DATA MINING PROJECT

1) The function "load_wine" from "sklearn.datasets" can be used to load the wine dataset into a "DataFrame" by using the below commands:

```
import pandas as pd
data = load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = pd.Series(data.target)
df.info()
```

```
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    int32
dtypes: float64(13), int32(1)
```

a) Load the wine dataset. Which feature is categorical and why? Compute the frequency (not the occurrence) of each value of the categorical feature. Include the code in your report.

We know that, categorical feature of a dataset means it has certain repeated values and we can group the data based on that. In the given wine dataset, the target feature is categorical because it has the values 0,1 and 2 repeatedly occurring in the dataset.

Code:

```
frequency = df["target"].value_counts()
print('Frequency of Categorical Value :\n', frequency)
```

b) Compute two different univariate and two different multivariate summaries for all numerical features. Include the code in your report.

```
Frequency of Categorical Value :
 1    71
0    59
2    48
Name: target, dtype: int64
```

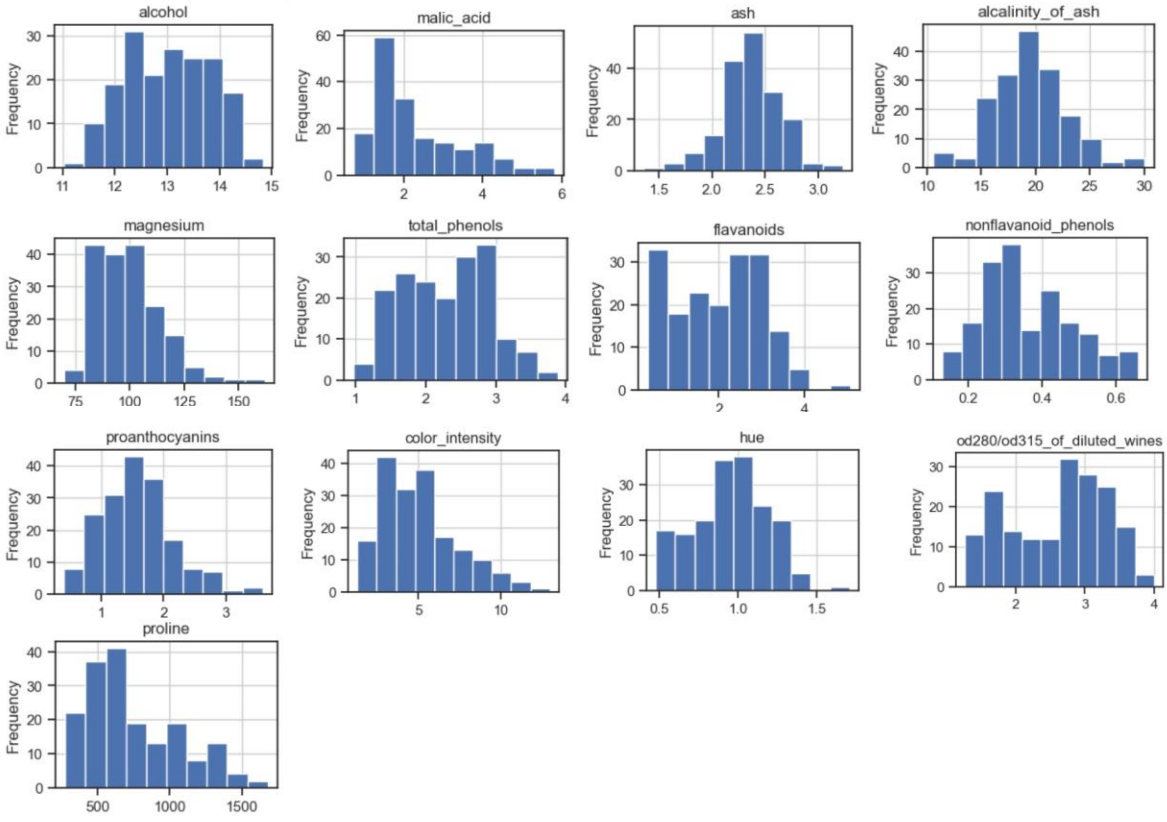
Univariate Summaries:

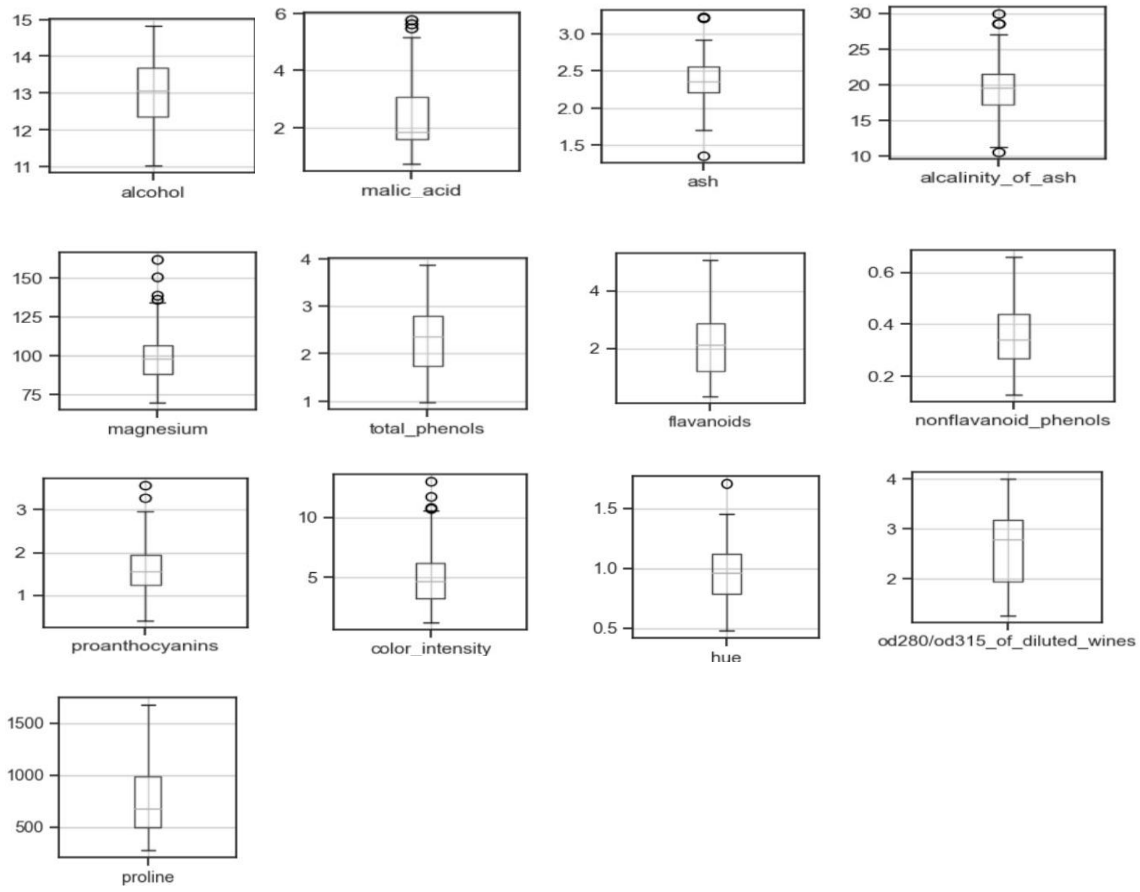
```
import pandas as pd
import matplotlib.pyplot as plt import seaborn as

sns numerical_values = df.loc[:, df.columns !=

'target'] numerical_values.describe()

for column in numerical_values:
    plt.figure(figsize= (3,2))
    df[column].plot.hist(title=f'{column}', bins= 10)
    plt.grid(True) plt.show()

for column in numerical_values:
    plt.figure(figsize= (2,2))
    df.boxplot(column=column)
    plt.show()
```

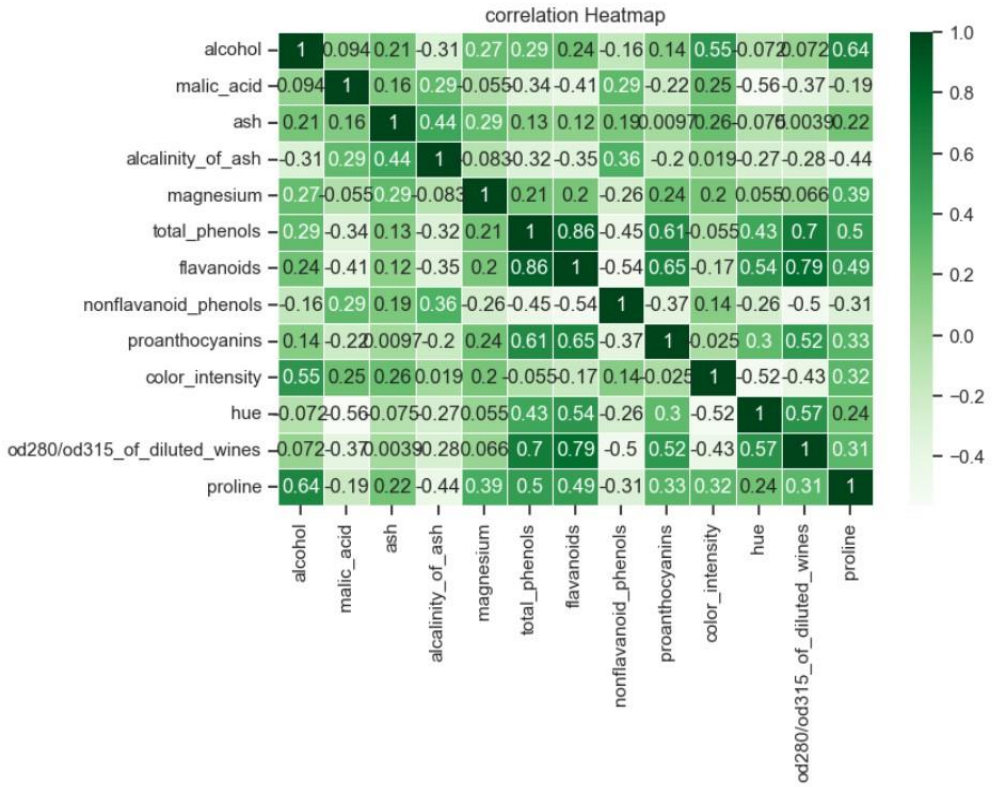
Multivariate Summaries:

```
correlation_matrix = numerical_values.corr()

sns.set(style="ticks")
sns.pairplot(numerical_values)
plt.show()

plt.figure(figsize=(8,5)) sns.heatmap(correlation_matrix, linewidths=0.5,
annot=True, cmap='Greens') plt.title("correlation Heatmap") plt.show()
```

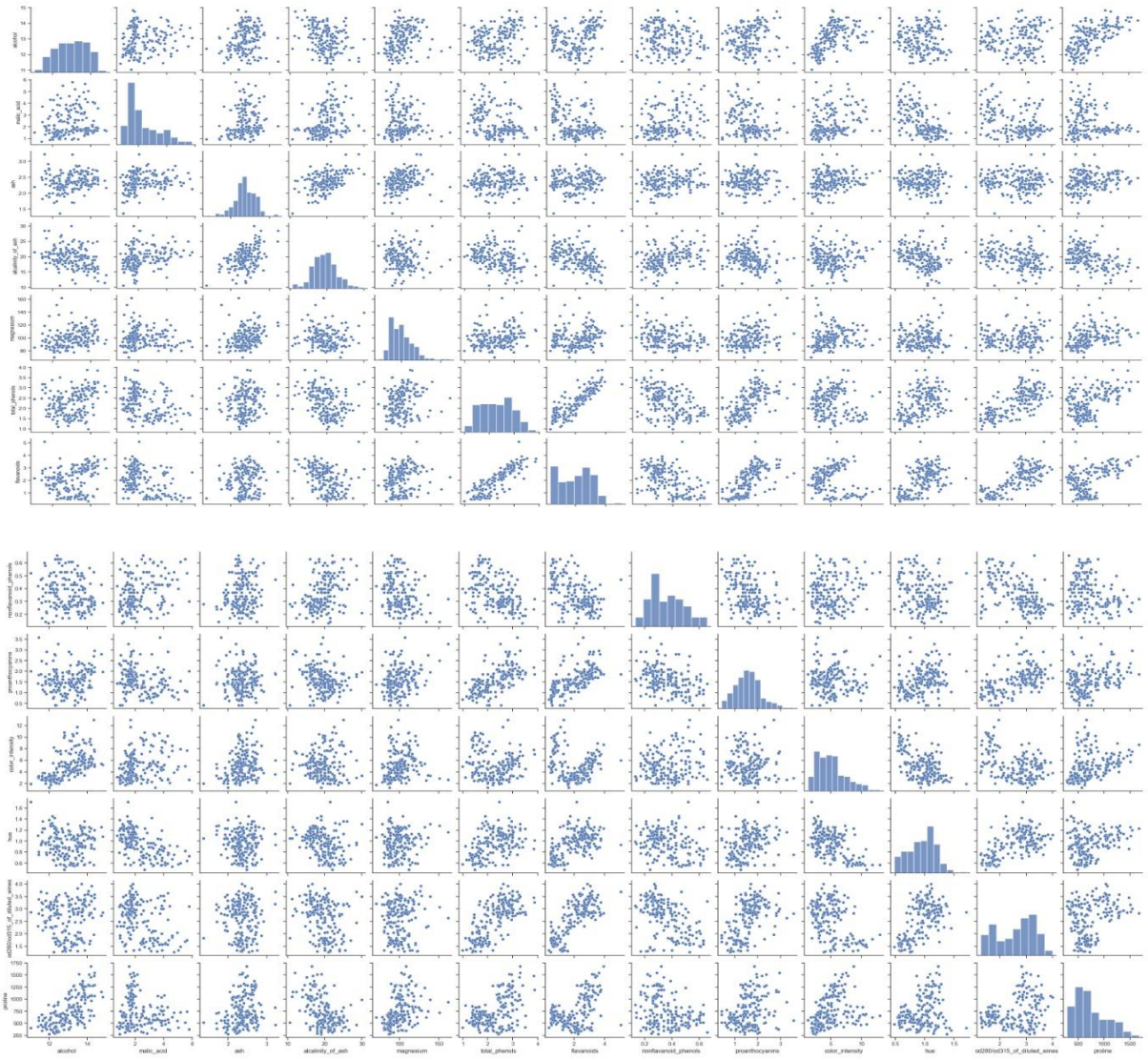
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	5.058090
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318286
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280000
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220000
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690000
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200000
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000000


alcohol
malic_acid
ash
alcalinity_of_ash
magnesium
total_phenols
flavanoids
nonflavanoid_phenols
proanthocyanins
color_intensity
hue
od280/od315_of_diluted_wines
proline
Frequency


alcohol
malic_acid
ash
alcalinity_of_ash
magnesium
total_phenols
flavanoids
nonflavanoid_phenols
proanthocyanins
color_intensity
hue
od280/od315_of_diluted_wines
proline



	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocya
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.289101	0.236815	-0.155929	0.136
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	-0.335167	-0.411007	0.292977	-0.220
ash	0.211545	0.164045	1.000000	0.443367	0.286587	0.128980	0.115077	0.186230	0.009
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	-0.321113	-0.351370	0.361922	-0.197
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	0.214401	0.195784	-0.256294	0.236
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	1.000000	0.864564	-0.449935	0.612
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	0.864564	1.000000	-0.537900	0.652
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	-0.449935	-0.537900	1.000000	-0.365
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	0.612413	0.652692	-0.365845	1.000
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	-0.055136	-0.172379	0.139057	-0.025
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	0.433681	0.543479	-0.262640	0.295
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	0.699949	0.787194	-0.503270	0.519
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	0.498115	0.494193	-0.311385	0.330



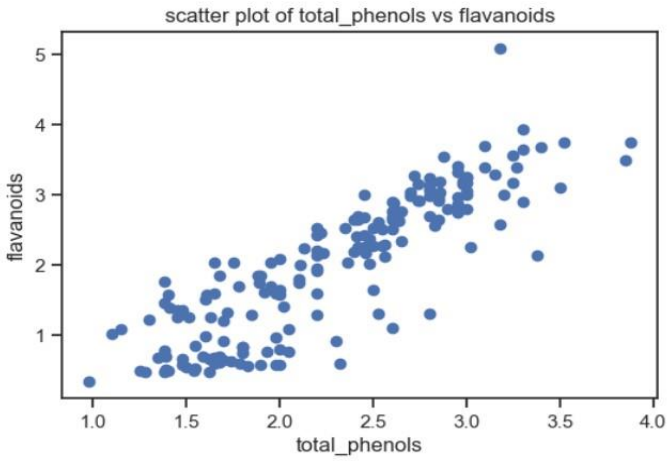
c) Group observations by the categorical feature & compute the corresponding median for each remaining numerical feature. Include the code in your report.

```
median = df.groupby('target').median()
print(median)
```

```
        alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  \
target
0        13.750       1.770  2.44               16.8      104.0
1        12.290       1.610  2.24               20.0       88.0
2        13.165       3.265  2.38               21.0       97.0

        total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
target
0               2.800       2.980                  0.29            1.870
1               2.200       2.030                  0.37            1.610
2               1.635       0.685                  0.47            1.105

        color_intensity    hue  od280/od315_of_diluted_wines  proline
target
0                  5.40  1.070                          3.17   1095.0
1                  2.90  1.040                          2.83    495.0
2                  7.55  0.665                          1.66    627.5
```

d) Create a scatter plot for the pair of distinct numerical features with the highest correlation. Include the code in your report.

```
correlation = numerical_values.corr() maximum_correlation =
correlation.unstack().sort_values(ascending=False) maximum_correlation_pair =
maximum_correlation[(maximum_correlation < 1.0) &
(maximum_correlation > 0.5)].index[0]
column_1,column_2 = maximum_correlation_pair

plt.figure(figsize=(6,4)) plt.scatter(numerical_values[column_1],
numerical_values[column_2], cmap='viridis') plt.xlabel(column_1)
plt.ylabel(column_2) plt.title(f' scatter plot of {column_1} vs {column_2} ') plt.show()
```



2) Consider the following sales data: [5, 20, 1, 6, 13, 8, 9, 11, 17, 7, 2, 12] Apply the following binning techniques on the data, assuming 3 bins in each case:

- **Equal-frequency binning**
- **Smoothing by bin boundaries**

A) EQUAL FREQUENCY BINNING

As we know, if we apply Equal-Frequency Binning Technique to a dataset, the data present in the dataset is divided into bins in such a way that each bins contains approximately the equal number of data points. So, In our case, we are assuming there are 3 bins, We can perform the binning in following way,

Manual Method:

a) We first assemble the data in the increasing order.

[1,2,5,6,7,8,9,11,12,13,17,20]

b) Then calculation of the number of data points per bin takes place. (Here we assume the number of bins to be 3).

Length_of_data - Number_of_bins

12 - 3 = 4 Data Points per Bin

c) At last we need to determine the bin boundaries using the above calculated number of data points per bin.

Answer:

Bin 1 = 1,2,5,6

Bin 2 = 7,8,9,11

Bin 3 = 12,13,17,20

B) SMOOTHING BY BINNING TECHNIQUES

Smoothing technique is used to analyze the data and to reduce the noise and unwanted fluctuations present in the data.

Manual Method :

a) Diving the data into intervals, the bins should be of equal range, in our case it is 3.

b) Assign the bins with respective data that falls into.

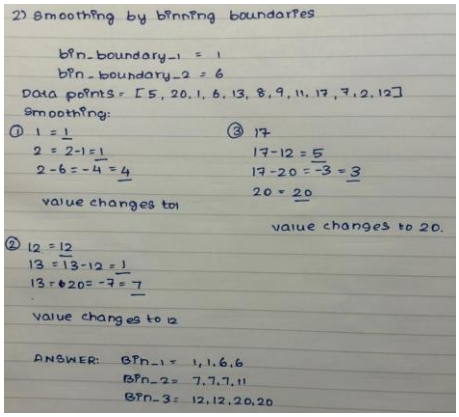
c) According to the statistical summary, (that is mean, median …) replace the data present in each bin.

Answer:

Bin 1 = 1,1,6,6

Bin 2 = 7,7,7,11

Bin 3 = 12,12,20,20



3) Load the file country-income.csv which includes numerical and categorical features. Perform data cleaning to replace any NaN values with the mean value of that particular feature. Then replace any categorical features with numerical features. Display the resulting dataset. You can use the sklearn.impute and sklearn.preprocessing packages to assist you. Include the code in your report. from sklearn.preprocessing import OneHotEncoder from sklearn.impute import SimpleImputer from sklearn.preprocessing import LabelEncoder import pandas as pd import numpy as np

```
df = pd.read_csv('./country-income.csv')
mean = SimpleImputer(strategy='mean')
dataframe = df[['Age','Income']]
data_cleaning = mean.fit_transform(dataframe)
data_cleaning = pd.DataFrame(data_cleaning, columns=dataframe.columns)
df[['Age','Income']] = data_cleaning[['Age','Income']]

region = df['Region'].to_numpy()
online_shopper = df['Online Shopper'].to_numpy()

encoder = OneHotEncoder()
region_1 = encoder.fit_transform(region.reshape(-1,1))
online_shopper_1 = encoder.fit_transform(online_shopper.reshape(-1,1))

region_df = pd.DataFrame(region_1.toarray(), columns =['Brazil','India','USA'])
online_shopper_df = pd.DataFrame(online_shopper_1.toarray(), columns =['Online Shopper - No',
'Online Shopper - No'])

df =pd.concat([df, region_df, online_shopper_df], axis=1)
df =df.drop(columns = ['Region', 'Online Shopper']) df
```

	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

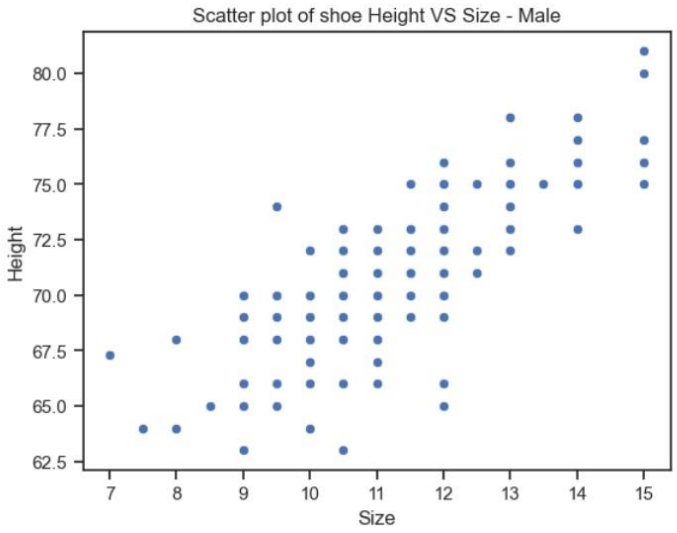
	Age	Income	Brazil	India	USA	Online Shopper - No	Online Shopper - No
0	49.000000	86400.000000	0.0	1.0	0.0	1.0	0.0
1	32.000000	57600.000000	1.0	0.0	0.0	0.0	1.0
2	35.000000	64800.000000	0.0	0.0	1.0	1.0	0.0
3	43.000000	73200.000000	1.0	0.0	0.0	1.0	0.0
4	45.000000	76533.333333	0.0	0.0	1.0	0.0	1.0
5	40.000000	69600.000000	0.0	1.0	0.0	0.0	1.0
6	43.777778	62400.000000	1.0	0.0	0.0	1.0	0.0
7	53.000000	94800.000000	0.0	1.0	0.0	0.0	1.0
8	55.000000	99600.000000	0.0	0.0	1.0	1.0	0.0
9	42.000000	80400.000000	0.0	1.0	0.0	0.0	1.0

4) Load the file shoesize.csv, which includes measurements of shoe size and height (in inches) for 408 subjects, both female and male. Plot the scatterplots of shoe size versus height for female and male subjects separately. Compute the Pearson's correlation coefficient of shoe size versus height for female and male subjects separately. What can be inferred by the scatterplots and computed correlation coefficients? Include the code in your report.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('./Shoesize.csv')
```

```
female = df.loc[df['Gender'] == 'F']
male = df.loc[df['Gender'] == 'M']

correlation_for_male = male['Size'].corr(male['Height'])
correlation_for_female = female['Size'].corr(female['Height'])
print('Correlation for male : ',correlation_for_male)
print('Correlation for female : ',correlation_for_female)

sns.scatterplot(data=male, x='Size', y='Height')
plt.title('Scatter plot of shoe Height VS Size - Male')
plt.xlabel('Size') plt.ylabel('Height') plt.show()

sns.scatterplot(data=female, x='Size', y='Height')
plt.title('Scatter plot of Shoe Height VS Size - Female')
plt.xlabel('size') plt.ylabel('Height') plt.show()
```

```
Correlation for male:  0.7677093547300968
Correlation for female:  0.7078119417143995
```



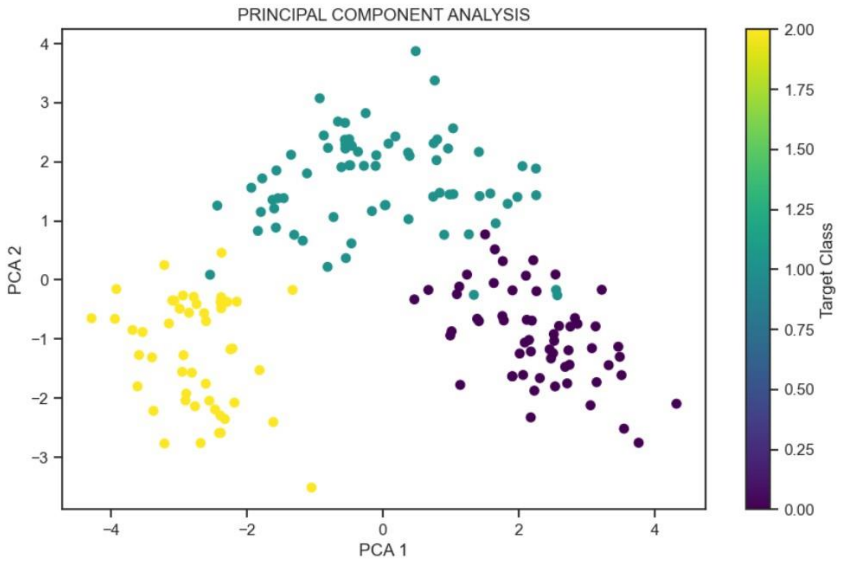
Inference :

So, by looking up into the scatter plots, I can observe that for both male and female, there is a strong positive correlation between Shoe Height and Size. But when I look at the correlation values for females and males as separate, I can note that correlation value of male is somewhat higher than the female. So I can conclude that the shoe attribute of male has a stronger relationship than females.

5) Using the wine dataset from question 1, perform Principal Component Analysis (PCA) with 2 components. Transform the data and plot the scatterplot of all samples along the two principal components, color-coded according to the "target" column (this column is the class and should not be used in the PCA analysis). What insights can you obtain by viewing the scatterplot of the principal components? Can you easily distinguish the samples that belong to one class from the samples that belong to another class and so on? In other words, are the different classes (quite) distinctive from the other, or is there a lot of overlap? If it is the latter, then why is this happening? What can be done to the data prior to performing PCA in order to

alleviate this issue? Do this action first and then perform PCA with 2 components, transform the data and plot the scatterplot of all samples along these two principal components, color-coded according to the "target" column. Now are the different classes (quite) distinctive from the other? Include the code in your report.

```
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_wine from
sklearn.decomposition import PCA import pandas
as pd
import matplotlib.pyplot as plt

data = load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = pd.Series(data.target)

numerical_values = df.drop('target', axis = 1)
target_df = df['target']

scaler = StandardScaler()
numerical_values_scaled = scaler.fit_transform(numerical_values)

pca = PCA(n_components=2)
numerical_values_pca = pca.fit_transform(numerical_values_scaled)

plt.figure(figsize=(10,6))
plt.scatter(numerical_values_pca[ :, 0], numerical_values_pca[ :, 1], c=target_df, cmap='viridis')
plt.xlabel('PCA 1') plt.ylabel('PCA 2')
plt.title('PRINCIPAL COMPONENT
ANALYSIS') plt.colorbar(label = 'Target Class')
plt.show()
```



After analyzing the scatter plot, I can say that no data preprocessing is required in this case because the classes are little overlapping and distinct. The little overlap here is because of low variance of the data compared to the main cluster. The data points that are very outer have less variance so that it is overlapping. But I can see that fewer points are only lying very outer so, we can exclude that compared to the other points that are related very closely.

6) In Lab session 3 (Data Exploration and Data Visualisation), in subsection 1.9 you had created and visualized a heatmap for the distance matrix for the graduation_rate.csv. You may have noticed that the distance matrix visualization is not very informative. However, it is still possible to infer that the 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. Explain how this inference is possible from the visualization.

In the data exploration and data visualization part, we have visualized a heatmap for the distance matrix, and yes it is not very informative. In the visualization, the darker colors are showing the larger distance or the dissimilarity between the other data points, on the other hand, the lighter colors indicate the lesser distance or the similarity between the other data points. So, by looking into the heatmap, I can observe it has a darker color so it inferred that the data points are dissimilar from the other data points and in the larger distance.

From this inference, we can get that the dark color is the distance between the parents of students who only have some high school education and the parents of students who hold a master's degree, because the data points are not similar to one another and in the larger distance too. On the other hand, the lighter color represents the distance between the parents of students who have only some high school education, this can be inferred as the data points seem to be similar with each other.

7) Use the file country-income.csv and perform the following: a) Load the CSV file using Cubes, create a JSON file for the data cube model, and create a data cube for the data. Use as dimensions the region, age, and online shopper fields. Use as measure the income. Define

aggregate functions in the data cube model for the total, average, minimum, and maximum income. Include the code in your report (and show the files created). [8 marks] b) Using the created data cube and data cube model, produce aggregate results for: i) the whole data cube; ii) results per region; iii) results per online shopping activity; and iv) results for all people aged between 40 and 50.

```
#install cubes and sqlalchemy
!pip install cubes
!pip install sqlalchemy==1.3.20

#Importing libraries
from sqlalchemy import create_engine

import collections.abc

collections.MutableMapping = collections.abc.MutableMapping
collections.Mapping = collections.abc.Mapping
collections.Iterable = collections.abc.Iterable
collections.MutableSet = collections.abc.MutableSet
collections.Callable = collections.abc.Callable

from cubes.tutorial.sql import create_table_from_csv

#loading the data and creating table engine =
create_engine('sqlite:///data.sqlite')
create_table_from_csv(engine, "country-
income.csv", table_name="country_income_cube",
fields=[
                    ("region", "string"),
                    ("age", "integer"),
                    ("income", "integer"),
                    ("online_shopper", "string")],
                 create_id=True
                )
#specifying data store from
cubes import Workspace
```

```
workspace = Workspace()
workspace.register_default_store("sql", url="sqlite:///data.sqlite")

#importing JSON file
workspace.import_model("cube.json")

#creating cube
cube = workspace.cube("country_cube")

#creating browser browser =
workspace.browser(cube)
```

JSON FILE

```
{
    "dimensions": [
        {
            "name": "region",
            "levels": [
                {
                    "name": "region",
                    "label": "Region"
                }
            ]
        },
        {
            "name": "age",
            "levels": [
                {
                    "name": "age",
                    "label": "Age"
                }
            ]
        },
        {
```

```
        "name": "online_shopper",
        "levels": [ {
                "name": "online_shopper",
                "label": "Online Shopper"
            }
        ]
    }
],
"cubes": [
    {
        "name": "country_income_cube",
        "dimensions": ["region", "age", "online_shopper"],
        "measures": [ {
                "name": "income",
                "label": "income"
            }
        ],
        "aggregates": [
            {
                "name": "total_income",
                "function": "sum",
                "measure": "income"
            },
            {
                "name": "average_income",
                "function": "avg",
                "measure": "income"
            },
            {
                "name": "min_income",
                "function": "min",
                "measure": "income"
            },
            {
                "name": "max_income",
```

```
                    "function": "max",
                    "measure": "income"
                }
            ]
        }
    ]
}
```

b) Using the created data cube and data cube model, produce aggregate results for:

i) the whole data cube

#Aggregate results for whole cube

result = browser.aggregate()

result.summary

```
{'total_income': 688800.0,
 'average_income': 68880.0,
 'min_income': 57600,
 'max_income': ''}
```

ii) results per region

#aggregate results per region result =

browser.aggregate(drilldown=["region"]) for

record in result:

print(record)

```
{'region': 'Brazil', 'total_income': 193200, 'average_income': 64400.0, 'min_income': 57600, 'max_income': 73200}
{'region': 'India', 'total_income': 331200, 'average_income': 82800.0, 'min_income': 69600, 'max_income': 94800}
{'region': 'USA', 'total_income': 164400.0, 'average_income': 54800.0, 'min_income': 64800, 'max_income': ''}
```

iii) results per online shopping activity

#aggregate results per online shopping activity result

= browser.aggregate(drilldown=["online_shopper"]) for

record in result: print(record)

```
{'online_shopper': 'No', 'total_income': 386400, 'average_income': 77280.0, 'min_income': 62400, 'max_income': 99600}
{'online_shopper': 'Yes', 'total_income': 302400.0, 'average_income': 60480.0, 'min_income': 57600, 'max_income': ''}
```

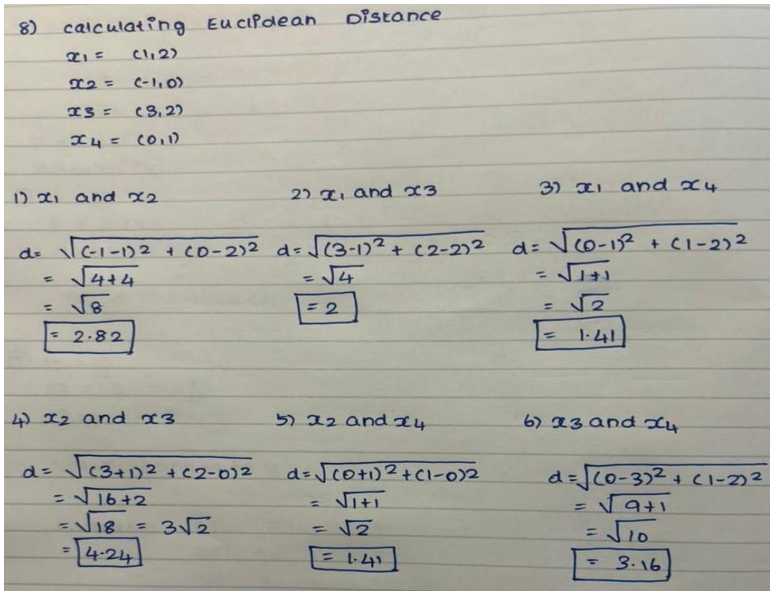
iv) results for all people aged between 40 and 50

#aggregate results for ages 40 to 50

```
import cubes as cubes
range_cut = [cubes.RangeCut("age", [40], [50])]
cubes_cell = cubes.Cell(cube, range_cut)
solution = browser.aggregate(cubes_cell,drilldown=["age"])
print(solution.summary)
```

```
{'total_income': 309600.0, 'average_income': 61920.0, 'min_income': 69600, 'max_income': ''}
```

8) Consider a dataset that contains only two observations x1 = (1,2) and x2 = (−1,0). Suppose that the class of the first observation is y1 = 1 and that the class of the second observation is y2 = 0. How would a 1-nearest neighbor classifier based on the Euclidean distance classify the observation x3 = (3,2) and why? How would the same classifier classify the observation x4 = (0,1) and why?



In the calculation above,

- The distance between x3 and x1 < x3 and x2, so x3 belongs to y1.
- The distance between x1 and x2 = x2 and x4, so here the x4 can either be in y1 or in y2, because it is in the center of both the classes.