Data Mining Assignment – 1

**We will import all the libraries required in only one cell so that it is much better understandable to the viewer about the libraries we have imported.**

import numpy as np

import pandas as pd

from sklearn.datasets import load\_wine

from seaborn import pairplot

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from cubes.tutorial.sql import create\_table\_from\_csv

from sqlalchemy import create\_engine

import cubes as cubes

from cubes import Workspace

## Question 1:

**The function “load\_wine” from “sklearn.datasets” can be used to load the wine dataset into a “DataFrame” by using the below commands:**

**data = load\_wine()**

**df = pd.DataFrame(data.data, columns=data.feature\_names)**

**df['target'] = pd.Series(data.target)**

**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.**

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

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

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

### We will now load the wine dataset as required by using the given commands:

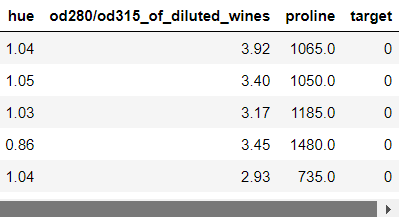
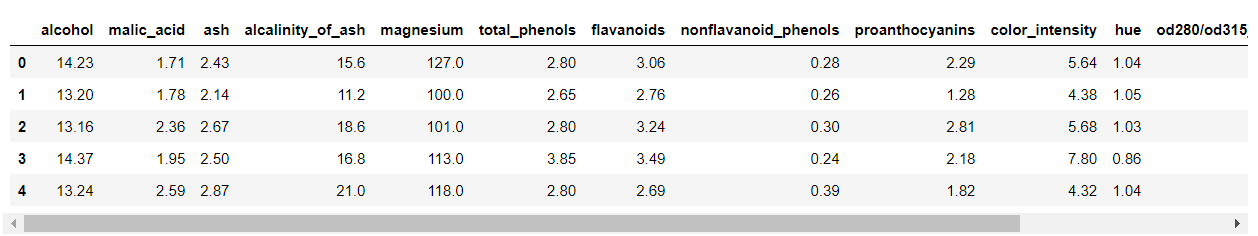
data = load\_wine()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = pd.Series(data.target)

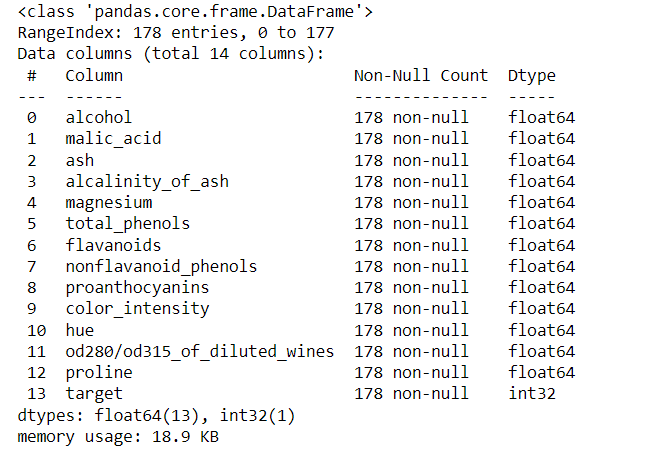
Let us check just the first 5 rows of our dataset:

df.head()



Now, in order to deliver the answer to the first (a) part of the question 1, we will first look at the data types of all the features present in the wine dataset.

df.info()



Here we can see that most of the features (13) have datatype as float and one has integer data type. By this we can see that there are no categorical features. As by the definition of a categorical feature, all the non-numerical features are comprised in categorical feature, we can easily say that there are no categorical features in this dataset as all the features are in a numerical format.

## BUT

Isn't the feature "target" seems a bit different to you? This is because, the “target” feature has 3 classes, due to what the “target” feature becomes our categorical feature. Let us try to calculate the frequencies using this target feature. We will need to segregate the target into different classes, and then find out the frequencies of each class.

unique\_classes = df['target'].unique()

# Get the class frequencies

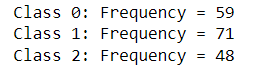
frequencies = df['target'].value\_counts()

# Display the unique classes and their frequencies

for i in unique\_classes:

frequency = frequencies[i]

print(f"Class {i}: Frequency = {frequency}")



We can see that the target has 3 classes which in turn makes it as the categorical feature and the classes have different frequencies in them.

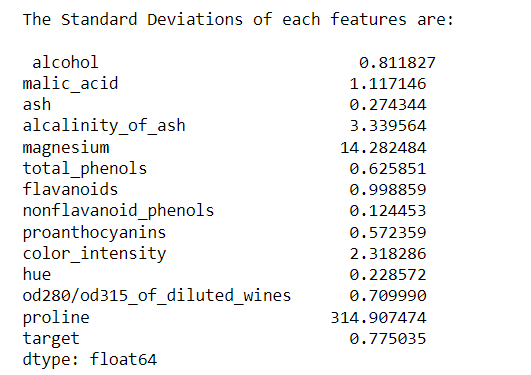
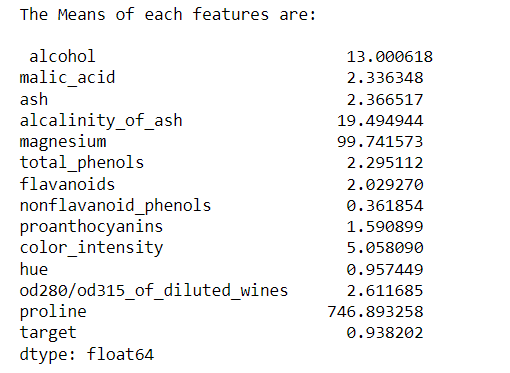
## We have completed the first part of the question 1!!

Now, in order to deliver the answer to the second (b) part of the question 1, we will first determine which two univariate summaries and two multivariate summaries are we planning to compute.

Let us compute mean and standard deviation univariate summaries, while for multivariate summaries, we can compute correlational matrix and a pairplot.

print("The Means of each features are: \n\n",df.mean(),"\n")

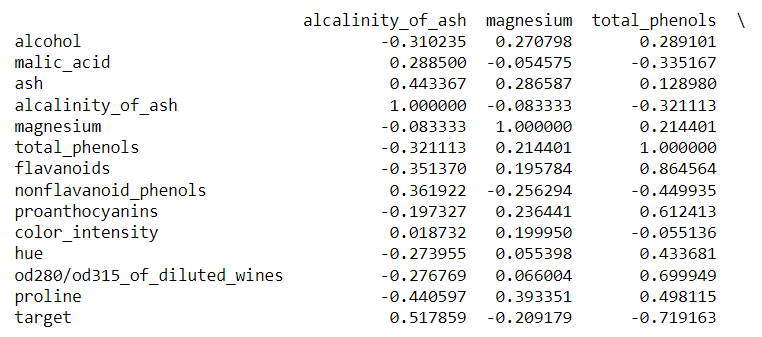
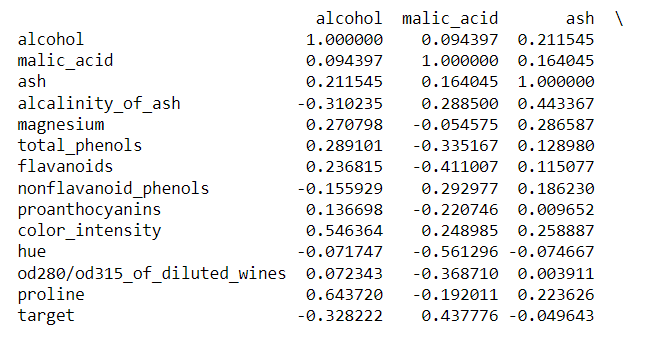
print("The Standard Deviations of each features are: \n\n",df.std())

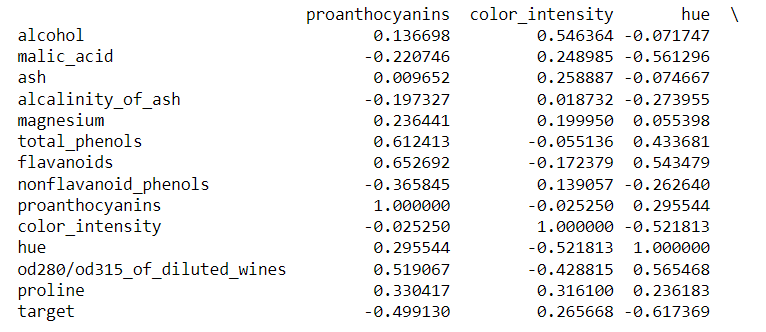
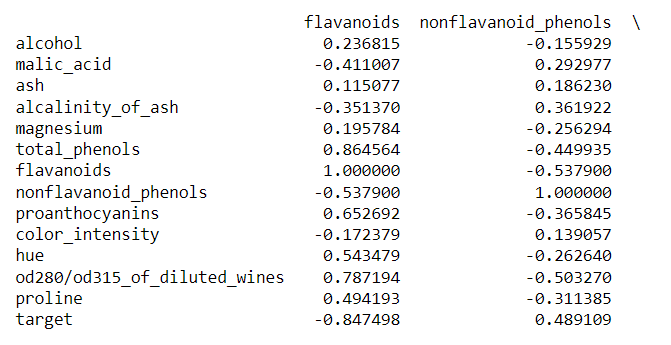


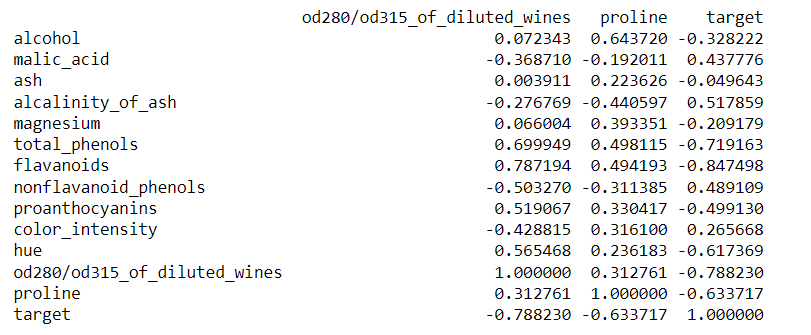
We can see all the means and standard deviations for each feature in the wine dataset.

correlation\_matrix = df.corr()

print(correlation\_matrix)



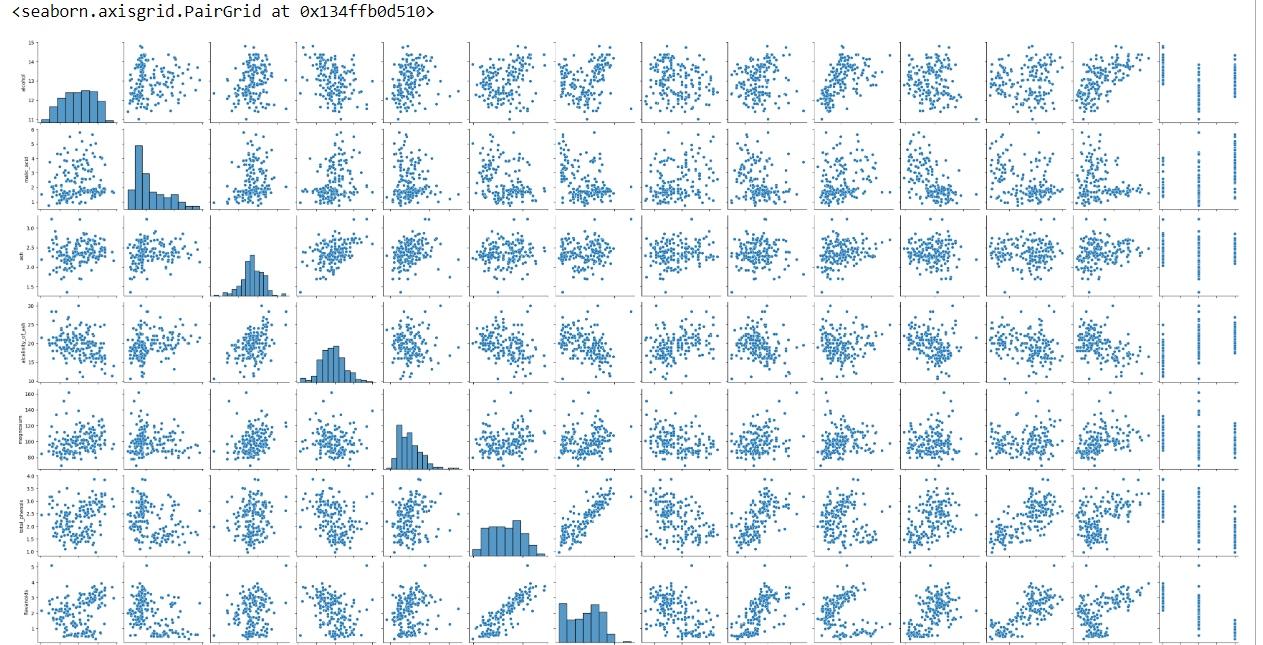


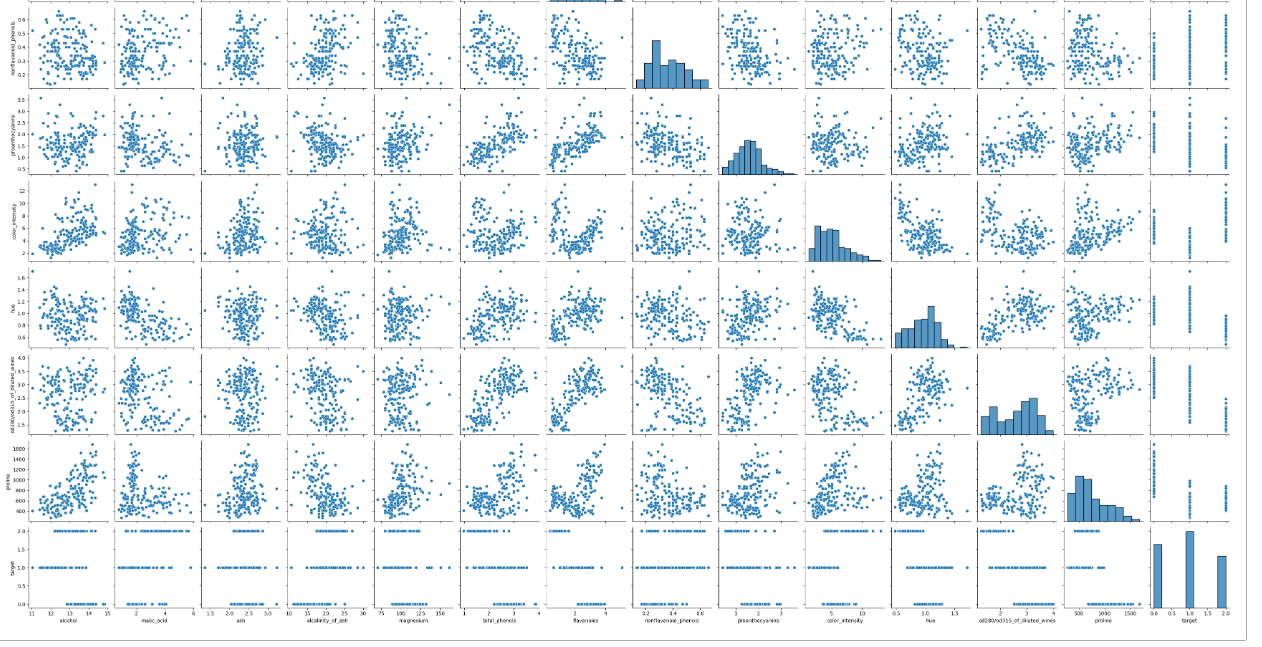


We can see the correlation of each feature with every other feature within the dataset. This is kind of difficult to read right? This is why we will now visually represent it by using a pair plot.

pairwise\_scatter = pairplot(df)

pairwise\_scatter





You will notice the histograms in the diagonal of the pairplot, this is because the histograms are the representations of the univariate summaries of each feature as in the part of the diagonal, we can find the paring of each feature with itself.

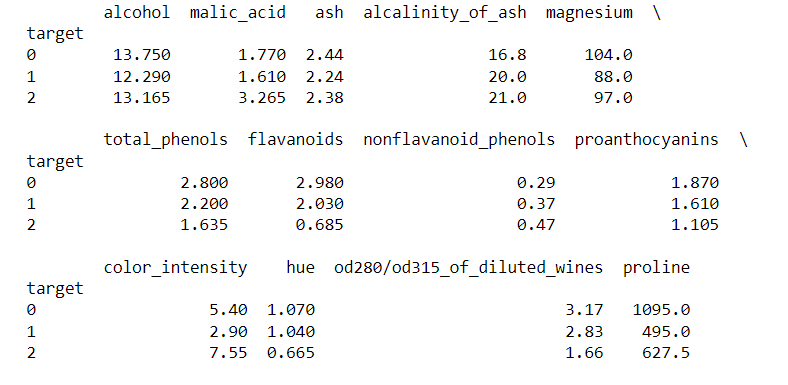
## We have completed the second part of the question 1!!

Now, in the third (c) part of the question 1 we need to group all of our observations with respect to the categorical feature. We only have one categorical feature that is the target feature, so let’s group our other features with respect to the target feature and find the median for each of the features.

grouped\_target = df.groupby('target')

grouped\_medians = grouped\_target.median()

print(grouped\_medians)



This is how we can now see the medians of each feature with respect to our categorical feature.

## We have completed the third part of the question 1!!

In the fourth part of the question 1 we will need to calculate the correlations of each feature with every other feature and then plot a scatter plot for the two features having the largest correlation among them.

# Calculate the correlation matrix

correlation\_matrix = df.corr()

# Find the pair of distinct features with the highest correlation

maximum\_correlation = 0

Column1 = []

Column2 = []

for i in range(len(correlation\_matrix.columns)):

for j in range(i):

if (correlation\_matrix.iloc[i, j]) > maximum\_correlation:

maximum\_correlation = (correlation\_matrix.iloc[i, j])

Column1, Column2 = correlation\_matrix.columns[i], correlation\_matrix.columns[j]

# Create a scatter plot for the pair of features with the highest correlation

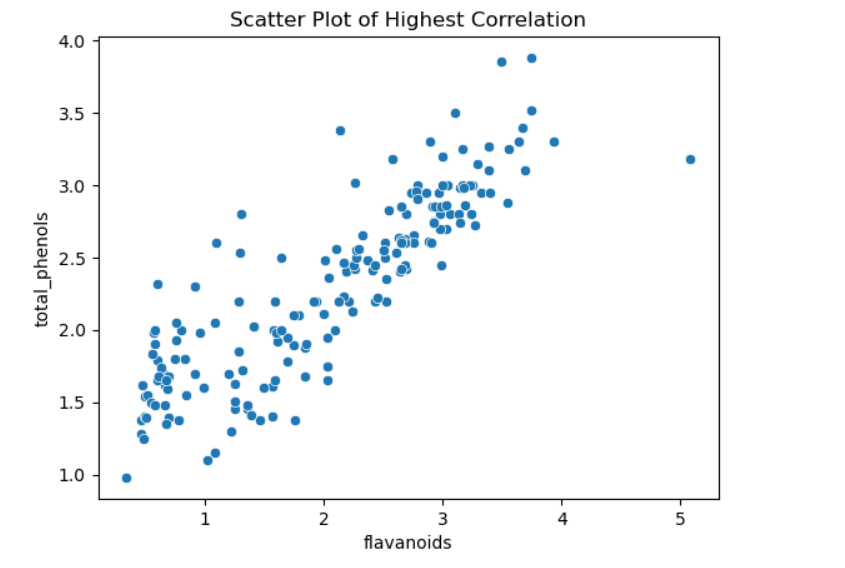
sns.scatterplot(x=Column1, y=Column2, data=df)

plt.title("Scatter Plot of Highest Correlation")

plt.xlabel(Column1)

plt.ylabel(Column2)

plt.show()



By this plot we can directly see that the 'total\_phenols' and the 'flavanoids' are the two features having the largest correlation among them.

## We have now completed all the parts of the question 1!!

## Question 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**

In order to determine the equal frequency binning and the smoothing by bin boundaries, we need to sort our data in increasing order first:

Our data will look like:

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

Now let us divide the data into 3 bins as it is given in the question:

The 3 bins are as follows:

Bin 1: 1, 2, 5, 6

Bin 2: 7, 8, 9, 11

Bin 3: 12, 13, 17, 20

Here are the 3 bins that have the equal frequencies in them or we can say that they have equal number of data points in them.

Now let us smooth the data by bin boundaries, and to do so we will need to replace the middle values of the bins to its near boundary.

For Bin 1:

We have 1 as the first boundary and 6 as the second boundary. 2 is the first middle value, so let us first replace it with the either of the boundaries.

|2 – 1| = 1

|2 – 6| = 4

As 1 < 4, we will replace the value of 2 with 1.

Let us calculate for 5 which is another middle value.

|5 – 1| = 4

|5 – 6| = 1

As 4 > 1, we will replace the value of 5 with 6.

So, The first bin will be : Bin 1: 1, 1, 6, 6

For Bin 2:

We have 7 as the first boundary and 11 as the second boundary. 8 is the first middle value, so let us first replace it with the either of the boundaries.

|8 – 7| = 1

|8 – 11| = 3

As 1 < 3, we will replace the value of 8 with 7.

Let us calculate for 9 which is another middle value.

|9 – 7| = 2

|9 – 11| = 2

Here, we got both the distances as equal, but as per the condition in Smoothing – By – Bin – Boundaries, we need to replace the vale of 9 by the first boundary, i.e. 7.

So, The second bin will be : Bin 2: 7, 7, 7, 11

For Bin 3:

We have 12 as the first boundary and 20 as the second boundary. 13 is the first middle value, so let us first replace it with the either of the boundaries.

|13 – 12| = 1

|13 – 20| = 7

As 1 < 7, we will replace the value of 13 with 12.

Let us calculate for 17 which is another middle value.

|17 – 12| = 5

|17 – 20| = 3

As 5 > 3, we will replace the value of 17 with 20.

So, The third bin will be : Bin 3: 12, 12, 20, 20

## We have now completed all the parts of the question 2!!

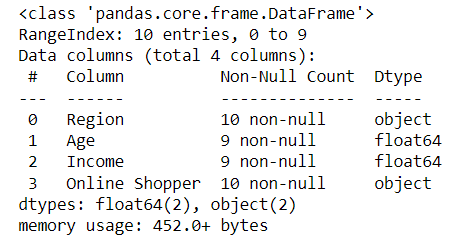
## Question 3:

**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.**

Now, here in the first part for question 3 we need to load the csv file. In order to do so, I am using Jupyter notebook so I first uploaded the csv file i.e. the 'country-income.csv' file into the environment and then I can fetch it now.

data\_country\_income = pd.read\_csv('country-income.csv')

data\_country\_income.info()



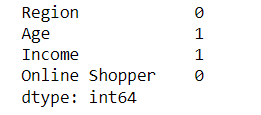
Here you can see that the country\_income.csv file has 4 columns and 2 of them are categorical and two of them are numerical features.

## We have completed the first part of the question 3!!

For the second part of our question 3, we need to first check whether there are any 'NaN' values in the dataset. We can do that by calculating the total sum of all the NaN values in each of the features of the dataset.

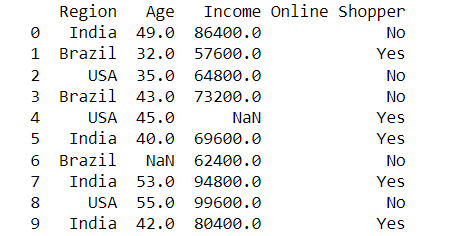
nan\_values = data\_country\_income.isna().sum()

print(nan\_values)



We are now able to see that there is one 'NaN' value in Age column and one 'NaN' value in the Income column. Now we need to replace these values by the mean, i.e. we need to calculate the mean of the features 'Age' and 'Income' and replace the value of respected 'NaN' value in age column by the mean of 'Age' and respected 'NaN' value in income column by the mean of 'Income'. But let us first check the rows where the 'NaN' values are present so that we know that it has been replaced.

print(data\_country\_income)



Here we can see that row 7 of our data has 'NaN' value for 'Age' feature and row 5 of our data has 'NaN' value for 'Income' feature.

# Initialize the SimpleImputer with strategy 'mean'

numeric\_columns = ['Age', 'Income']

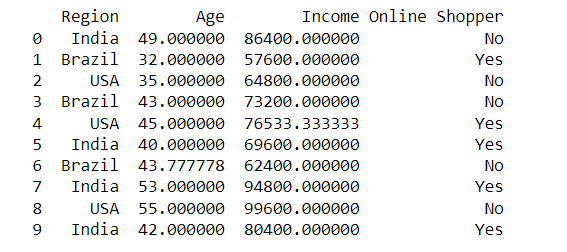
imputer = SimpleImputer(strategy='mean')

# Apply the imputer to fill NaN values with the mean of each feature

#data\_country\_income = pd.DataFrame(imputer.fit\_transform(data\_country\_income), columns=data\_country\_income.columns)

data\_country\_income[numeric\_columns] = imputer.fit\_transform(data\_country\_income[numeric\_columns])

print(data\_country\_income)



Now, you can clearly see that the 'NaN' values are replaced by the respective column means.

## We have completed the second part of the question 3!!

We have our third part of the question to replace the categorical feature to a numerical one. Here we have two categorical features, lets replace them.

label\_encoder = LabelEncoder()

# Iterate through the columns and check if they are of object data type (categorical)

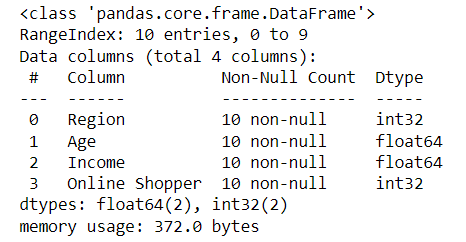
for column in data\_country\_income.columns:

if data\_country\_income[column].dtype == 'object':

data\_country\_income[column] = label\_encoder.fit\_transform(data\_country\_income[column])

We have used Label\_Encoder from the sklearn library to transform the categorical features into numerical format.

data\_country\_income.info()

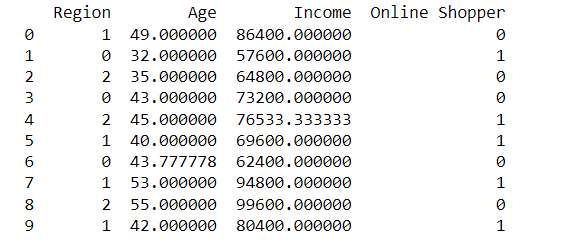


Now we are able to see that the two categorical features are converted to integer.

## We have completed the third part of the question 3!!

Now for our last part we need to display our dataset.

print(data\_country\_income)



## We have now completed all the parts of the question 3!!

## Question 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.**

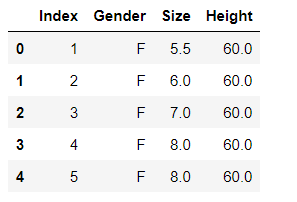
First part of the question 4 is simple, we just need to load the dataset "shoesize.csv" into our Jupyter Notebook as done in the previous question.

shoe\_data = pd.read\_csv('shoesize.csv')

## We have completed the first part of the question 4!!

Let us check the data before performing any further activities.

shoe\_data.head()



Now as the second part of the question 4, we need to plot the scatter plot having shoesize on x-axis and height on the y-axis. This is to be done for male subjects and female subjects separately. Let us first divide the data into male and female subjects.

male\_subjects = shoe\_data[shoe\_data['Gender'] == 'M']

female\_subjects = shoe\_data[shoe\_data['Gender'] == 'F']

Now that we have our data separated into male and female subjects we can now plot the scatterplot of them.

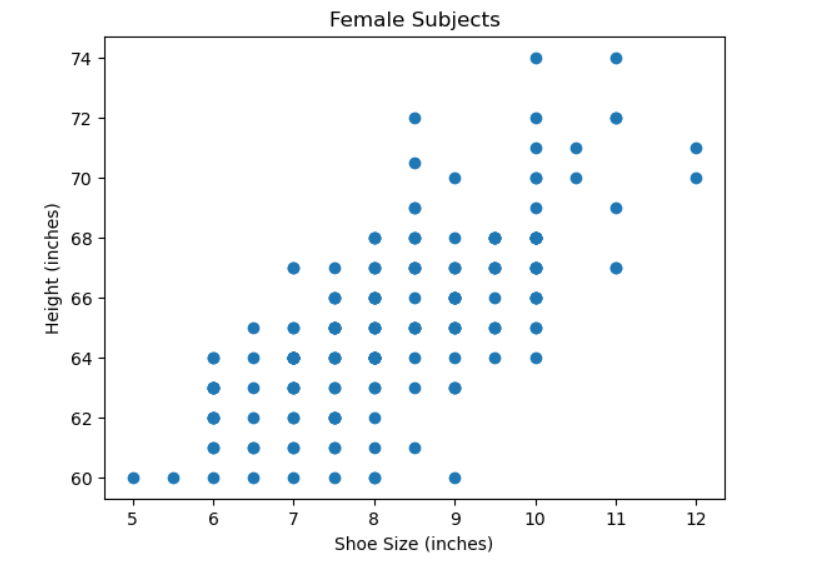
plt.scatter(female\_subjects['Size'], female\_subjects['Height'])

plt.title('Female Subjects')

plt.xlabel('Shoe Size (inches)')

plt.ylabel('Height (inches)')

plt.show()



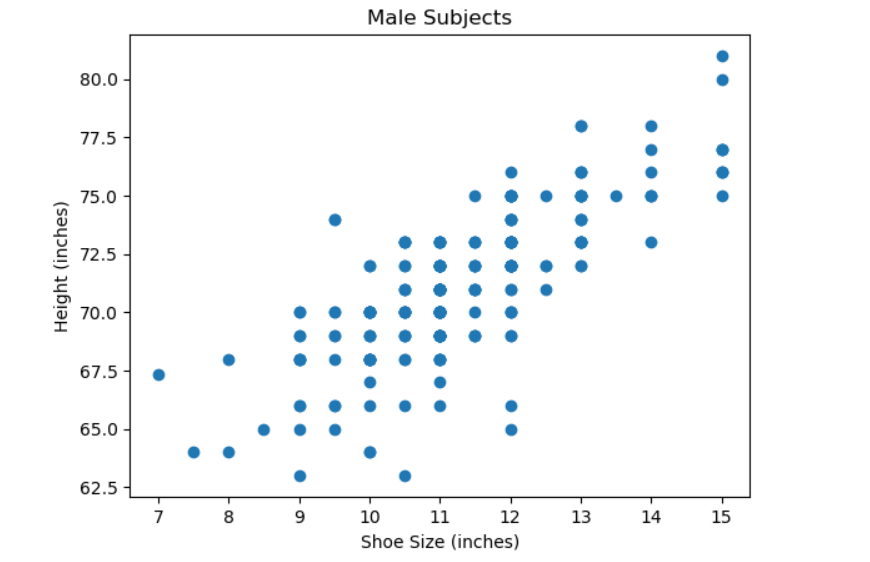
plt.scatter(male\_subjects['Size'], male\_subjects['Height'])

plt.title('Male Subjects')

plt.xlabel('Shoe Size (inches)')

plt.ylabel('Height (inches)')

plt.show()



## We have completed the second part of the question 4!!

Now for the part 3 of the question 4, we need to calculate the Pearson's correlation coefficient of shoe versus the height of male and female separately.

corr\_female = female\_subjects['Size'].corr(female\_subjects['Height'], method='pearson')

corr\_male = male\_subjects['Size'].corr(male\_subjects['Height'], method='pearson')

print("Pearson's correlation coefficient for female subjects:", corr\_female)

print("Pearson's correlation coefficient for male subjects:", corr\_male)



## We have completed the third part of the question 4!!

As the last part of the question 4 is concerned, we need to give the inference of the scatterplot and the Pearson's correlation coefficient.

1] The Scatter Plot:

We can visualize the relationship between the shoe size and the height of the male and female observations, which is what scatterplot is eventually used for. If in case we see that the data points are moving upwards from left to right then we can say that it is a positive correlation between the two features of a dataset, and if vice versa then we can say that it is a negative correlation between the two features. We can clearly see that there is a type of positive correlation between the shoe size and the height of the female and it is also a positive correlation for the same features in the case of male subjects.

2] The Pearson Correlation Coefficient:

In order to determine the strength of a relationship between the two features and also to determine the direction of their relationship, we can compute the Pearson's Correlation Coefficient. If in case we see that the result of this coefficient is greater than zero then it is a positive correlation and if the case is vise versa then it is a negative correlation. In our case, we can clearly see that the Person's Correlation Coefficient for both male and female subjects are greater than zero, so we can easily say that there is a positive correlation between shoe size and height features of male and female subjects.

## We have now completed all the parts of the question 4!!

## Question 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 one 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 one from the other? Include the code in your report.**

Here we need to compute the PCA with two components such that the plot will have color coding with respect to the "target" feature that we added to the dataset in Question 1. In order to do so, we have the variable "data" that we used to load the wine dataset, not lets convert this variable into a Data Frame.

wine\_df = pd.DataFrame(data=data.data, columns=data.feature\_names)

Now, in ouder to perform PCA we need our data to be in a standard format in such a way that the mean of the dataset we have should be close to zero and the standard deviation should be close to one. So, let us transform the dataset.

scaler = StandardScaler()

wine\_std = scaler.fit\_transform(wine\_df)

We need only two components in our PCA as per the question states, so let us specify that.

pca = PCA(n\_components=2)

wine\_pca = pca.fit\_transform(wine\_std)

In order to plot the scatterplot with these two principal components, we need to create a Data Frame that will contain these two components.

wine\_df\_pca = pd.DataFrame(data=wine\_pca, columns=['PC1', 'PC2'])

We need to plot our scatterplot that will have a color coding of the 'target' feature, so we need to combine the two components and the 'target' feature of our dataset.

plt.scatter(wine\_df\_pca['PC1'], wine\_df\_pca['PC2'], c=data.target)

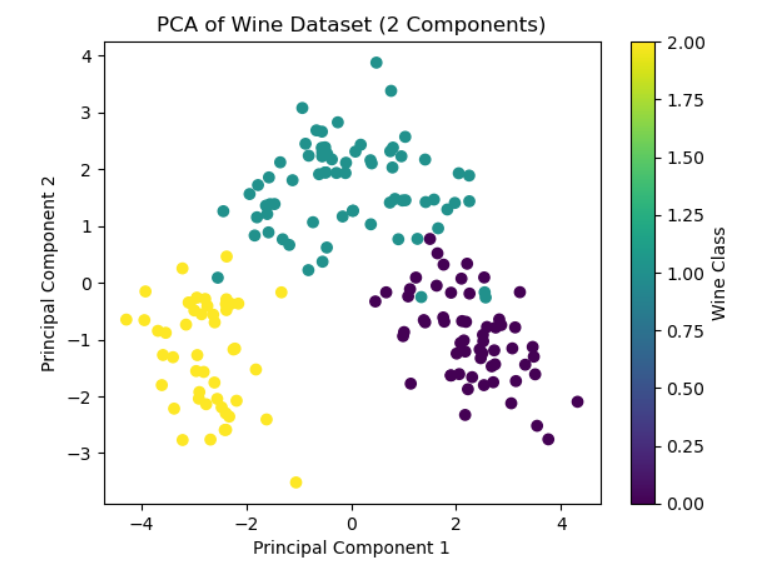
plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA of Wine Dataset (2 Components)')

plt.colorbar(label='Wine Class')

plt.show()



## We have completed the first part of the question 5!!

As the second part is concerned, we need to specify the insights that we can gather from the above plotted scatterplot of PCA.

1] The first thing to be observed is that the property of PCA, which is that it reduces the dimensionality of the data. This can be clearly seen in our plot.

2] We can observe the 'Outliers' which are known as the data points or the records which when plotted on a graph appears to be deviated from the group of other records or behaves totally different from the rest of the data points. Here, in our plot, the outliers are the points that are away from the main-colored cluster.

3] We can observe the 'Patterns' in the dataset. We can see that two classes that are colored in yellow and violet are two distinct classes, which do not tend to mix with each other, while there is one cluster which i colored in greenish-blue, which is getting mixed with the other two classes or the clusters.

4] As we observed the 'Patterns', we can say that it is difficult for the classification in the cluster which is greenish-blue colored, as its samples are mixed with the other clusters. So based on these components alone, we cannot determine the classes or clusters of few samples.

5] In order to determine the variance between the two clusters, the clusters need to be purely distinct from each other. In our case, it is not possible as one cluster is getting mixed with the rest of two clusters.

## We have completed the second part of the question 5!!

The third part of the question states that whether can we distinguish between the samples belonging to one class from samples belonging to the other classes.

In order to determine this, we need to understand the properties of PCA. When a PCA is plotted with the help of scatterplot, the data points on those scatterplot are plotted in such a way that their positions are determined by the first 2 components determined in the PCA. The color of the classes in the scatterplot is given by our 'target' feature of wine dataset. In the scatterplot that we have plotted, we can see 3 clusters plotted with 3 different colors according to the 'target' feature. There is a slight overlap of the datapoints in the greenish-blue cluster with that of the rest of two clusters i.e. yellow and violet clusters. This is because of the covariance of the features. The covariance of the features shows the Principal Component the direction in the feature space. As some points are lying far from the cluster, this is because of the difference in their variances. So, YES, we can distinguish between the samples belonging to one class from samples belonging to the other classes except a few samples. As the overlap is minimal, we can surely say that the PCA have performed effectively.

## We have now completed all the parts of the question 5!!

## Question 6:

**In Lab session 3 (Data Exploration and Data Visualisation), in subsection 1.9 you had created and visualised a heatmap for the distance matrix for the graduation\_rate.csv. You may have noticed that the distance matrix visualisation 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 visualisation.**

In this question, we need to look into the 1.9 section of Lab 3 in which we have plotted the 'HeatMap' for distance matrix for the "graduation\_rate.csv" dataset.

The Heat Map we created does not give us much information about the individual distances between the classes of students, but it is possible that we can gather some insights over this heat map regarding the average distance between the 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 as asked in the question.

Let us gather the insights from the heat map:

1] If we look into the colors of the heat map, then we can understand that the dark color will show us the greater distances and the light color will show us the smaller distances.

2] We can also see that the students are grouped in such a way that it reflects their parents education and the students those who share parents with same level of degree in education are grouped together.

3] We can also see the thin diagonal line on our heat map which has the light color which shows us the distances between the students who share the parents having same level degree in their education with itself. This line is light because the distance from one observation with itself is always zero.

4] The rest of the part from the diagonal has a representation of students that do not share parents having same degree in education.

5] If the lines that you come across is dark then the distance between those two students compared will be higher, while the distance will be smaller if the color of the line is light.

6] The two groups that are "students whose parents have some high school education" and "students whose parents have a master's degree" and taken their average will be higher which shows that these two groups do not share any kind of similarity between them.

## We have now completed all the parts of the question 6!!

## Question 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).**

**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.**

We need to load the data, create a table and populate it with contents of the CSV file.

engine = create\_engine('sqlite:///data.sqlite')

create\_table\_from\_csv(engine,

"country-income.csv",

table\_name="income\_cube",

fields=[

("region", "string"),

("age", "integer"),

("income", "integer"),

("online\_shopper", "string")],

create\_id=True

)

Let us now create the workspace object and register a default store for SQLite in the cubes library.

workspace = Workspace()

workspace.register\_default\_store("sql", url="sqlite:///data.sqlite")

The structure of data cubes is in JSON file “country\_json1.json”. The code of JSON file is below:

{

"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": "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"

}

]

}

]

}

Let us now import the JSON file.

workspace.import\_model("country\_json1.json")

Now let us create the data cube.

cube = workspace.cube("income\_cube")

Now, we will create an object called browser which will perform our aggregate functions.

browser = workspace.browser(cube)

Let us try to calculate the sum of the cube for instance.

result = browser.aggregate()

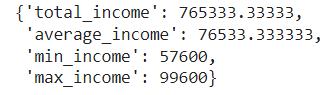
result.summary["total\_income"]



Let us produce aggregate results for the whole data cube.

result = browser.aggregate()

result.summary

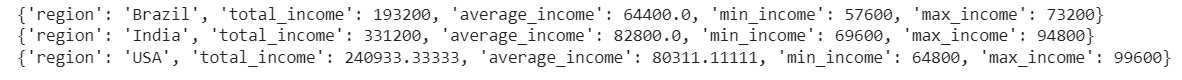


Let us produce aggregate results for the results per region.

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

for record in result:

print(record)



Let us produce aggregate results for the results per online shopping activity.

result = browser.aggregate(drilldown=["online\_shopper"])

for record in result:

print(record)



Let us produce aggregate results for the results for all people aged between 40 and 50.

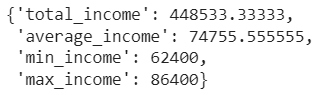
import cubes as cubes

cuts = [cubes.RangeCut("age", [40], [50])]

cell = cubes.Cell(cube, cuts)

result = browser.aggregate(cell)

result.summary



## We have now completed all the parts of the question 7!!

## Question 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 neighbour 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?**

Here, we have:

x1 = (1,2)

x2 = (-1,0)

y1 = 1

y2 = 0

To Find:

Classify the observations below based on 1- Nearest Neighbour Classifier:

1) x3 = (3,2)

2) x4 = (0,1)

Let us calculate the Euclidean Distance for x3 from x1:

sqrt((3 - 1)^2 + (2 - 2)^2) = sqrt((2^2) + (0^2))

= sqrt(4) = 2

Let us calculate the Euclidean Distance for x3 from x2:

sqrt((3 - (-1))^2 + (2 - 0)^2) = sqrt((4^2) + (2^2))

= sqrt((16 + 4)) = sqrt(20) = approx. 4.47

As the Euclidean distance of x3 from x1 is shorter that the Euclidean distance of x3 from x2, we can classify the point x3 in class - 1.

Let us calculate the Euclidean Distance for x4 from x1:

sqrt((0 - 1)^2 + (1 - 2)^2) = sqrt(((-1)^2) + ((-1)^2))

= sqrt(1 + 1) = approx. 1.41

Let us calculate the Euclidean Distance for x4 from x2:

sqrt((0 - (-1))^2 + (1 - 0)^2)) = sqrt(((-1)^2) + (1^2))

= sqrt(1 + 1) = approx. 1.41

Now, as x3 is already the part of class 1, let us calculate the Euclidean Distance for x4 from x3:

sqrt((0 - 3)^2 + (1 - 2)^2) = sqrt(((-3)^2) + ((-1)^2))

= sqrt(9 + 1) = approx. 3.16

As we can see here, we have two points (i.e. x1 and x3) from class - 1 and one point(i.e. x2) from class - 2. Now, since we are not using the code to classify the points into the clusters, we can consider x3 as the nearest neighbour as we cannot specify the nearest neighbour by just looking at the points. Now, if we consider the distance of x4 from x1 of class – 1 and distance of x4 from x2 of class – 2, we can say that the point x4 can be classified in either of the classes, as the distances are equal. If we consider the distance of x4 from x3 of class – 1 and distance of x4 from x2 of class – 2, we can say that the point x4 can be classified in class – 2. But, on the other hand, our code will never try to check the distance of x4 from x3, when considering 1-NN. So, the result of the code for classifying x4, will be either class – 1 or class – 2.

## We have now completed all the parts of the question 8!!

End Of Assignment!!