# Data Mining Assignment 1

## Alex Chau

### November 2023

## 1 Question 1

a) The categorical feature is "Target", this can be shown via the following:

```python
import pandas as pd
from sklearn.datasets import load_wine

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

```
array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

Getting the frequency of each categorical feature is simple.

```python
df.groupby('target').size()/len(df)
```

```
target
0    0.331461
1    0.398876
2    0.269663
dtype: float64
```

b) For univariate summaries I chose to calculate the standard deviation and the mean of each numerical feature. For multivariate summaries I created a scatter matrix and a covariance matrix of the numerical features.

```
1  display(df.drop('target', axis = 1).std())
```

```
alcohol                           0.811827
malic_acid                        1.117146
ash                               0.274344
alcalinity_of_ash                 3.339564
magnesium                        14.282484
total_phenols                     0.625851
flavanoids                        0.998859
nonflavanoid_phenols              0.124453
proanthocyanins                   0.572359
color_intensity                   2.318286
hue                               0.228572
od280/od315_of_diluted_wines      0.709990
proline                         314.907474
dtype: float64
```

```
1  display(df.drop('target', axis = 1).mean())
```

```
alcohol                          13.000618
malic_acid                        2.336348
ash                               2.366517
alcalinity_of_ash                19.494944
magnesium                        99.741573
total_phenols                     2.295112
flavanoids                        2.029270
nonflavanoid_phenols              0.361854
proanthocyanins                   1.590899
color_intensity                   5.058090
hue                               0.957449
od280/od315_of_diluted_wines      2.611685
proline                         746.893258
dtype: float64
```

```
1  pd.plotting.scatter_matrix(df.drop('target', axis = 1))
```

|  | alcohol | malic_acid | ash | alcalinity_of_ash | magnesium | total_phenols | flavanoids | nonflavanoid_phenols | proanthocyanins | color_intensity | hue | od280/od315_of_diluted_wines | proline |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alcohol | 0.659062 | 0.085611 | 0.047115 | -0.841093 | 3.139878 | 0.146887 | 0.192033 | -0.015754 | 0.063518 | 1.028283 | -0.013313 | 0.041698 | 164.567185 |
| malic_acid | 0.085611 | 1.248015 | 0.050277 | 1.076332 | -0.870780 | -0.234338 | -0.458630 | 0.040733 | -0.141147 | 0.644838 | -0.143326 | -0.292447 | -67.548867 |
| ash | 0.047115 | 0.050277 | 0.075265 | 0.406208 | 1.122937 | 0.022146 | 0.031535 | 0.006358 | 0.001516 | 0.164654 | -0.004682 | 0.000762 | 19.319739 |
| alcalinity_of_ash | -0.841093 | 1.076332 | 0.406208 | 11.152686 | -3.974760 | -0.671149 | -1.172083 | 0.150422 | -0.377176 | 0.145024 | -0.209118 | -0.656234 | -463.355345 |
| magnesium | 3.139878 | -0.870780 | 1.122937 | -3.974760 | 203.989335 | 1.916470 | 2.793087 | -0.455563 | 1.932832 | 6.620521 | 0.180851 | 0.669308 | 1769.158700 |
| total_phenols | 0.146887 | -0.234338 | 0.022146 | -0.671149 | 1.916470 | 0.391690 | 0.540470 | -0.035045 | 0.219373 | -0.079998 | 0.062039 | 0.311021 | 98.171057 |
| flavanoids | 0.192033 | -0.458630 | 0.031535 | -1.172083 | 2.793087 | 0.540470 | 0.997719 | -0.066867 | 0.373148 | -0.399169 | 0.124082 | 0.558262 | 155.447492 |
| nonflavanoid_phenols | -0.015754 | 0.040733 | 0.006358 | 0.150422 | -0.455563 | -0.035045 | -0.066867 | 0.015489 | -0.026060 | 0.040121 | -0.007471 | -0.044469 | -12.203586 |
| proanthocyanins | 0.063518 | -0.141147 | 0.001516 | -0.377176 | 1.932832 | 0.219373 | 0.373148 | -0.026060 | 0.327595 | -0.033504 | 0.038665 | 0.210933 | 59.554334 |
| color_intensity | 1.028283 | 0.644838 | 0.164654 | 0.145024 | 6.620521 | -0.079998 | -0.399169 | 0.040121 | -0.033504 | 5.374449 | -0.276506 | -0.705813 | 230.767480 |
| hue | -0.013313 | -0.143326 | -0.004682 | -0.209118 | 0.180851 | 0.062039 | 0.124082 | -0.007471 | 0.038665 | -0.276506 | 0.052245 | 0.091766 | 17.000223 |
| od280/od315_of_diluted_wines | 0.041698 | -0.292447 | 0.000762 | -0.656234 | 0.669308 | 0.311021 | 0.558262 | -0.044469 | 0.210933 | -0.705813 | 0.091766 | 0.504086 | 69.927526 |
| proline | 164.567185 | -67.548867 | 19.319739 | -463.355345 | 1769.158700 | 98.171057 | 155.447492 | -12.203586 | 59.554334 | 230.767480 | 17.000223 | 69.927526 | 99166.717355 |

```
1  df.drop('target', axis = 1).cov()
```
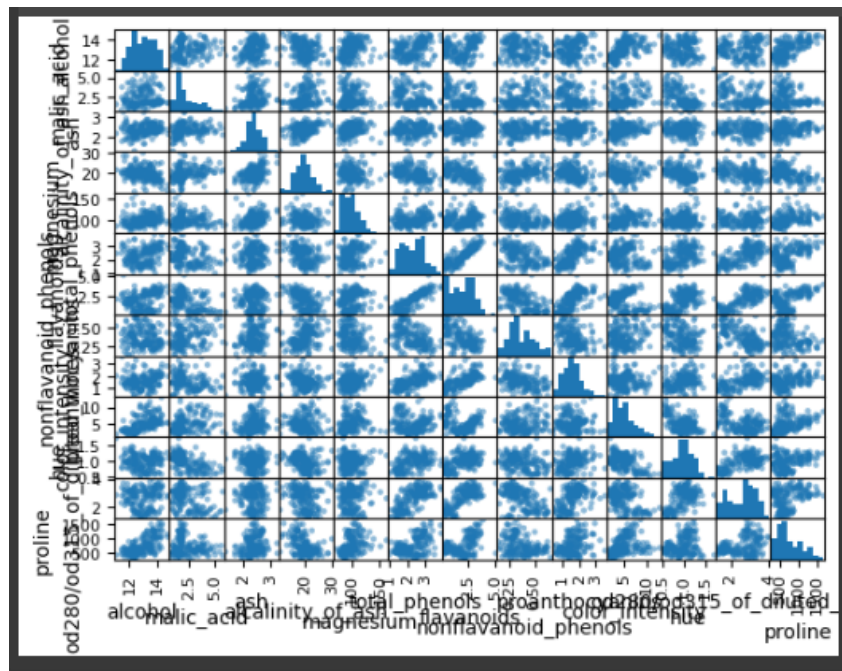


c)

```
1  df.groupby('target').median()
```

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

d) First I find, the pair with the highest correlation, flavanoids and total_phenols

```
1  matrix = df.drop('target', axis = 1).corr()
2  max_corr = matrix.mask(matrix==1).stack().sort_values(ascending=False).idxmax()
```

I then use this max_corr variable to create a scatter plot
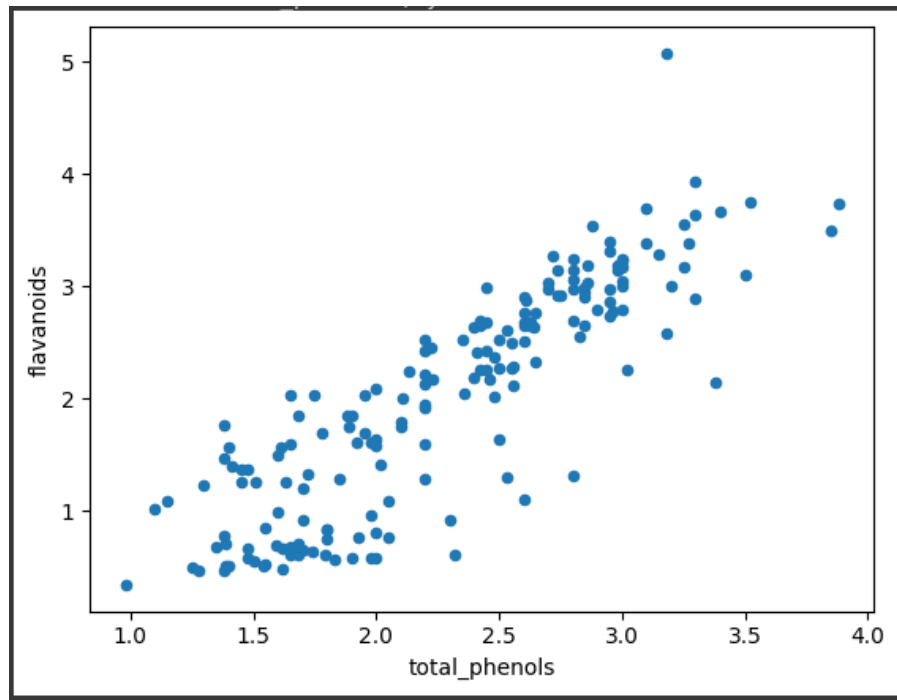
# 2   Question 2

First I read the csv file into a dataframe, I then created a SimpleImputer which replaces
NaN values with the mean. I then created a separate dataframe "numerical_data" which
selected only the numerical values of the data. I then applied the imputer to the numerical
data which cleaned the NaN values.

Next was to handle the categorical data, like before I split the original dataframe into its
categorical features and then created a LabelEncoder and applied it to the categorical
data. This normalises the labels into numerical values.

With both numerical and categorical features cleaned I concatenated them back together
into a single dataframe "cleaned_data".

```python
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder

ci = pd.read_csv("country-income.csv")
imp = SimpleImputer(missing_values=np.nan, strategy='mean')
numerical_data = ci.select_dtypes(include = ["number"])
numerical_data = pd.DataFrame(imp.fit_transform(numerical_data),
                        columns = numerical_data.columns)
categorical_data = ci.select_dtypes(include = ["object"])
le = LabelEncoder()
categorical_data = categorical_data.apply(le.fit_transform)
cleaned_data = pd.concat([numerical_data, categorical_data], axis=1)
cleaned_data
```

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

# 3  Question 3

First, load the shoesize csv into a dataframe, I drop the index column since it isn't of any use          in          this          question.

I then separated the dataframe into two dataframes based on the Gender

```python
import pandas as pd
ss = pd.read_csv("shoesize.csv")
ss.drop(labels='Index',inplace=True, axis='columns')
female_data = ss[ss['Gender'] == 'F']
male_data = ss[ss['Gender'] == 'M']
```

Then plot the scatter plot of the female and male data using DataFrame.plot()

```python
male_data.plot(x = 'Size', y = 'Height', kind = 'scatter')
female_data.plot(x = 'Size', y = 'Height', kind = 'scatter')
```
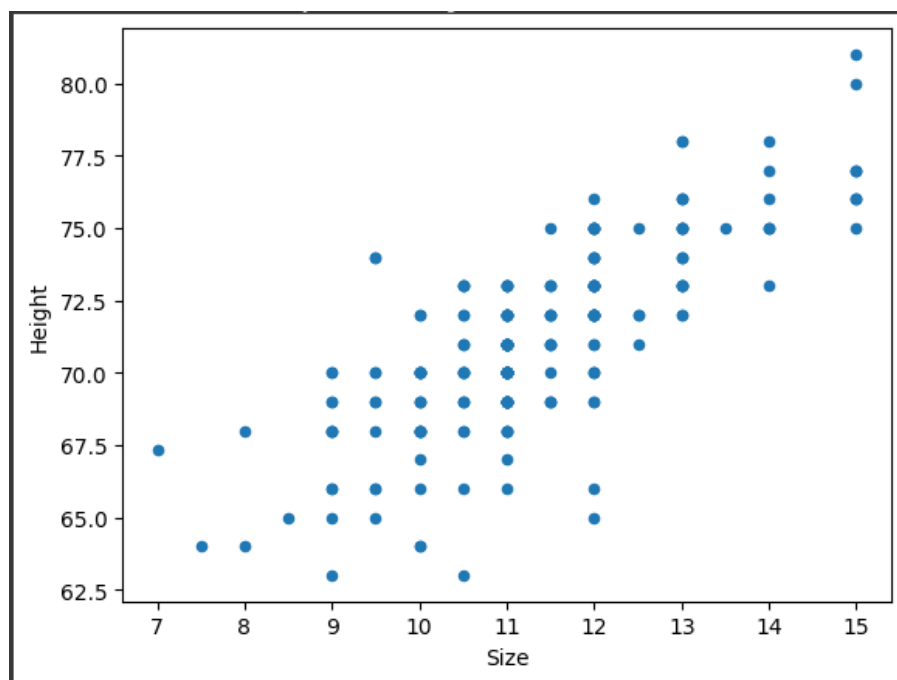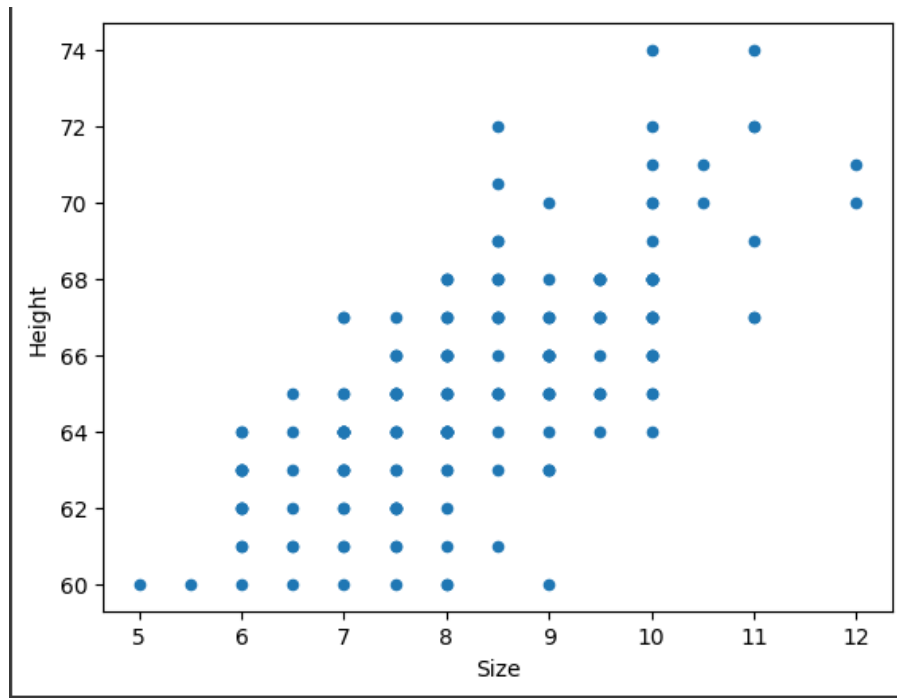
By default, DataFrame.corr() uses the pearson method, the data suggests that both male and female height to shoe size, have a strong linear correlation. In the case of males the male data having higher correlation than the female data.
male pearson correlation coefficient: 0.767709 female pearson correlation coefficient: 0.707812

```
1  male_data.corr()
2  female_data.corr()
```



(a) male pearson correlation



(b) female pearson correlation

# 4 Question 4

Looking at the below scatterplot of the principal components, you can see that target 0 (red) is somewhat distinguishable from target 2 and 1. There is some overlap. Whereas targets 1 and 2 have significant overlap and are not very distinguishable from each other when PCA is performed. This shows that when Targets 2 and 1 rows are projected into a lower dimension representation via PCA they are hard to distinguish due to having similar features.

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

wine = load_wine()
wdf = pd.DataFrame(wine.data, columns=wine.feature_names)

numComponents = 2
pca = PCA(n_components=numComponents)
pca.fit(wdf)
principal_components = pca.fit_transform(wdf)
principal_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])
principal_df['target'] = wine.target

colors = ['r', 'g', 'b']
plt.figure(figsize=(8, 6))
for target, color in zip(wine.target_names, colors):
    targetIndex = wine.target_names.tolist().index(target)
    subset = principal_df[principal_df['target'] == targetIndex]
    plt.scatter(subset['PC1'], subset['PC2'], c=color, label=target, alpha=0.7)

plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.grid(True)
```
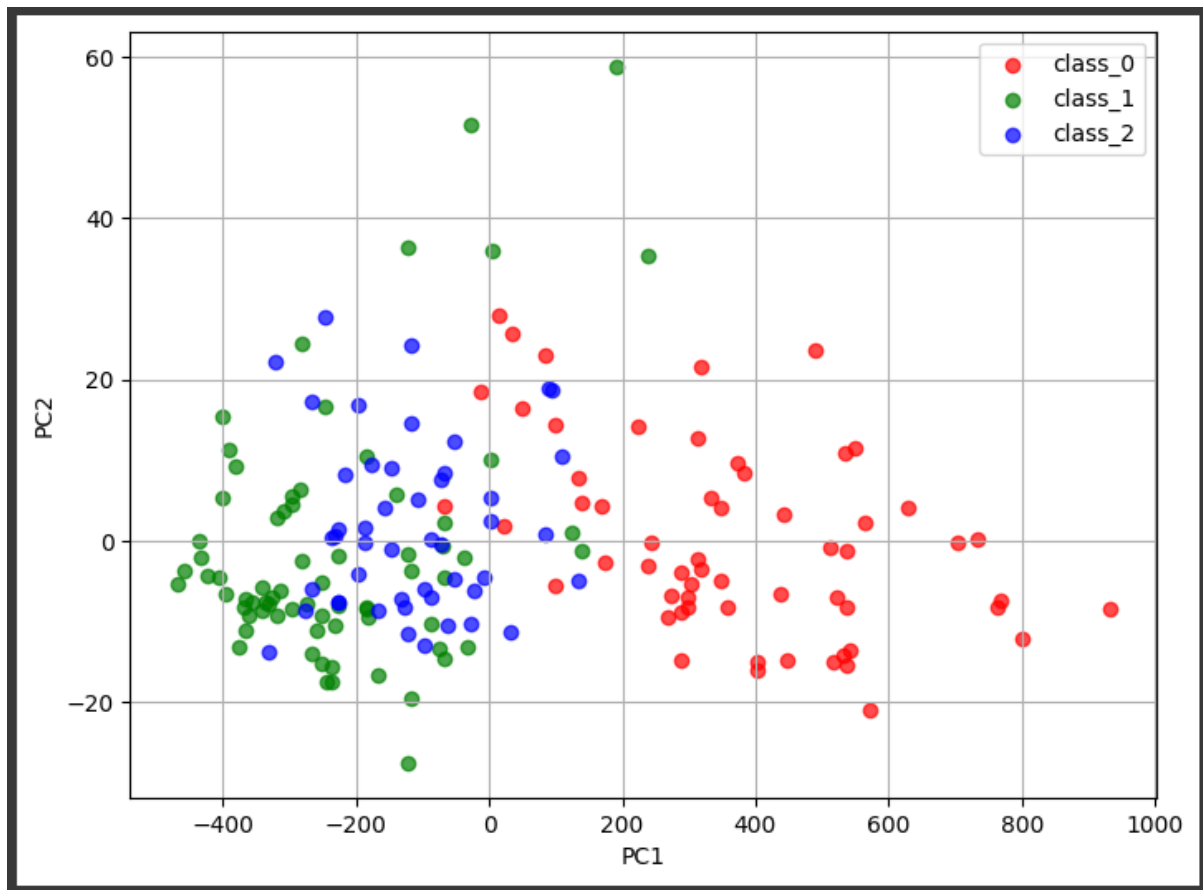
To address the overlap I will normalise the dataframe, using min max normalisation. The resulting scatterplot shows that the target classes are now very distinctive from each other, with very clear separation and extremely minimal overlap.

```
1  normalized_wdf=(wdf-wdf.min())/(wdf.max()-wdf.min())
2
3  numComponents = 2
4  pca = PCA(n_components=numComponents)
5  pca.fit(normalized_wdf)
6  principal_components = pca.fit_transform(normalized_wdf)
7  principal_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])
8  principal_df['target'] = wine.target
9
10 colors = ['r', 'g', 'b']
11 plt.figure(figsize=(8, 6))
12 for target, color in zip(wine.target_names, colors):
13     targetIndex = wine.target_names.tolist().index(target)
14     subset = principal_df[principal_df['target'] == targetIndex]
15     plt.scatter(subset['PC1'], subset['PC2'], c=color, label=target, alpha=0.7)
16
17 plt.xlabel('PC1')
18 plt.ylabel('PC2')
19 plt.legend()
20 plt.grid(True)
```
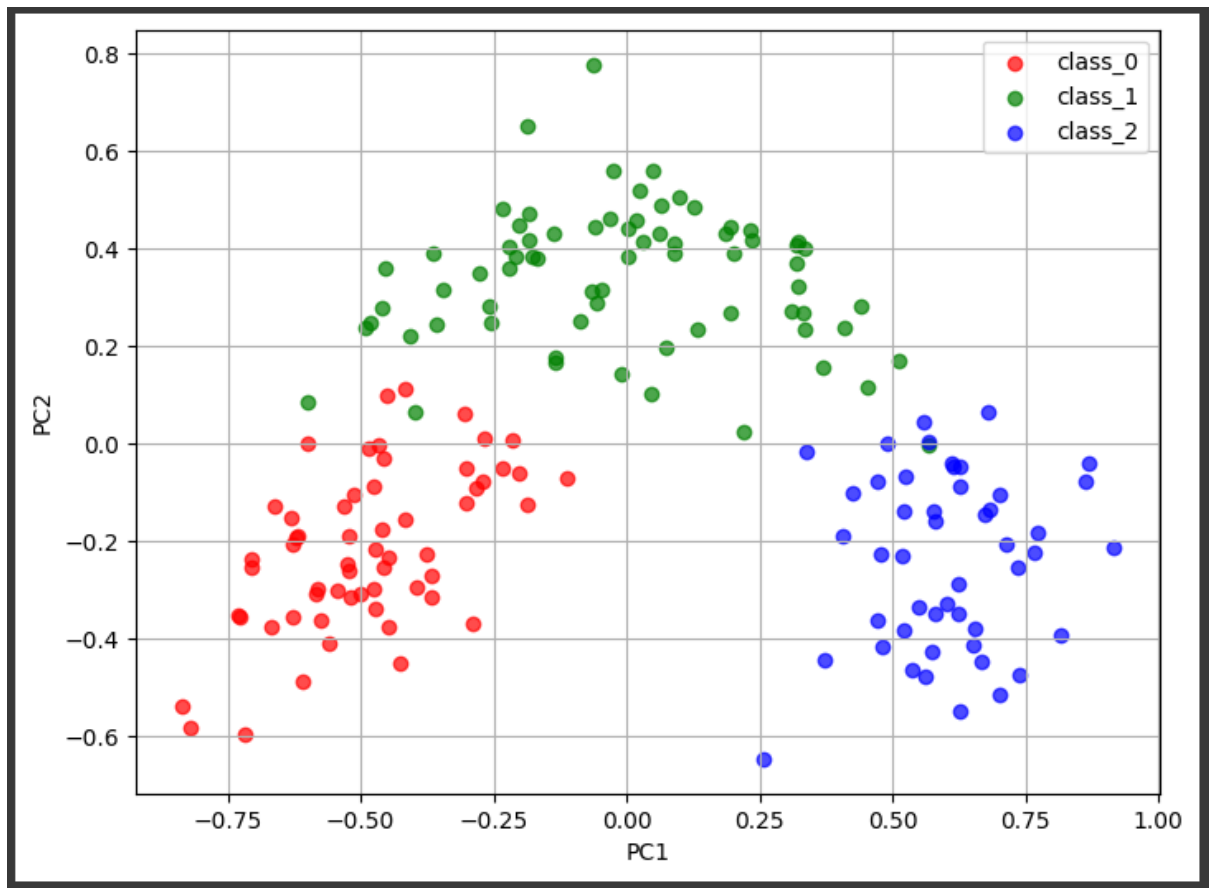
# 5   Question 5

$d(x, y)$ is the Euclidean distance between two points x and y. For $x_3 = (3, 2)$:

$$d(x_1, x_3) = \sqrt{(1 - 3)^2 + (2 - 2)^2} = 2$$

$$d(x_2, x_3) = \sqrt{(-1 - 3)^2 + (0 - 2)^2} = 2\sqrt{5}$$

Since $d(x_1, x_3) < d(x_2, x_3)$, $x_3$ will be classified with the same class as $x_1$, which is $y_1 = 1$.

For $x_4 = (0, 1)$:

$$d(x_1, x_4) = \sqrt{(1 - 0)^2 + (2 - 1)^2} = \sqrt{2}$$

$$d(x_2, x_4) = \sqrt{(-1 - 0)^2 + (0 - 1)^2} = \sqrt{2}$$

The two distances are the same, you can choose either, so i'll pick the arbritraty class $y_2 = 0$