Assigment 1 Data mining

Monday, 25 October 2021 13:16

Q1 Data:- [3,16, 20, 4, 2, 5, 10, 9, 13, 7, 14, 8]

Ans 1) Equal - frequency binning

sorted data:- [2, 3, 4, 5, 7, 8, 9, 10, 13, 14, 16, 20]

Output of equal frequency binning:-

$$\left.\begin{array}{l}[2,3,4,5]\\ [7,8,9,10]\\ [13,14,16,20]\end{array}\right\}\text{divided into equal sized bins.}$$

It is a pre processing technique where the input data is divided into smaller sets which are called "bins". This is used for smoothing, which can further lead to decrease overfitting.

2) Smoothing by bin boundaries

For performing binning by boundaries we need to sort our data first.

Sorting in ascending order.

[2, 3, 4, 5, 7, 8, 9, 10, 13, 14, 16, 20]

Bins :-

Bin 1:- [2, 3, 4, 5]

Bin 2:- [7, 8, 9, 10]

Bin 3:- [13, 14, 16, 20]

Bins after performing smoothing by boundaries

Bin 1:- [2, 2, 5, 5] → closer to right boundary

"closer the the left boundary"

Bin 2:- [7, 7, 10, 10]

Bin 3:- [13, 13, 13, 20]

Hence we've obtained the new bins after performing binning.

Q2 Data Normalization:-

Ans Data : [10, 5, 25, 50, 35]

i) Min- Max normalization (min=0, max= 1)

Here we will transform our data into values ranging from min to max (0-1).

∴ The minimum value of the data will take the value '0' and the max would take '1'. The other values will be transformed using $\frac{\text{value} - \text{min}}{\text{max} - \text{min}}$ → min from data

↳ max from data.

∴

min from data = 5

max from data = 50

∴ The values 5, 50 will be transformed to 0 and 1.

Transforming other values.

∴ Value 10:-

$$\rightarrow \frac{10-5}{50-5} = \frac{5}{45} = 0.11$$

Value 25:-

$$\frac{25-5}{50-5} = \frac{20}{45} = 0.44$$

Value 35:-

$$\frac{35-5}{50-5} = \frac{30}{45} = 0.66$$

∴ The transformed data is

$$\boxed{[0, 0.11, 0.44, 0.66, 1]}$$

2) z- score normalization.

In z-score normalization we use the mean and the standard deviation of the data to transform our data.

$$\mu \rightarrow \text{mean} = \frac{[5+10+25+35+50]}{5}$$

$$= \frac{125}{5} = \boxed{25}$$

$$\sigma \rightarrow \text{std. dev.} = \Sigma\sqrt{\frac{(x_i-\mu)^2}{N}}$$

$$\therefore \sqrt{\frac{(5-25)^2+(10-25)^2+(25-25)^2+(35-25)^2+(50-25)^2}{5}}$$

$$= \sqrt{\frac{400+225+0+100+625}{5}}$$

$$= \sqrt{270}$$

$$= 16.43$$

$\therefore$ value 5 $\rightarrow$ $\frac{5-\mu}{sd} = \frac{5-25}{16.43} = -1.217$

value 10 $\rightarrow$ $\frac{10-25}{16.43} = -0.912$

value 25 $\rightarrow$ $\frac{25-25}{16.43} = 0$

value 35 $\rightarrow$ $\frac{35-25}{16.43} = 0.609$

value 50 $\rightarrow$ $\frac{50-25}{16.43} = 1.521$

Transformed z-score normalised data

$$\boxed{= [-1.217, -0.912, 0, 0.609, 1.521]}$$

Q3 Perform chi-square test.

Data:-

Rating/University	Uni A	Uni B
Satisfied	71	129
Dissatisfied	37	73

To check if student satisfaction is correlated with a specific university.

level of significance = 0.001

chi square significance value = 10.828

Null hypothesis:-

Student satisfaction is independent from specific university.

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

Calculating expected values using the given values from the table.

$$E = \frac{\text{row total} \times \text{column total}}{\text{sample size.}}$$

$\therefore$ satisfied, A $= \frac{108 \times 200}{310} = 69.65$

satisfied, B $= \frac{202 \times 200}{310} = 130.32$

dissatisfied, A $= \frac{108 \times 110}{310} = 38.32$

dissatisfied, B $= \frac{202 \times 110}{310} = 71.68.$

Now calculating the chi score. which is $\boxed{x^2}$

$$= \frac{\Sigma (O-E)^2}{E^2}$$

$$\frac{(71-69.68)^2}{69.68} + \frac{(37-38.32)^2}{38.32} + \frac{(73-71.68)^2}{71.68} + \frac{(129-130.32)^2}{130.32}$$

$$= \frac{1.74}{69.68} + \frac{1.74}{38.32} + \frac{1.74}{71.68} + \frac{1.74}{130.32}$$

$$\boxed{= 0.108}$$

The value obtained 0.108 is far smaller than the significance value 10.828.

$\therefore$ We don't reject the null hypothesis.

$\boxed{\text{Hence student satisfaction is independent from the university they study in.}}$

QUESTION NUMBER 4

In [ ]:

```
#importing pandas to create a dataframe
import pandas as pd

#importing the library numpy
import numpy as np

#reading the csv file into a dataframe
data = pd.read_csv('country-income.csv')
data
```

Out[ ]:

	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

In [ ]:

```
#replacing all the NaN values in the column 'Income' with its mean which is 76533.33
data['Income']= data['Income'].replace(np.nan, data['Income'].mean())
data
```

Out[ ]:

	Region	Age	Income	Online Shopper
0	India	49.0	86400.000000	No
1	Brazil	32.0	57600.000000	Yes
2	USA	35.0	64800.000000	No
3	Brazil	43.0	73200.000000	No
4	USA	45.0	76533.333333	Yes
5	India	40.0	69600.000000	Yes
6	Brazil	NaN	62400.000000	No
7	India	53.0	94800.000000	Yes
8	USA	55.0	99600.000000	No
9	India	42.0	80400.000000	Yes

In [ ]:

```
from sklearn.preprocessing import LabelEncoder

# creating instance of labelencoder
```

```
labelencoder = LabelEncoder()

# Using the labelencoder we transform the categorical values of the column Online Shopper
to Numerical
# values where "No" corresponds to a "0" and "Yes" corresponds to "1".
data['Online Shopper Numerical'] = labelencoder.fit_transform(data['Online Shopper'])

# Using the labelencoder we transform the categorical values of the column Region to Nume
rical
# values where "India" corresponds to a "1", "Brazil" corresponds to "0" and "USA" corres
ponds to "2".
data['Region Numerical'] = labelencoder.fit_transform(data['Region'])
data
```

Out[ ]:

	Region	Age	Income	Online Shopper	Online Shopper Numerical	Region Numerical
0	India	49.0	86400.000000	No	0	1
1	Brazil	32.0	57600.000000	Yes	1	0
2	USA	35.0	64800.000000	No	0	2
3	Brazil	43.0	73200.000000	No	0	0
4	USA	45.0	76533.333333	Yes	1	2
5	India	40.0	69600.000000	Yes	1	1
6	Brazil	NaN	62400.000000	No	0	0
7	India	53.0	94800.000000	Yes	1	1
8	USA	55.0	99600.000000	No	0	2
9	India	42.0	80400.000000	Yes	1	1

QUESTION NUMBER 5

In [ ]:

```
#importing pandas to create a dataframe
import pandas as pd

#importing the library numpy
import numpy as np

#reading the csv file into a dataframe
data = pd.read_csv('shoesize.csv')
data
```

Out[ ]:

	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
...	...	...	...	...
403	404	M	13.0	78.0
404	405	M	13.0	78.0
405	406	M	14.0	78.0
406	407	M	15.0	80.0

407	408	M	15.0	81.0
	Index	Gender	Size	Height

408 rows × 4 columns

In [ ]:

```
#importing the matplotlb library for generating plots
import matplotlib.pyplot as plt

#Defining the plot variables for size and height for each gender
plot_xf = data[data['Gender'] == 'F']['Size']
plot_xm = data[data['Gender'] == 'M']['Size']
plot_yf = data[data['Gender'] == 'F']['Height']
plot_ym = data[data['Gender'] == 'M']['Height']
```

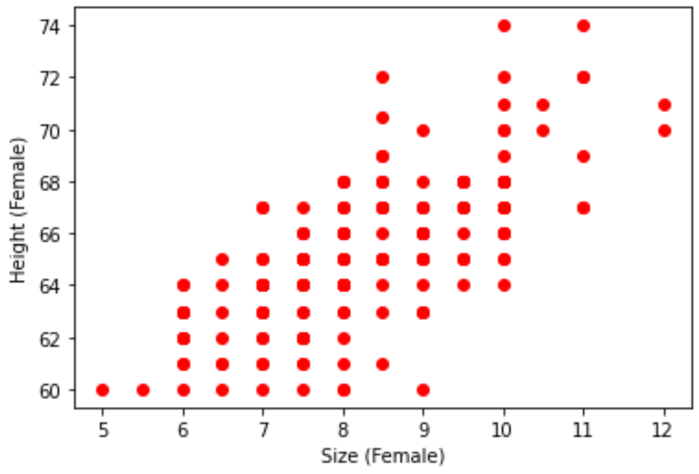
In [ ]:

```
# importing the library pearsonr from the scipy.stats module
from scipy.stats import pearsonr

# creating a function to calculate the pearson's correlation coefficient
def pcc(data1, data2):
  coef = pearsonr(data1, data2)
  return coef
```

In [ ]:

```
#scatter plot for Size vs Height Female

plt.xlabel('Size (Female)')
plt.ylabel('Height (Female)')

#plotting both the variables and setting the dot color to red
plt.scatter(plot_xf,plot_yf, c='r')

# using the function "pcc" we created, we calculate the correlation coefficient
pcc_female = pcc(plot_xf,plot_yf)
print(pcc_female)

#the first value obtained is the correlation coefficient, hence we take the first value f
rom the varibale pcc_female
print(f'The pearsons correlation coefficient for shoe size vs height for female is {pcc_f
emale[0]}')
```

```
(0.7078119417143971, 9.773450790332586e-30)
The pearsons correlation coefficient for shoe size vs height for female is 0.707811941714
3971
```

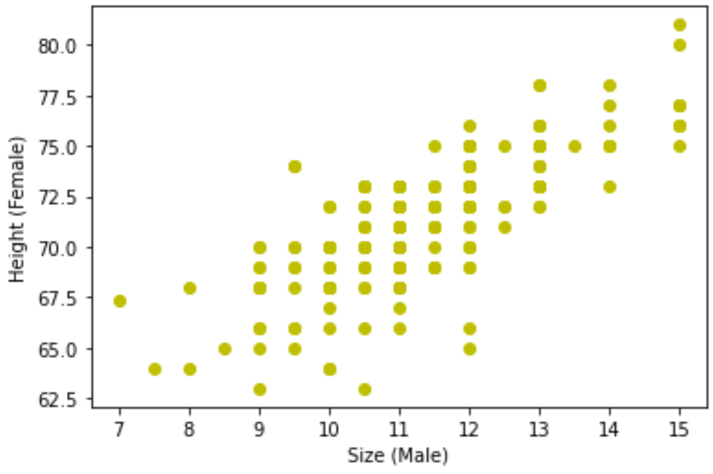


In [ ]:

```
#scatter plot for Size vs Height Male

plt.xlabel('Size (Male)')
plt.ylabel('Height (Female)')
```

```
#plotting both the variables and setting the dot color to yellow
plt.scatter(plot_xm,plot_ym, c='y')

# using the function "pcc" we created, we calculate the correlation coefficient
pcc_male = pcc(plot_xm,plot_ym)
print(pcc_male)

#the first value obtained is the correlation coefficient, hence we take the first value f
rom the varibale pcc_male
print(f'The pearsons correlation coefficient for shoe size vs height for male is {pcc_mal
e[0]}')
```

```
(0.7677093547300977, 3.2857111133112256e-44)
The pearsons correlation coefficient for shoe size vs height for male is 0.76770935473009
77
```



By looking at the scatterplots for both the genders, it can be inferred that the attribute "shoe size" and "height" are correlated with each other. The value of correlation (Pearson's Correlation Coefficient) in case if Female is "0.707" and in case if Males is "0.7677". The value is slightly higher in case of Males so it means that shoe size depends on height or vice versa more in case if Males as compared to Females.

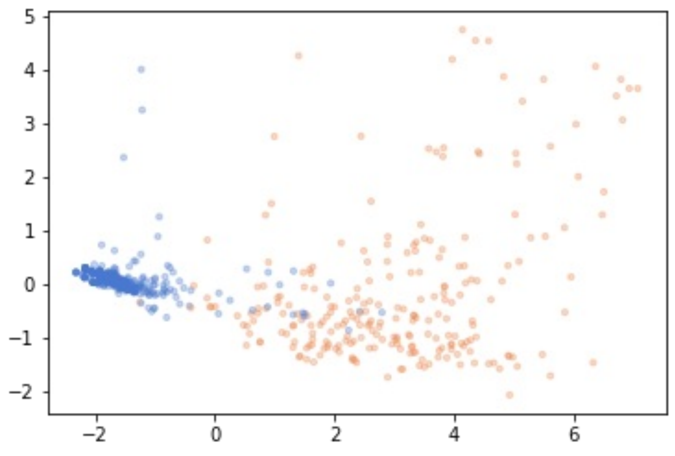
QUESTION NUMBER 6

In [ ]:

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns


from sklearn.preprocessing import StandardScaler
data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-canc
er-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', 'Blan
d Chromatin',
                'Normal Nucleoli', 'Mitoses','Class']

from sklearn import preprocessing
from scipy import stats

data_color = data.copy()

data = data.drop(['Sample code'],axis=1)
data = data.drop(['Class'],axis=1)
data = data.replace('?', np.NaN)
data['Bare Nuclei'] = data['Bare Nuclei'].fillna(data['Bare Nuclei'].median())
```

```
#normalizing the data using z-score normalization

# data_norm = (data-mean) / data.std()
scalar = StandardScaler()
data_norm = scalar.fit_transform(data)



pca = PCA(n_components=2)
finaldata = pca.fit_transform(data_norm)

# variance ratio for the two components
print(" variance ratio ", pca.explained_variance_ratio_)

pca_df = pd.DataFrame(finaldata,columns=['PC1','PC2'])
pca_df['Class'] = data_color['Class']
print(pca_df)

x = pca_df['PC1']
y = pca_df['PC2']

colors = {'2':'red', '4':'green'}

# Get Unique Classes from our dataframe
color_labels = pca_df['Class'].unique()

# List of colors in the color palettes
rgb_values = sns.color_palette("muted")

# Map Class to the colors
color_map = dict(zip(color_labels, rgb_values))

# Finally use the mapped values to plot the scatterplot

plt.scatter(x, y, s=10, alpha=0.3,
            c= pca_df['Class'].map(color_map))
plt.show()
```

```
 variance ratio  [0.65445704 0.0860859 ]
          PC1       PC2  Class
0   -1.456220 -0.110210      2
1    1.466279 -0.544894      2
2   -1.579311 -0.074854      2
3    1.505247 -0.558853      2
4   -1.330551 -0.089657      2
..        ...       ...    ...
694 -1.711249  0.188019      2
695 -2.063036  0.234224      2
696  3.825359 -0.180466      4
697  2.269482 -1.113435      4
698  2.664453 -1.197242      4

[699 rows x 3 columns]
```



ASSIGNMENT 1 PART 2 ANSWERS

Question 1

The correlation value 0.885884 between 'ACT composite score' and 'SAT total score' shows that both the variables are highly correlated. As the value of 'ACT composite score' increases the value of 'SAT total score' increases as well. This relationship can be best visualized using the "Scatter Plot"

Question 2

The box plot shows a correlation between 'the parental level of education' and 'parental income'. The parental income increases as their level of education gets higher.

In [24]:

```
#import the library pandas
import pandas as pd

#read the csv data as a pandas dataframe
df = pd.read_csv('graduation_rate.csv')

#selecting all the data where the parental level of education is master's degree
df1 =  df.loc[df['parental level of education'] == "master's degree"]

#sorting the data into ascending order in order to look for outliers
df1_sorted = df1.sort_values(by='parental income', ascending=True)
df1
```

Out[24]:

	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
27	28	2058	master's degree	96573	3.9	3.6	4
37	34	2363	master's degree	90775	4.0	3.6	4
65	32	2161	master's degree	75974	4.0	3.8	3
99	30	2223	master's degree	85135	4.0	3.8	3
...	...	...	...	...	...	...	...
912	31	2078	master's degree	83814	3.9	3.5	4
927	33	2330	master's degree	92185	4.0	3.8	4
929	35	2338	master's degree	79398	4.0	3.7	4
964	27	1999	master's degree	93314	3.7	3.3	5
968	32	2259	master's degree	85534	4.0	3.6	5

68 rows × 7 columns

From looking at the table visualization we can come to a conclusion that there are two outliers in the data set. One is the maximum value and the other is the minimum value. Both the outliers have a difference of atlesat 9000.

In [25]:

```
#printing the first value of the dataframe which is the first outlier
```

```
df1_sorted.head(1)
```

Out[25]:

	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

In [26]:

```
#printing the last value of the dataframe which is the second outlier
df1_sorted.tail(1)
```

Out[26]:

	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

Question 3

Feature scaling is an important part of data visualisation. Using the StandardScaler we standardize all our data into a comparable magnitude. This is done to overcome false values when the distance is calculated. If feature scaling would not have been performed to this data frame the disatnce matrix would have shown abnormally large values resulting in a heatmap of diverse values. For example:- When calculating a Body Mass Index the values for height and weight needs to be in meters and kilograms respectively. For a data with height in centimeres and weight in kilograms, the Body Mass Index values would br far different from the true values. Hence there is the need to standardize the data.

Question 4

It can be inferred by looking at the heat map that the top right and the bottom left parts are darker than the other regions of the heatmap. The data before plotting the heatmap was sorted according to the attribute parental level education which increases from left to right and top to bottom. The minimum being just some high school education and the maximum being a master's degree.Hence it can be inferred that the average distance between students whose parents only have some high school education and students whose parents have a master's degree (darker part of the map/bottom left and top right parts of the heatmap) is larger than the average distance between students whose parents only have some high school education(top left part of the heatmap).

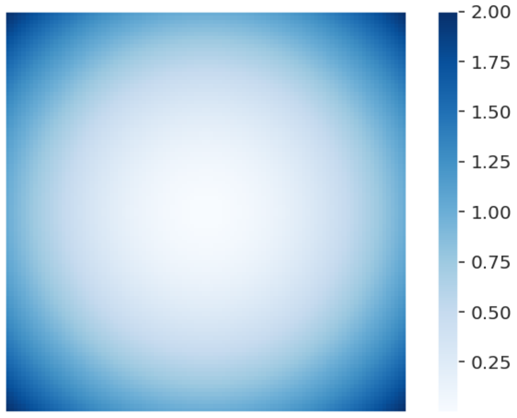
Question 5

In [27]:

```
import numpy as np

# changing the number of spaces between the values -1 and 1 to 100 from 10
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

# Dataset representation
# The newly generated data with 100x100 values
df = pd.DataFrame({'x': X.reshape(-1), 'y': Y.reshape(-1), 'z = f(x,y)': Z.reshape(-1)})
```

In [28]:

```
# 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([])
plt.yticks([])

plt.show()
```



On changing the number of evenly spaced values from 10 to 100 for both the variables, it can be seen that the resulting heatmap is smoother than the one obtained when lesser values were used. This gives used a basic relation that states that, more the number of data points, smoother the heatmap. Furthermore, on increasing the number of data points, it becomes difficult to visually calculate the magnitude of distance between data points as now the size of each pixel becomes significantly smaller.

Question 6

6.1

In [29]:

```
#importing the wine dataset from sklean.dataset
from sklearn.datasets import load_wine
data = load_wine()

#loading the dataset into a dataframe
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = pd.Series(data.target)
df
```

Out[29]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82
...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35

178 rows × 14 columns

In [30]:

```
#displaying the freruency of each value of the attribute target from the dataframe

df['target'].value_counts()
```

Out[30]:

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

6.2

In [31]:

```
#the method describe gives a univariate analysis of the dataset with all the metrics like
the mean,counts,min,max
df_new = df.drop(['target'], axis=1)
df_new.describe()
```

Out[31]:

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

In [32]:

```
#the multivariate summary is the correlation between the attributes
print("\nCorrelation coefficients:")
display(df_new.corr())
```

```
Correlation coefficients:
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.289101	0.236815	
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	-0.335167	-0.411007	
ash	0.211545	0.164045	1.000000	0.443367	0.286587	0.128980	0.115077	
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	-0.321113	-0.351370	
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	0.214401	0.195784	
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	1.000000	0.864564	
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	0.864564	1.000000	

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	-0.449935	-0.537900	
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	0.612413	0.652692	
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	-0.055136	-0.172379	
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	0.433681	0.543479	
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	0.699949	0.787194	
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	0.498115	0.494193	

In [33]:

```
#grouping the dataset by the attribute "target" and calculating median
group_df = df.groupby('target').median()
group_df
```

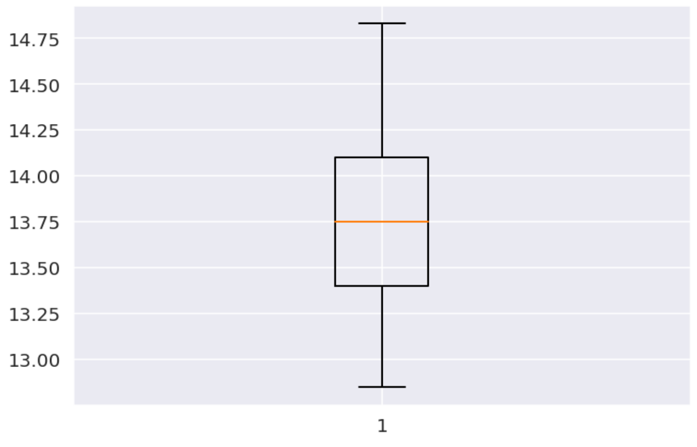
Out[33]:

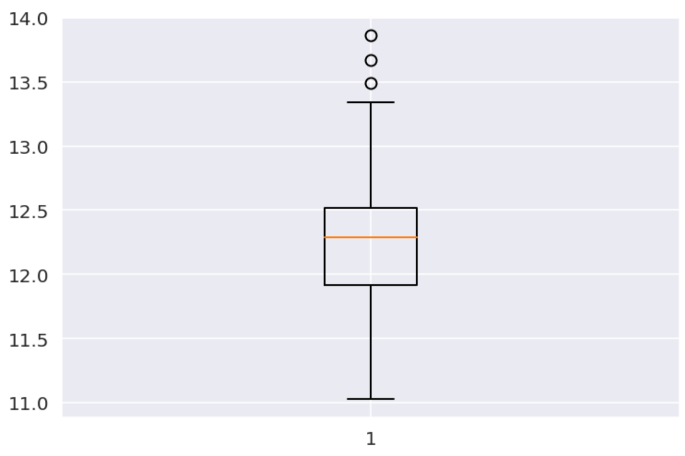
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyani
target									
0	13.750	1.770	2.44	16.8	104.0	2.800	2.980	0.29	1.8
1	12.290	1.610	2.24	20.0	88.0	2.200	2.030	0.37	1.6
2	13.165	3.265	2.38	21.0	97.0	1.635	0.685	0.47	1.1

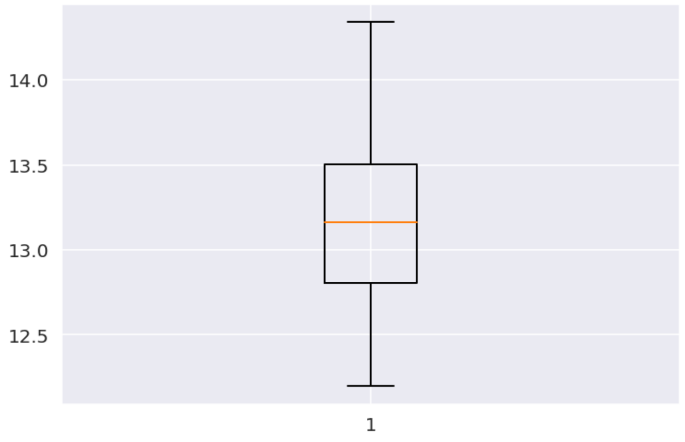
6.3

In [44]:

```
#grouping the dataframe according to the attribute target
new_df = df.groupby(["target"])

# storing categories of the attribute target to separate variables in order to make box p
lots
var1 = new_df.get_group(0)
var2 = new_df.get_group(1)
var3 = new_df.get_group(2)


plt.boxplot(var1['alcohol'])
plt.show()

plt.boxplot(var2['alcohol'])
plt.show()

plt.boxplot(var3['alcohol'])
plt.show()
```







6.4

In [35]:

```
# displaying correlation between different pairs and sorting them from lowest to highest
# from looking at the values, the pair with highest correlation is flavanoids and total p
henols
# hence plotting a scatter plot between the two

df.corr().unstack().sort_values().drop_duplicates()
```

Out[35]:

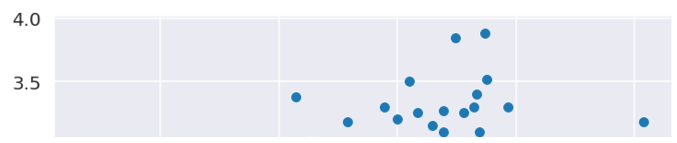
```
flavanoids                    target            -0.847498
od280/od315_of_diluted_wines  target            -0.788230
target                        total_phenols     -0.719163
proline                       target            -0.633717
hue                           target            -0.617369
      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

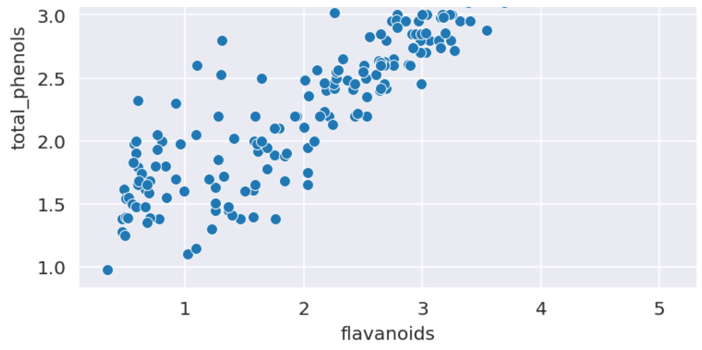
In [36]:

```
# scatter plot between the two highest correlated pairs

sns.scatterplot(x = 'flavanoids', y= 'total_phenols', data=df)
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0e1df30890>
```





6.5

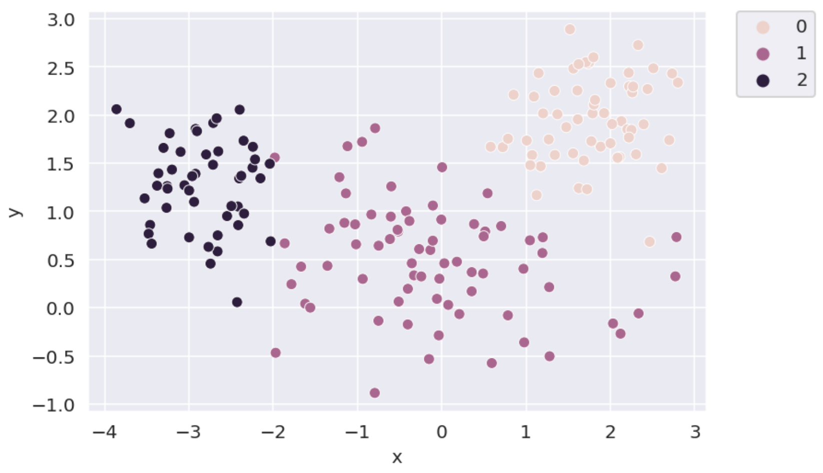
In [40]:

```
# from sklearn importing the Standard Scalar library for performing standardization
from sklearn.preprocessing import StandardScaler

df2 = df.drop(columns=['target'])
df_sorted = df.sort_values(by='target', ascending=True)

sort_t = df_sorted['target']
var = df_sorted.drop(columns='target').to_numpy()
print(var)

#z-score standardization

scale = StandardScaler()
var = scale.fit_transform(var)
print(var)
```

```
[[1.423e+01 1.710e+00 2.430e+00 ... 1.040e+00 3.920e+00 1.065e+03]
 [1.368e+01 1.830e+00 2.360e+00 ... 1.230e+00 2.870e+00 9.900e+02]
 [1.376e+01 1.530e+00 2.700e+00 ... 1.250e+00 3.000e+00 1.235e+03]
 ...
 [1.350e+01 3.120e+00 2.620e+00 ... 5.900e-01 1.300e+00 5.000e+02]
 [1.311e+01 1.900e+00 2.750e+00 ... 6.100e-01 1.330e+00 4.250e+02]
 [1.413e+01 4.100e+00 2.740e+00 ... 6.100e-01 1.600e+00 5.600e+02]]
[[ 1.51861254 -0.5622498   0.23205254 ...  0.36217728  1.84791957
   1.01300893]
 [ 0.83921681 -0.45453022 -0.02382132 ...  1.19577163  0.36485461
   0.7741719 ]
 [ 0.93803801 -0.72382916  1.21899459 ...  1.28351841  0.54847218
   1.55437286]
 ...
 [ 0.61686912  0.70345524  0.92656731 ... -1.61212515 -1.85268061
  -0.78623004]
 [ 0.13511578 -0.3916938   1.40176163 ... -1.52437837 -1.81030733
  -1.02506707]
 [ 1.39508604  1.58316512  1.36520822 ... -1.52437837 -1.42894777
  -0.59516041]]
```

In [43]:

```
# importing the library MDS from sklearn

from sklearn.manifold import MDS

# defining the number of components for multi dimensional scaling (i.e 2)
e = MDS(n_components=2)

x = embedding.fit_transform(var)

projected_data = pd.DataFrame({'x': x[:, 0],'y': x[:, 1], 'target': sort_t})

# plotting the scatterplot/ projection
sns.scatterplot(x='x', y='y', hue='target', data = projected_data)
plt.legend (bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt. show()
```



```
In [ ]:
```