```python
import numpy as np
import pandas as pd
```

```python
# import sys
# !{sys.executable} -m pip install -U notebook-as-pdf
```

ECS766 Coursework 1 - Elliot Linsey

Question 1

Original list of numbers: [3, 16, 20, 4, 2, 5, 10, 9, 13, 7, 14, 8]

For binning operations, it helps to sort the values first.

Sorted list of numbers: [2, 3, 4, 5, 7, 8, 9, 10, 13, 14, 16, 20]

For equal frequency binning with 3 bins, we divide the total length of our list by 3 = 12/3 = 4.

This leaves us with the following bins and resulting frequencies:

(2,5] : 4

(7,10] : 4

(13,20] : 4

For binning by smoothing the boundaries, we need to find what interval between our values is common. In this case, the interval of close to 5 appears to fit our values.

This leaves us with the following bins and frequencies:

(2,7] : 5

(8,13] : 4

(14,20] : 3

Question 2

Our list of numbers for normalise: [10,5,25,50,35]

The formula for min-max normalisation is : (value - min)/(max - min)

Applying this to our data results in:

(10 - 5)/(50-5) = 5/45 = 0.11

(5 - 5)/(50-5) = 0/45 = 0

(25 - 5)/(50-5) = 20/45 = 0.44

(50 - 5)/(50-5) = 45/45 = 1

(35 - 5)/(50-5) = 30/45 = 0.66

The formula for Z-score normalisation is (value - mean)/standard deviation

The mean of our dataset is (10 + 5 + 25 + 50 + 35)/5 = 25

To calculate the standard deviation we need to calculate the variance, this involves calculating the difference of each point to the mean, squaring and summing these differences, then dividing this sum by the number of values in our data.

Our differences = [-15,-20,0,25,10]

Our squared differences = [225,400,0,625,100]

This sums to: 1350

For our final variance score, we calculate 1350/5 = 270.

To calculate the standard deviation, we find the square root of this 270 = 16.43

Now we can perform Z-score normalisation.

(10-25)/16.43 = -0.91

(5-25)/16.43 = -1.21

(25-25)/16.43 = 0

(50-25)/16.43 = 1.52

(35-25)/16.43 = 0.61

Question 3

For the chi-square test, first recreate the table and sum both the rows and columns.

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

Then calculate the expected values. This involves multiplying the column and row total of the variable you wish to use, then dividing this figure by the overall total.

The expected value for University A and Satisfied is : (108 * 200)/310 = 69.7

The expected value for University A and Dissatisfied is : (108 * 110)/310 = 38.3

The expected value for University B and Satisfied is : (202 * 200)/310 = 130.3

The expected value for University B and Dissatisfied is : (202 * 110)/310 = 71.7

Update the table with these figures in parentheses.

Rating/University	University A	University B	Total
Satisfied	71 (69.7)	129 (130.3)	200
Dissatisfied	37 (38.3)	73 (71.7)	110
Total	108	202	310

Calculating the x^2 statistic uses this formula : $x^2 = \sum \frac{(O-E)^2}{E}$

This means finding the difference between the original and expected value, squaring this difference then dividing it by the expected value then summing all these resulting values.

Here are the calculations:

(71-69.7)^2/69.7 = 1.3^2/69.7 = 1.69/69.7 = 0.024

(37-38.3)^2/38.3 = -1.3^2/38.3 = 1.69/38.3 = 0.044

(129-130.3)^2/130.3 = -1.3^2/130.3 = 1.69/130.3 = 0.013

(73-71.7)^2/71.7 = 1.3^2/71.7 = 1.69/71.7 = 0.024

x^2 = 0.024 + 0.044 + 0.013 + 0.024 = 0.105

The hypothesis that satisfaction and university are independent with 1 degree of freedom requires a x^2 value of 10.828 or below. With this resulting statistic of 0.105, we can at this stage accept this hypothesis at a significance level of 0.001 and state that these categorical variables are independent.

Question 4

```python
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
import seaborn as sns
```

```python
C1 = pd.read_csv('country-income.csv')
C1
```

	Region	Age	Income	Online Shopper
0	India	49.0	86400.0	No
1	Brazil	32.0	57600.0	Yes
2	USA	35.0	64800.0	No
3	Brazil	43.0	73200.0	No
4	USA	45.0	NaN	Yes
5	India	40.0	69600.0	Yes
6	Brazil	NaN	62400.0	No
7	India	53.0	94800.0	Yes
8	USA	55.0	99600.0	No
9	India	42.0	80400.0	Yes

```python
C12 = C1.copy()
C12 = C12.fillna(C12[['Age','Income']].mean())
C12
```

	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

```python
C13 = C12.copy()
region_num = {'India':1, 'Brazil':2, 'USA':3}
shopper_num = {'Yes':1,'No':0}

C13['Region'] = C13.Region.apply(lambda x: region_num[x] if x in region_num.keys() else x)
C13['Online Shopper'] = C13['Online Shopper'].apply(lambda x: shopper_num[x] if x in shopper_num.keys() else x)
C13
```

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

Question 5

The scatterplots and correlation coefficients (0.71 for female and 0.77 for male) both imply positive correlation between shoe size and height for both genders. This means that as either height increases, so does shoe size and vice versa. Therefore you would expect somebody with larger feet to be taller or somebody that is tall to have larger feet.

```python
shoes = pd.read_csv('shoesize.csv')
shoes.head()
```

	Index	Gender	Size	Height
0	1	F	5.5	60.0
1	2	F	6.0	60.0
2	3	F	7.0	60.0
3	4	F	8.0	60.0
4	5	F	8.0	60.0

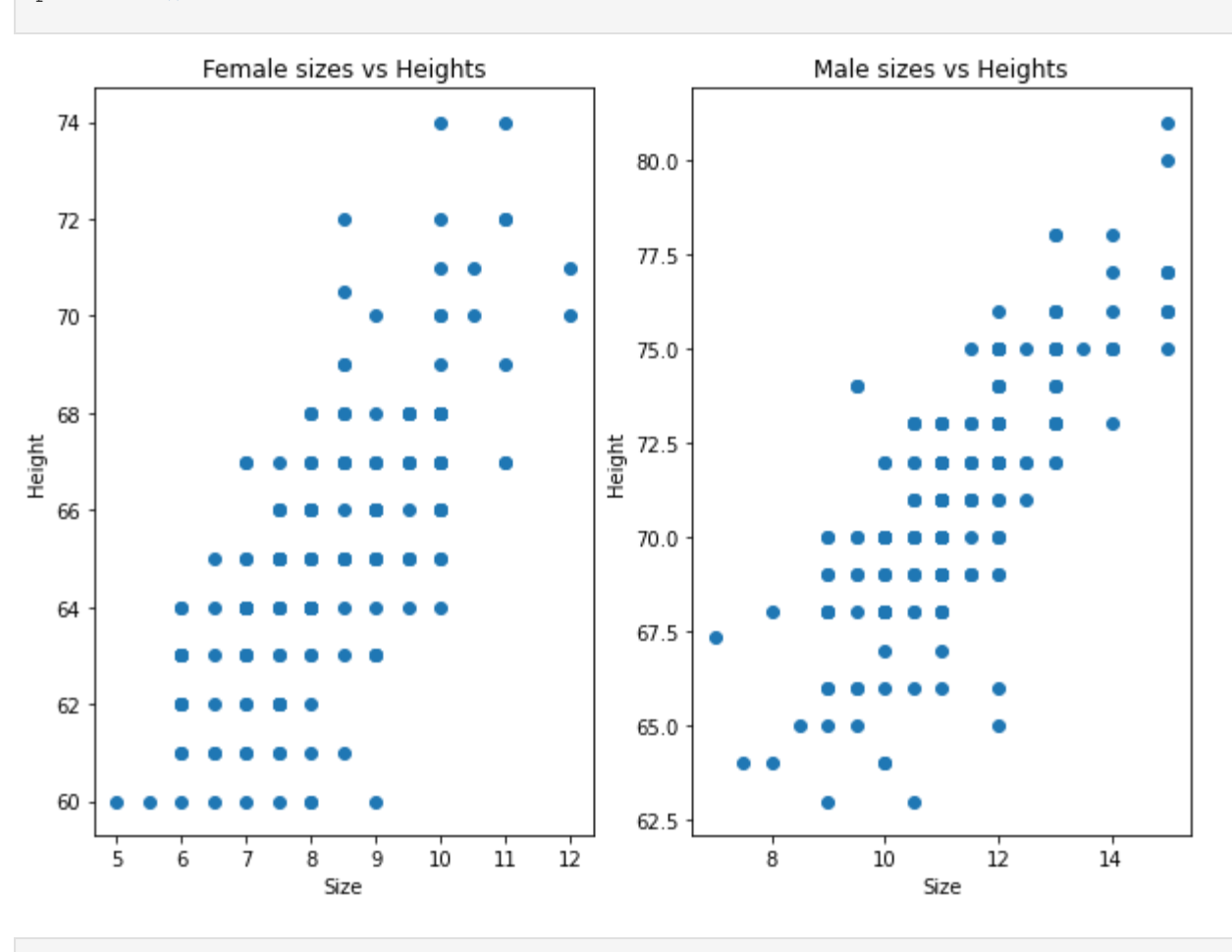
```python
sizes_f = shoes[shoes.Gender == 'F']
sizes_f.head()
```

	Index	Gender	Size	Height
0	1	F	5.5	60.0
1	2	F	6.0	60.0
2	3	F	7.0	60.0
3	4	F	8.0	60.0
4	5	F	8.0	60.0

```python
sizes_m = shoes[shoes.Gender == 'M']
sizes_m.head()
```

	Index	Gender	Size	Height
187	188	M	10.5	63.0
188	189	M	9.0	63.0
189	190	M	7.5	64.0
190	191	M	8.0	64.0
191	192	M	10.0	64.0

```python
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(10,7))
ax1.scatter(sizes_f.Size,sizes_f.Height)
ax1.set_title('Female sizes vs Heights')
ax1.set_xlabel('Size')
ax1.set_ylabel('Height')
ax2.scatter(sizes_m.Size,sizes_m.Height)
ax2.set_title('Male sizes vs Heights')
ax2.set_xlabel('Size')
ax2.set_ylabel('Height')
plt.show()
```




```python
print('Female correlation between shoe size and height: ' + str(sizes_f.Size.corr(sizes_f.Height,method='pears
```

Female correlation between shoe size and height:0.7078119417143971

```python
print('Male correlation between shoe size and height: ' + str(sizes_m.Size.corr(sizes_m.Height,method='pearson'
```

Male correlation between shoe size and height: 0.7677093547300977

Question 6

```python
breasts = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast
breasts.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']
breasts = breasts.drop(['Sample code'],axis=1)
breasts.head()
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	5	1	1	1	2	1	3	1	1	2
1	5	4	4	5	7	10	3	2	1	2
2	3	1	1	1	2	2	3	1	1	2
3	6	8	8	1	3	4	3	7	1	2
4	4	1	1	3	2	1	3	1	1	2

```python
breasts = breasts.replace('?',np.NaN)
breasts2 = breasts.fillna(breasts['Bare Nuclei'].median()).copy()
#print(breasts2.isna().value_counts())
#breasts2.head()
breasts2['Bare Nuclei'] = pd.to_numeric(breasts2['Bare Nuclei'])
breasts3 = breasts2.drop(['Class'],axis=1).copy()
```

```python
Z = (breasts3-breasts3.mean())/breasts3.std()
Z.head()
```

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses
0	0.206788	-0.699494	-0.742767	-0.632794	-0.549168	-0.686488	-0.179534	-0.611387	-0.343666
1	0.206788	0.283642	0.266684	0.768071	1.708882	1.798376	-0.179534	-0.283909	-0.343666
2	-0.503505	-0.699494	-0.742767	-0.632794	-0.549168	-0.410392	-0.179534	-0.611387	-0.343666
3	0.561934	1.594490	1.612618	-0.632794	-0.097558	0.141800	-0.179534	1.353485	-0.343666
4	-0.148359	-0.699494	-0.742767	0.067638	-0.549168	-0.686488	-0.179534	-0.611387	-0.343666

```python
from sklearn.decomposition import PCA
num_components = 2
pca = PCA(n_components=num_components)
projected = pca.fit_transform(Z)

projected = pd.DataFrame(projected,columns=['pc1','pc2'])
projected['Class'] = breasts2.Class
projected.head()
```

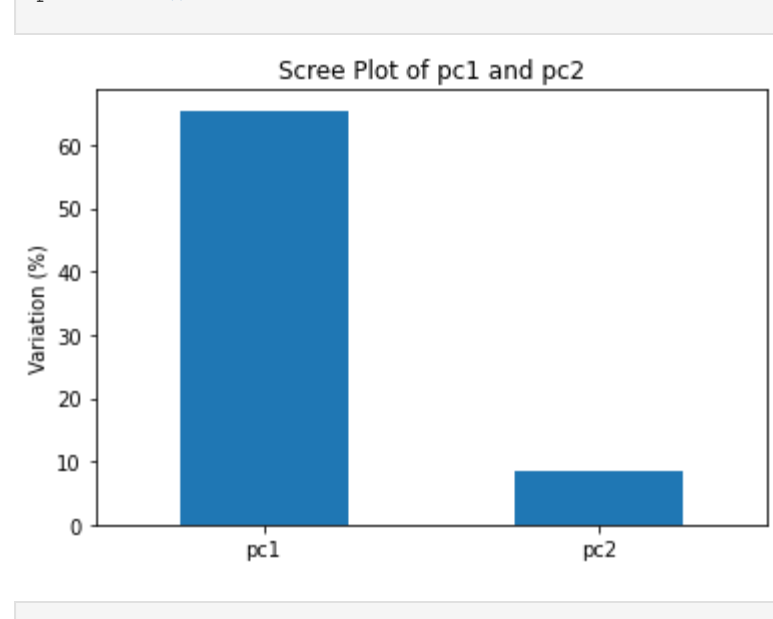
	pc1	pc2	Class
0	-1.455178	-0.110132	2
1	1.465230	-0.544504	2
2	-1.578181	-0.074800	2
3	1.504170	-0.558454	2
4	-1.329599	-0.089592	2

Q6 analysis

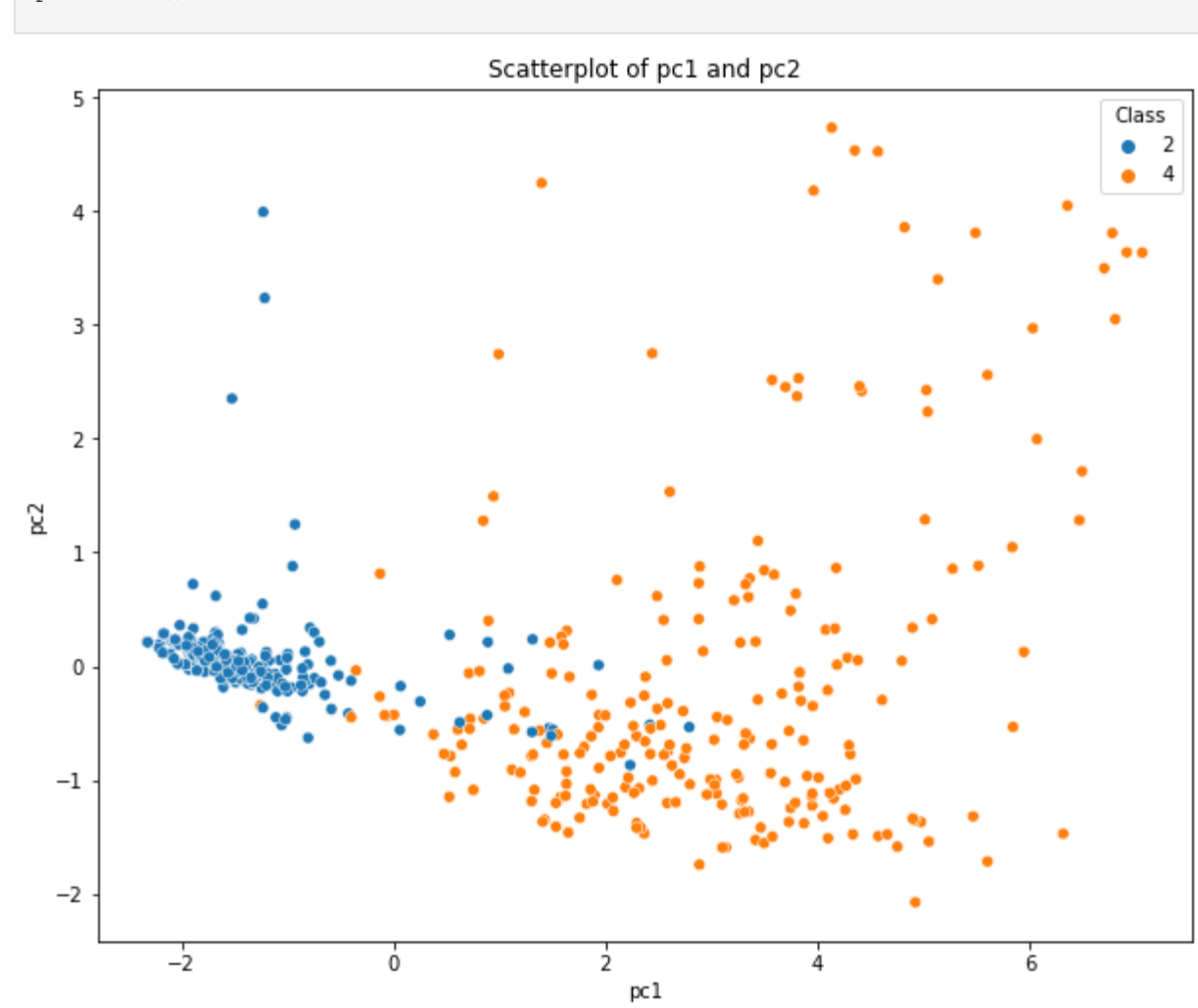
The scree plot is an effective tool that let's us determine the relative importance of principal components. Here we can see that the majority of the variability in the data is explained by pc1 with a value of over 60%. On the other hand, pc2 is far lower with just under 10%. As we can see on the scatterplot below, there is far greater movement along the x-axis of pc1 compared to the y-axis of pc2 which fits our scree plot results that pc1 captures more variation in the data than pc2. We can also see that class 2 appears to be clustered around coordinates [-2,0] whereas class 4 appears to have a greater range and variation of values. Pc1 also appears to capture more of the variation in class 2 than pc2 as class 2 moves along the x-axis more than the y-axis. This is also slightly apparent for class 4 as a lot of the movement is within the x-axis compared to the y-axis. This could be expected though as pc1 captures 60% more of the variation in the data than pc2 so you would expect a greater variation of values.

The scree plot bars do not sum to 100%, this is because we have set our number of principal components to only 2 so as to plot best onto a 2-D plot. If we were to remove this limit we would end up with 9 principal components as there are 9 variables in our breast cancer dataframe. However, we would likely see a descending trend in variation percentage for these extra 7 principal components whilst using the first 2 captures over approximately 70% of the variation in the data. Within this task, 70% may be acceptable to capture the groupings of data, but in other situations we could set a threshold of 80% of above and therefore require more principal components to reach that figure of variation.

```python
fit_test = pca.fit(Z)
exp_var = pd.DataFrame(fit_test.explained_variance_ratio_*100)
ax = exp_var.plot.bar(legend=False)
ax.set_xticklabels(['pc1','pc2'],rotation=0)
ax.set_ylabel('Variation (%)')
ax.set_title('Scree Plot of pc1 and pc2')
plt.show()
```




```python
fig = plt.figure(figsize=(10,8))
sns.scatterplot(data=projected,x='pc1',y='pc2',hue='Class',palette='tab10')
plt.title('Scatterplot of pc1 and pc2')
plt.show()
```



Second Part of the Assignment

Q1

```python
df = pd.read_csv('graduation_rate.csv')
df.head()
```

	ACT composite score	SAT total score	parental level of education	parental income	high school gpa	college gpa	years to graduate
0	30	2206	master's degree	94873	4.0	3.8	3
1	26	1953	some college	42767	3.6	2.7	9
2	28	2115	some high school	46316	4.0	3.3	5
3	33	2110	some high school	52370	4.0	3.5	4
4	30	2168	bachelor's degree	92665	4.0	3.6	4

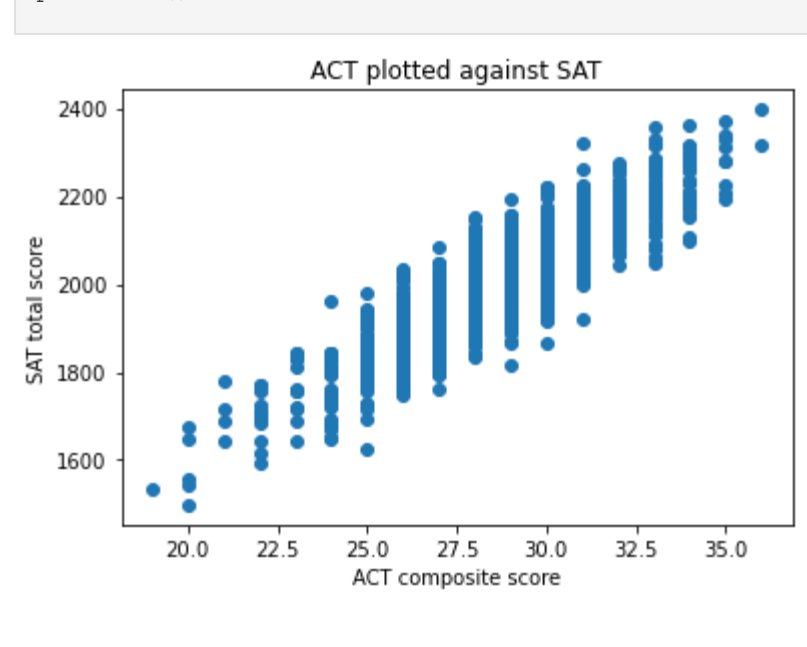
Question 1

As both ACT and SAT score are numerical values, we can find association between them using a correlation matrix or a scatter plot. The correlation value is 0.88 which would imply a very high positive correlation. This is further displayed in the scatter plot, as ACT increases, so does SAT and vice versa.

```python
df.corr()
```

	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

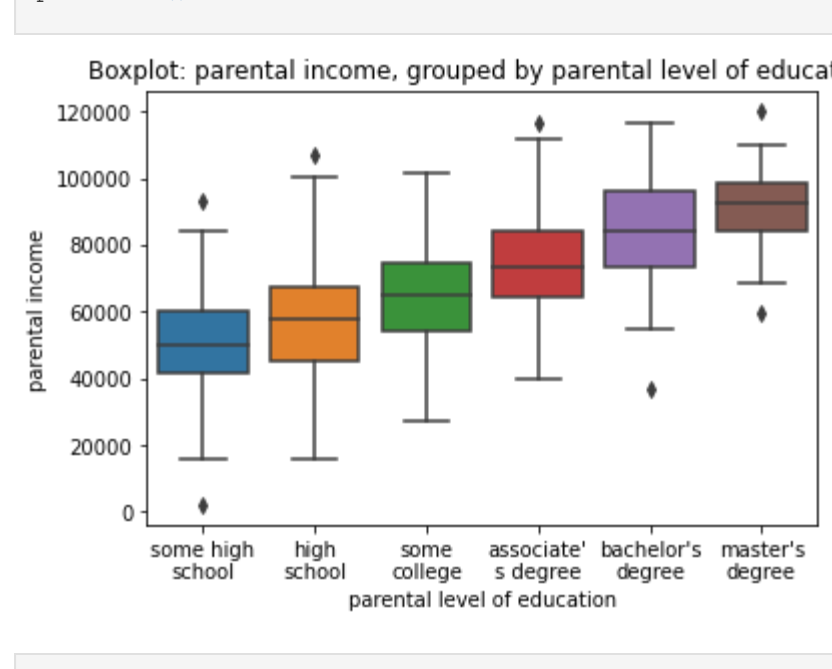
```python
plt.scatter(df['ACT composite score'], df['SAT total score'])
plt.xlabel('ACT composite score')
plt.ylabel('SAT total score')
plt.title('ACT plotted against SAT')
plt.show()
```



Question 2

There appears to be a clear positive trend as you progress higher through parental level of education, as the education increases so does the parental income's median value from around 50,000 for some high school to 90,000 for master's degree. The ranges also start to decrease from approximately 60,000 for some high school to 40,000 for master's degrees however the interquartile ranges stay approximately the same size. The numbers of outliers seem to be fairly consistent throughout the boxplots, however their values correspond differently between each education level, for instance the lower outlier for the masters degree is close to the median value of some high school education.

```python
ax = sns.boxplot(x='parental level of education', y='parental income', data=df,order=['some high school','high
                                                                                      ,'some college',"associate
                                                                                      "bachelor's degree","maste
plt.title('Boxplot: parental income, grouped by parental level of education')

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

plt.show()
```




```python
df_masters = df[df['parental level of education'] == 'master\'s degree']
display(df_masters.sort_values(by='parental income',ascending=False)[0:1])
display(df_masters.sort_values(by='parental income',ascending=True)[0:1])
```

	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

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

Question 3

It is important to scale different features when comparing distances as if their magnitudes are too different then one feature may take precedence over another. For example, if you were trying to find the distances between individuals based on shoe size and height, both height and shoe_size use different magnitudes or scales. Shoe size ranges from 5 to 10, so a max distance of 5, whilst heights ranges from 153 to 194 with a max distance of 41. In trying to find the distances between these individuals, heights would be weighted more heavily because there is a greater numerical distance between most of the values. By using scaling, we put all of the variables into a similar range to negate this effect. In essence, we are converting the values to a common scale without distorting the differences in the range of values.

Below we have two arrays of heights and shoe size, both on different scales. By normalising the data, we have moved both scales to between approximately -2 to 2.

```python
from sklearn.preprocessing import StandardScaler
heights = np.array([160,180,155,173,153,163,177,175,194,169])
shoe_size = np.array([6,7,8,8,9,10,8,8,6,5])
zipped = np.column_stack((heights,shoe_size))
scaler = StandardScaler()
zipped = scaler.fit_transform(zipped)
zipped
```

```
array([[-0.8299425 , -1.04764544],
       [ 0.84670902, -0.34921515],
       [-1.24910538,  0.34921515],
       [ 0.25988098,  0.34921515],
       [-1.41677053,  1.04764544],
       [-0.57844477,  1.74607574],
       [ 0.59521129,  0.34921515],
       [ 0.42754614,  0.34921515],
       [ 2.02036508, -1.04764544],
       [-0.07544932, -1.74607574]])
```

Question 4

The distance matrix shows Euclidean distances between variables. The diagonal shows a mirror image of both halves so the distances between the variables are shown twice. At the top of the y-axis is some high school, the y-axis is sorted in descending order so the bottom of the y-axis is master's degree. The x-axis is sorted in ascending order from some high school to master's degree. Therefore, the top left is the distance between some high school and some high school which will have the smallest distance and therefore be coloured lightest. The bottom left will compare distances between master's degree (on y-axis) and some high school (on x-axis), these will be the most different and will be coloured darkest to signify this.

Question 5

What this form of interpolation is doing is assigning a numerical value for every combination of x and y. It is then assigning each of these values a colour which is plotted in this first heatmap. By using interpolation='nearest', every value between the z-values is assigned the colour of the z-value that is closest. For example, the point at x=-0.85 and y=-1 is not formally defined in the dataframe but its distance is closest to the z value 1.60. Therefore, it will also take the colour that z=1.60 holds.

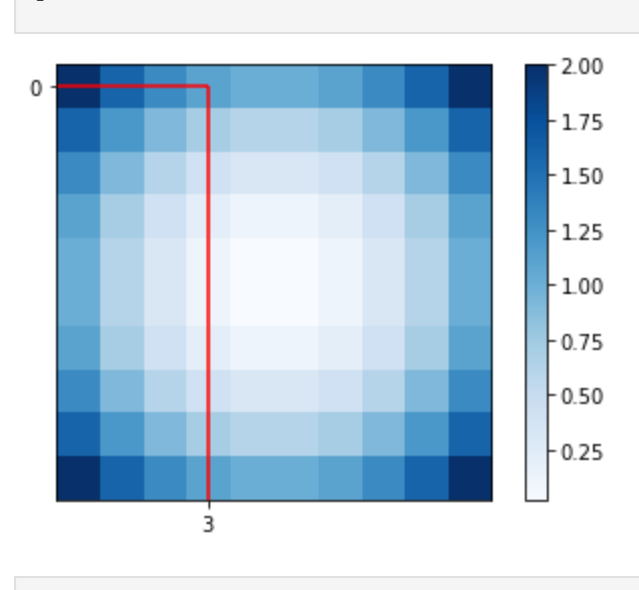
By using 100 x and y values to create the Z matrix from, you are reducing the space between the Z values and therefore increasing the numbers or shades of colours available. Similar to how video resoluton works, there are now more pixels as with our previous example of x=-0.85 and y = -1, the closest z-value would no longer be 1.6 but could be something like 1.84 which would have a more specific rgb value.

The resulting heatmap axes are also a map of the Z matrix indices. The first map has red lines showing the colour of Z[0,3] which is a value of 1.11. The second heatmap has the same indice of Z[0,3] but because there are far more z-values it now has a value of 1.88.

```python
import numpy as np
x_range = np.linspace(-1, 1, 10)
y_range = np.linspace(-1, 1, 10)

# meshgrid: X[i, j] == x_range[j] and Y[i, j] == y_range[i]
X, Y = np.meshgrid(x_range, y_range)
# Z[i, j] == f(x_range[j], y_range[i])
Z = X**2 + Y**2
print('Z value = ' + str(Z[0,3]))
# Dataset representation
df = pd.DataFrame({'x': X.reshape(-1), 'y': Y.reshape(-1), 'z = f(x,y)': Z.reshape(-1)})
#display(df)
```

Z value = 1.1111111111111112

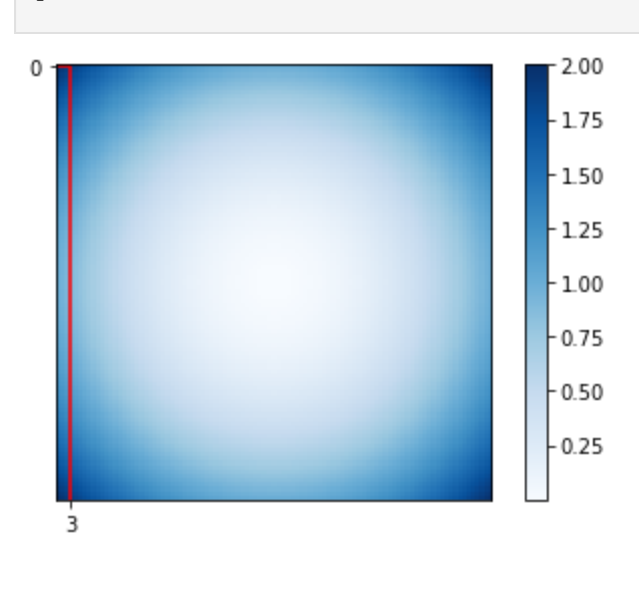
```python
# Interpolation: point (x, y) is colored according to the value z of the nearest point in the dataset
plt.imshow(Z, cmap='Blues', aspect='equal', interpolation='nearest')
plt.colorbar()

# xticks and yticks would show Z matrix indices
plt.xticks([3])
plt.yticks([0])
plt.vlines(3,0,9.5,color='r')
plt.hlines(0,-0.5,3,color='r')

plt.show()
```



```python
x_range = np.linspace(-1, 1, 100)
y_range = np.linspace(-1, 1, 100)

# meshgrid: X[i, j] == x_range[j] and Y[i, j] == y_range[i]
X, Y = np.meshgrid(x_range, y_range)
# Z[i, j] == f(x_range[j], y_range[i])
Z = X**2 + Y**2
print('Z value = ' + str(Z[0,3]))
# Dataset representation
df = pd.DataFrame({'x': X.reshape(-1), 'y': Y.reshape(-1), 'z = f(x,y)': Z.reshape(-1)})
#display(df)
```

Z value = 1.8824609733700643

```python
plt.imshow(Z, cmap='Blues', aspect='equal', interpolation='nearest')
plt.colorbar()

# xticks and yticks would show Z matrix indices
plt.xticks([3])
plt.yticks([0])
plt.vlines(3,99.5,0,color='r')
plt.hlines(0,0,3,color='r')

plt.show()
```



Question 6

```python
from sklearn.datasets import load_wine
from sklearn.manifold import MDS
data = load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = pd.Series(data.target)
df.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	h
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.

6.1

```python
df.target.value_counts()
```

```
1    71
0    59
2    48
Name: target, dtype: int64
```

6.2

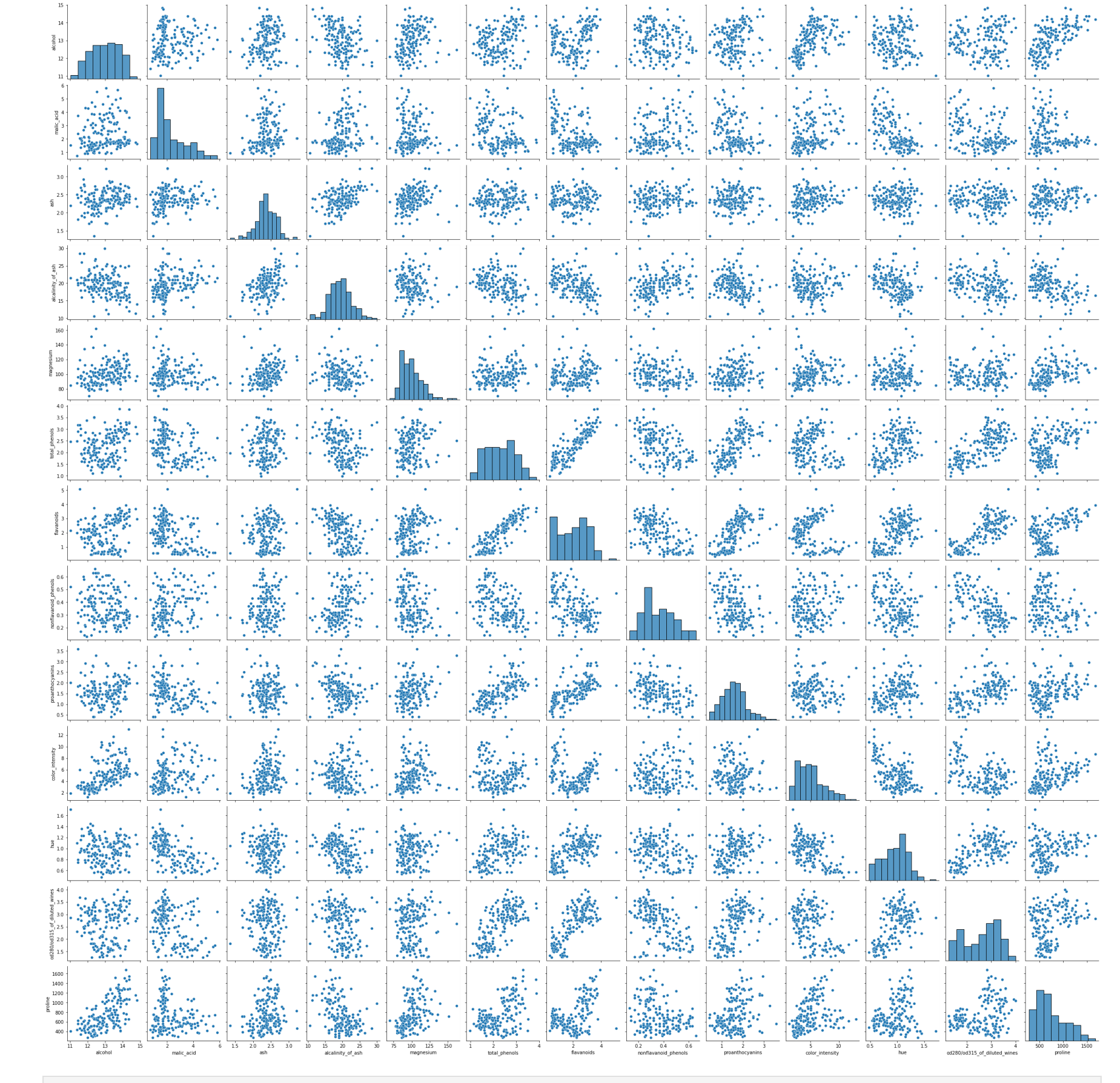
```python
df.iloc[:, 0:13].describe()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	col
count	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	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	

```python
df.iloc[:,0:13].corr()
```

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

```python
sns.pairplot(df.iloc[:, 0:13], diag_kind='hist')
plt.show()
```



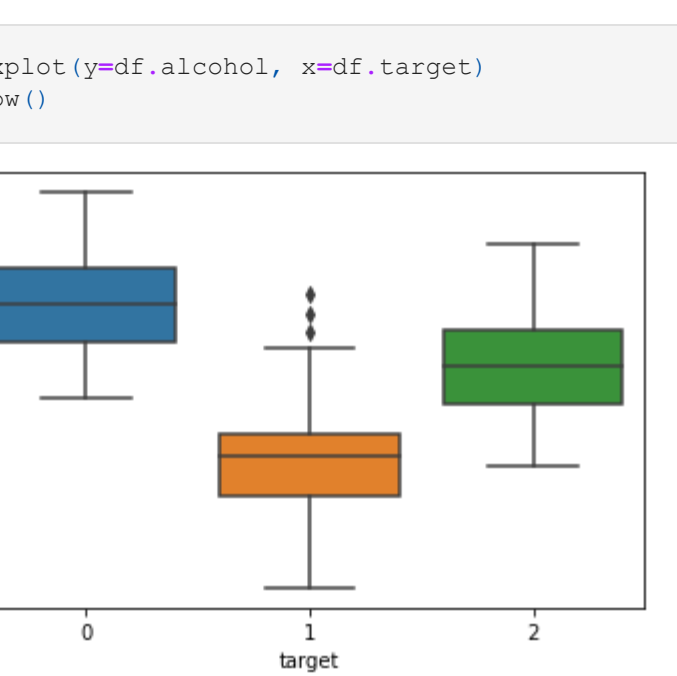
In [41]:
```
df_gby = df.groupby('target').median()
df_gby
```

Out[41]:

target	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensit
0	13.750	1.770	2.44	16.8	104.0	2.800	2.980	0.29	1.870	5.4
1	12.290	1.610	2.24	20.0	88.0	2.200	2.030	0.37	1.610	2.9
2	13.165	3.265	2.38	21.0	97.0	1.635	0.685	0.47	1.105	7.5

6.3

In [42]:
```
sns.boxplot(y=df.alcohol, x=df.target)
plt.show()
```



6.4

In [43]:
```
df.corr()
```

Out[43]:

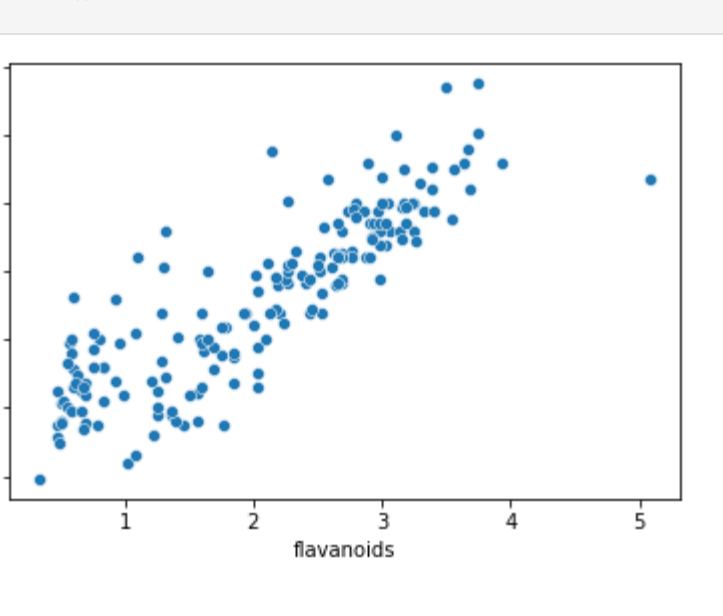
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.289101	0.236815	-0.155929
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	-0.335167	-0.411007	0.292977
ash	0.211545	0.164045	1.000000	0.443367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

Finding the variables with the highest correlation.

In [44]:
```
df.corr().unstack().sort_values(ascending=False).drop_duplicates()
```

Out[44]:
```
alcohol                       alcohol                         1.000000
flavanoids                    total_phenols                   0.864564
                              od280/od315_of_diluted_wines    0.787194
od280/od315_of_diluted_wines  total_phenols                   0.699949
flavanoids                    proanthocyanins                 0.652692
                                                                ...
target                        hue                            -0.617369
proline                       target                         -0.633717
target                        total_phenols                  -0.719163
od280/od315_of_diluted_wines  target                         -0.788230
target                        flavanoids                     -0.847498
Length: 92, dtype: float64
```

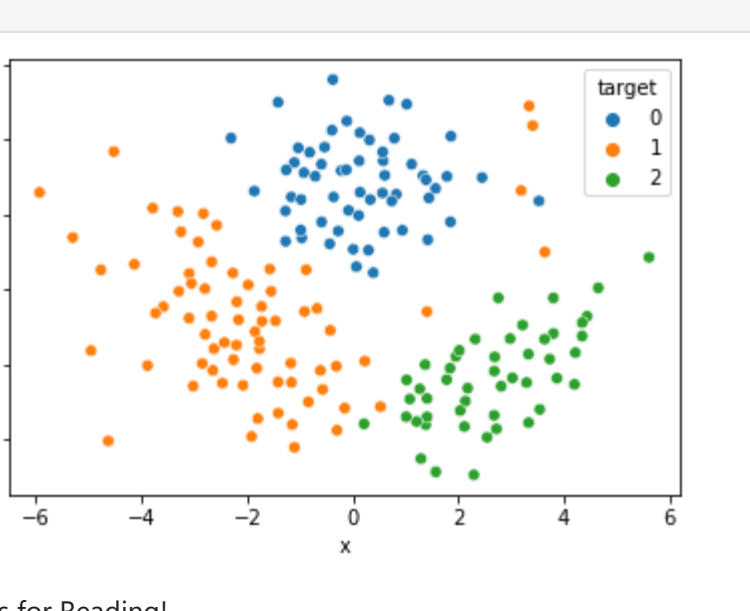
In [45]:
```
sns.scatterplot(data=df, x='flavanoids', y='total_phenols')
plt.show()
```



6.5

In [46]:
```
scaler = StandardScaler()
df_no_target = df.iloc[:, 0:13]
df_standard = scaler.fit_transform(df_no_target)
#df_standard
```

In [49]:
```
embedding = MDS(n_components=2)
Xp = embedding.fit_transform(df_standard)
df_projection = pd.DataFrame({'x': Xp[:, 0], 'y': Xp[:, 1],
                              'target': df.target})
df_projection.head()
sns.scatterplot(data=df_projection, x='x', y='y', hue='target',palette='tab10')
plt.show()
```



Thanks for Reading!

Elliot Linsey

QMUL