comments.

```
# 6.5. Exclude the categorical feature, standardize the numerical features, and display a projection obtained by multidimensional scaling. Color the points by the categorical feature.

from sklearn.manifold import MDS

from sklearn.preprocessing import StandardScaler


ex_dq6 = dq6.loc[:,'alcohol':'proline'] # exclude categorical feature

ex_dq6_sorted = dq6['target'] # sort by target values for ease of calculations

scaler = StandardScaler() # Do Z-normalization

ex_dq6 = scaler.fit_transform(ex_dq6) # standardize the numerical feature

embedding = MDS(n_components=2) # embed into 2 components, follow from the tutorial part

ex_dq6_embed = embedding.fit_transform(ex_dq6) # fit MDS transform

dq6_projection = pd.DataFrame({'x': ex_dq6_embed[:, 0], 'y': ex_dq6_embed[:, 1],

                    'target': ex_dq6_sorted}) # log the results in a projection dataframe

sns.scatterplot(x='x', y='y', hue='target', data=dq6_projection) # use dataframe on sns to plot scatterplot classified by target

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.) # print legend

plt.show() # show the graph.
```

The following result is obtained:

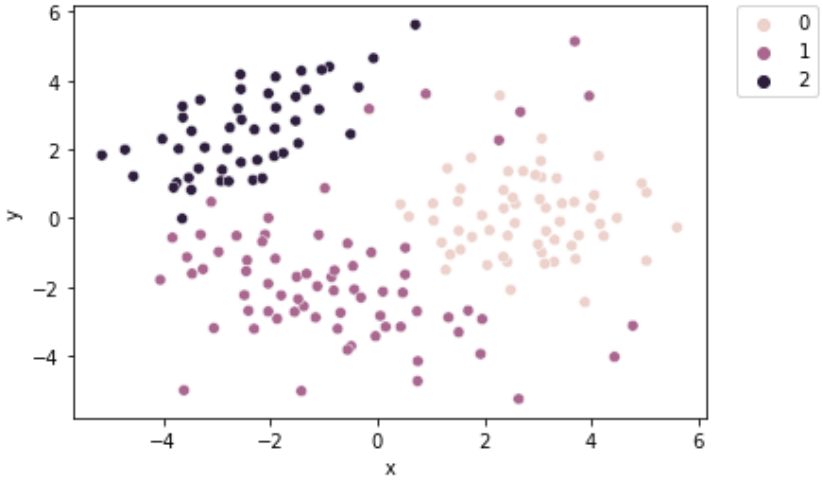


Figure 32. MDS Projection colored by the Categorical Feature

Appendix:

Question 1.1:

The code for Question 1 may be found below:

```
import numpy as np
# Given the list first do the sorting:
q1_data = [2, 15, 20, 5, 1, 4, 7, 9, 10, 3, 14, 8]
print('Given data: ', q1_data)
q1_data.sort()
print('Sorted data: ',  q1_data)
print()
# Equal frequency binning: - Assumed 4 bins are available
# (As for equal frequency division available options are: 2, 3, 4, 6)
NO_BINS = 4
BIN_WIDTH = int(len(q1_data)/NO_BINS)
# Generate new data with equal width. I could
# use transform as well on q1.data to avoid this
data = np.zeros((NO_BINS, BIN_WIDTH))
# Divided the bins - One by one fill the data
count = 0
for i in range(NO_BINS):
    for j in range(BIN_WIDTH):
        data[i,j] = q1_data[count]
        count += 1
print('Data Divided into 4 Bins:')
print(data)
data_bound = data.copy()
print()
# Smoothing by bin means:
print('Smoothing by bin means:')
# Take the mean of the bin; set everything to that value
# for that bin
for b in range(len(data)):
    meanVal =  np.mean(data[b])
    data[b] = meanVal * np.ones((1, BIN_WIDTH))
# Flatten back to original form:
print(data)
data = data.flatten()
print(data)
print()
# Smoothing by bin boundaries:
print('Smoothing by bin boundaries:')
def bin_to_boundary(dataBin):
    min_val = min(dataBin) # min boundary
    max_val = max(dataBin) # max boundary
    binWidth = len(dataBin) #length of bin
    bound = []
    for i in range(binWidth):
        # Calculate distance to bounds
        mn = abs(dataBin[i]-min_val)
        mx = abs(dataBin[i]-max_val)
        if mn >= mx: # in ties attend to max value
            bound.append(max_val)
        else:
            bound.append(min_val)
    return bound
for b in range(len(data_bound)):
    data_bound[b] = bin_to_boundary(data_bound[b])
print(data_bound)
print(data_bound.flatten())
```

Question 1.4:

```
# Bring in the data:

hosa_sat = 71

hosa_dissat = 37


hosb_sat = 129

hosb_dissat = 73


CHI_001 = 10.828


# Calculate the values:

no_sat = hosa_sat + hosb_sat

no_dissat = hosa_dissat + hosb_dissat


total_hosa = hosa_sat + hosa_dissat

total_hosb = hosb_sat + hosb_dissat


n = total_hosa + total_hosb


# Calculate expected frequencies:

exp_hosa_sat = no_sat * total_hosa / n

exp_hosb_sat = no_sat * total_hosb / n


exp_hosa_dissat = no_dissat * total_hosa / n

exp_hosb_dissat = no_dissat * total_hosb / n


# Calculate chi-squared for all and add them all up!

chi_hosa_sat = (hosa_sat - exp_hosa_sat)**2 /
exp_hosa_sat

chi_hosa_dissat = (hosa_dissat - exp_hosa_dissat)**2 /
exp_hosa_dissat

chi_hosb_sat = (hosb_sat - exp_hosb_sat)**2 /
exp_hosb_sat

chi_hosb_dissat = (hosa_dissat - exp_hosb_dissat )**2 /
exp_hosb_dissat


print('Chi hospital A satisfaction score: ' , str(chi_hosa_sat))

print('Chi hospital B satisfaction score: ' , str(chi_hosb_sat))

print('Chi hospital A dissatisfaction score: ' ,
str(chi_hosa_dissat))

print('Chi hospital B dissatisfaction score: ' ,
str(chi_hosb_dissat))


chi = chi_hosa_sat + chi_hosa_dissat  + chi_hosb_sat +
chi_hosb_dissat

print('Calculated chi score: ', str(chi))

print("The premise -patient satisfaction with hospitals are
independent- is " + str(chi <= CHI_001) + '.')

# reject hypothesis when above the threshold
```

Question 1.8:

```
# Migrating code above for quick independant setup:

import pandas as pd

import numpy as np

data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data', header=None)

data.columns = ['Sample code', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape',

            'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',

            'Normal Nucleoli', 'Mitoses','Class']



data = data.replace('?',np.NaN) # Replace ? values with nan

data = data.drop(['Sample code'],axis=1)



## Migrate code from previous section to remove nans; remove duplicates; eliminate outliers:
###################################

# Only Bare nuclei had missing data. Remove them to avoid bias (with data loss)

#print('Number of instances = %d' % (data.shape[0]))

#print('Number of attributes = %d' % (data.shape[1]))

#data.head()

## Migrate code from previous section to remove nans; remove duplicates; eliminate outliers:
###################################

# Only Bare nuclei had missing data. Remove them to avoid bias (with data loss)

data2 = data.dropna()



## Here we drop the class column as we should not use it in the normalization - it is our label

#data2 = data.drop(['Class'],axis=1)
```

```
# Bare Nuclei data had ? for empty areas, so the column was kept as string; even though

# we removed nans, the other samples are stored in string form - lets convert them

data2['Bare Nuclei'] = pd.to_numeric(data2['Bare Nuclei'])

data_dedup = data2.drop_duplicates()



# Deduplication process:

list_class = data_dedup['Class'].tolist()

data3 = data_dedup.drop(['Class'],axis=1) # drop class, saved to somewhere else



#data3.head(30)

print('Number of instances = %d' % (data3.shape[0]))

print('Number of attributes = %d' % (data3.shape[1]))

# Lets normalize the data: and remove the outliers:

Z = (data3-data3.mean())/data3.std()

Z2 = Z.loc[((Z > -3).sum(axis=1)==9) & ((Z <= 3).sum(axis=1)==9),:] # Remove Z score below -3 above +3



print('Number of instances = %d' % (Z2.shape[0]))

print('Number of attributes = %d' % (Z2.shape[1]))



import pandas as pd

from sklearn.decomposition import PCA



# PCA parameters:

PCA_COMP_SIZE = 2

pca = PCA(n_components=PCA_COMP_SIZE)



# Normalize Data - CLASS SHOULD NOT BE USED:

pca.fit_transform(Z)



projected = pca.transform(Z)
```

```
projected = pd.DataFrame(projected,columns=['pc1','pc2'],index=range(1,Z.shape[0]+1))

projected['Class'] = list_class # works but check this probably indices are off.

Projected


# View scatterplot of the results:

import matplotlib.pyplot as plt


colors = {2:'b', 4:'r'}

markerTypes = {2:'o', 4:'o'}


for classtype in markerTypes:

    d = projected[projected['Class']==classtype]


plt.scatter(d['pc1'],d['pc2'],c=colors[classtype],s=60,marker=markerTypes[classtype])


plt.title("PCA Analysis on Breast Cancer Data")

plt.xlabel("Principal Component 1")

plt.ylabel("Principal Component 2")


print('Printing eigenvalues: ', pca.explained_variance_)

print('Variance explained by respective eigenvalue: ' , pca.explained_variance_ratio_)

print('Total Explained Variance Percentage: ', sum(pca.explained_variance_ratio_)*100, '%.')
```

Question 2.3:

```
from scipy.spatial import distance

data = [[-8,-46],[-5,-30],[-2,-15],[-1,-4],[0,-
1],[1,10],[3,23],[4,36],[7,39],[9,55]]



dist = distance.squareform(distance.pdist(data))

sns.heatmap(dist, square=True, xticklabels=False,
yticklabels=False,

            cmap='Blues')



plt.show()



scaler = StandardScaler()

data_norm = scaler.fit_transform(data)



print(data_norm)



dist_norm =
distance.squareform(distance.pdist(data_norm))

sns.heatmap(dist_norm, square=True, xticklabels=False,
yticklabels=False,

            cmap='Blues')



plt.show()
```