ECS766P: Data Mining

Assignment 1

In [1]:
```
from sklearn.datasets import load_wine
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
```

1a) 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. [8 marks]

The wine dataset is loaded to identify the "target" feature as categorical because it assigns discrete attributes. Using Pandas, to print the frequencies in the "target" feature, revealing how many times 0,1,2 appears in the dataset.

In [2]:
```
#Load dataset and print frequency of the categorical variable
data= load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = pd.Series(data.target)
df['target'].value_counts()
```

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

b) Compute two different univariate and two different multivariate summaries for all numerical features. Include the code in your report. [8 marks]

In [3]:
```
df.describe()
```

Out[3]:

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

In [4]:
```
#variance
df.var()
```

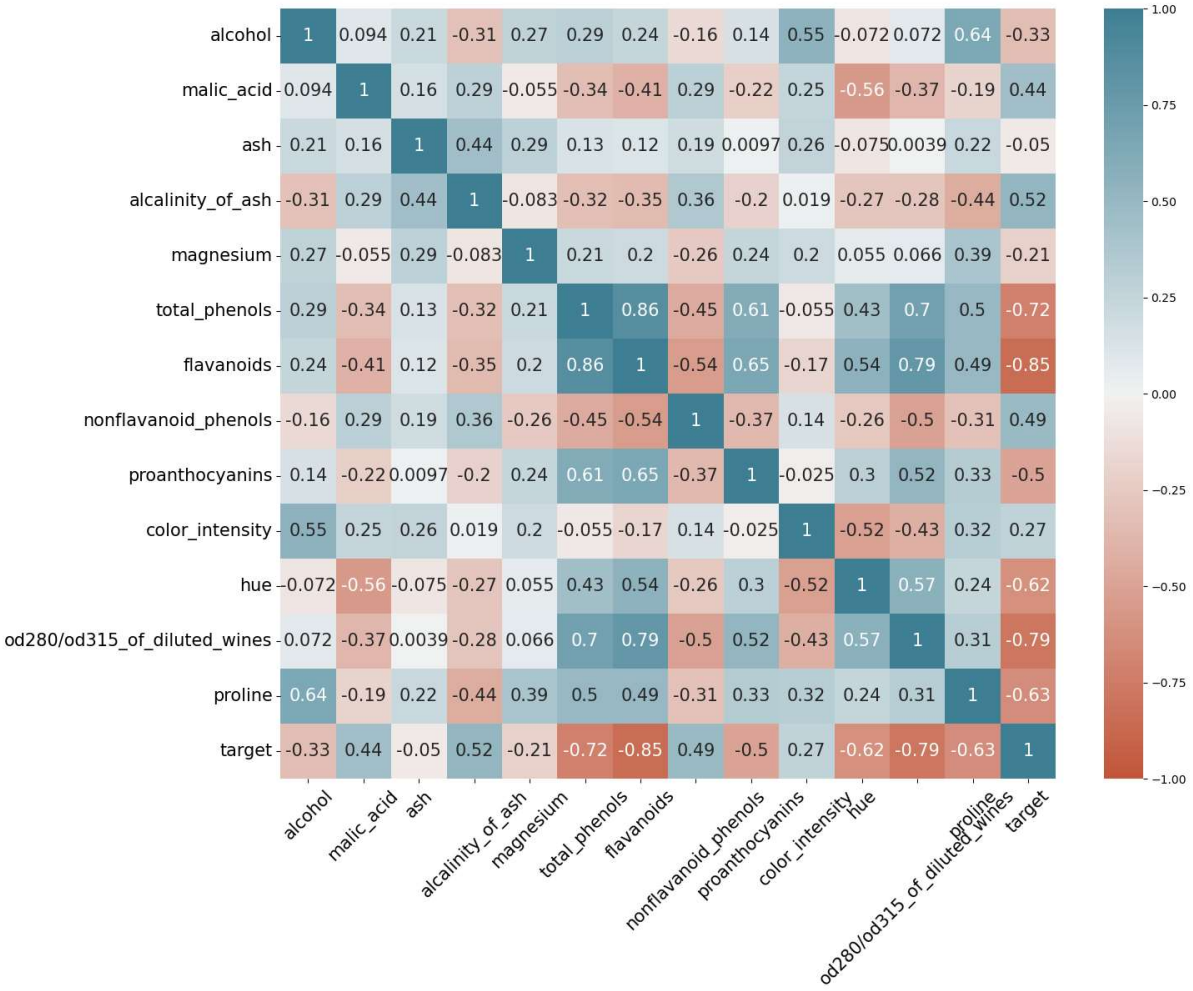
Out[4]:
```
alcohol                          0.659062
malic_acid                       1.248015
ash                              0.075265
alcalinity_of_ash               11.152686
magnesium                      203.989335
total_phenols                    0.391690
flavanoids                       0.997719
nonflavanoid_phenols             0.015489
proanthocyanins                  0.327595
color_intensity                  5.374449
hue                              0.052245
od280/od315_of_diluted_wines     0.504086
proline                      99166.717355
target                           0.600679
dtype: float64
```

In [5]:
```
# covariance matrix
df.cov()
```

Out[5]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	to
alcohol	0.659062	0.085611	0.047115	-0.841093	3.139878	
malic_acid	0.085611	1.248015	0.050277	1.076332	-0.870780	
ash	0.047115	0.050277	0.075265	0.406208	1.122937	
alcalinity_of_ash	-0.841093	1.076332	0.406208	11.152686	-3.974760	
magnesium	3.139878	-0.870780	1.122937	-3.974760	203.989335	
total_phenols	0.146887	-0.234338	0.022146	-0.671149	1.916470	
flavanoids	0.192033	-0.458630	0.031535	-1.172083	2.793087	
nonflavanoid_phenols	-0.015754	0.040733	0.006358	0.150422	-0.455563	
proanthocyanins	0.063518	-0.141147	0.001516	-0.377176	1.932832	
color_intensity	1.028283	0.644838	0.164654	0.145024	6.620521	
hue	-0.013313	-0.143326	-0.004682	-0.209118	0.180851	
od280/od315_of_diluted_wines	0.041698	-0.292447	0.000762	-0.656234	0.669308	
proline	164.567185	-67.548867	19.319739	-463.355345	1769.158700	
target	-0.206515	0.379039	-0.010555	1.340364	-2.315495	

In [6]:
```
# fig,ax = plt.subplots(1,1, figsize=(15,12))
# sns.heatmap(df.corr(), vmin=-1, vmax=1, cmap=sns.diverging_palette(20, 220, as_cm
#             annot=True, ax=ax, annot_kws={"size": 15})
# _y = plt.yticks(rotation=0, fontsize=15)
# _x = plt.xticks(rotation=45, fontsize=15)
# plt.show()
#sns heatmap
fig, ax = plt.subplots(1, 1, figsize=(15, 12))
sns.heatmap(df.corr(), vmin=-1, vmax=1, cmap=sns.diverging_palette(20, 220, as_cmap
           annot=True, ax=ax, annot_kws={"size": 15})
_y = plt.yticks(rotation=0, fontsize=15)
_x = plt.xticks(rotation=45, fontsize=15)
plt.show()
```



c) Group observations by the categorical feature & compute the corresponding median for each remaining numerical feature. Include the code in your report. [8 marks]

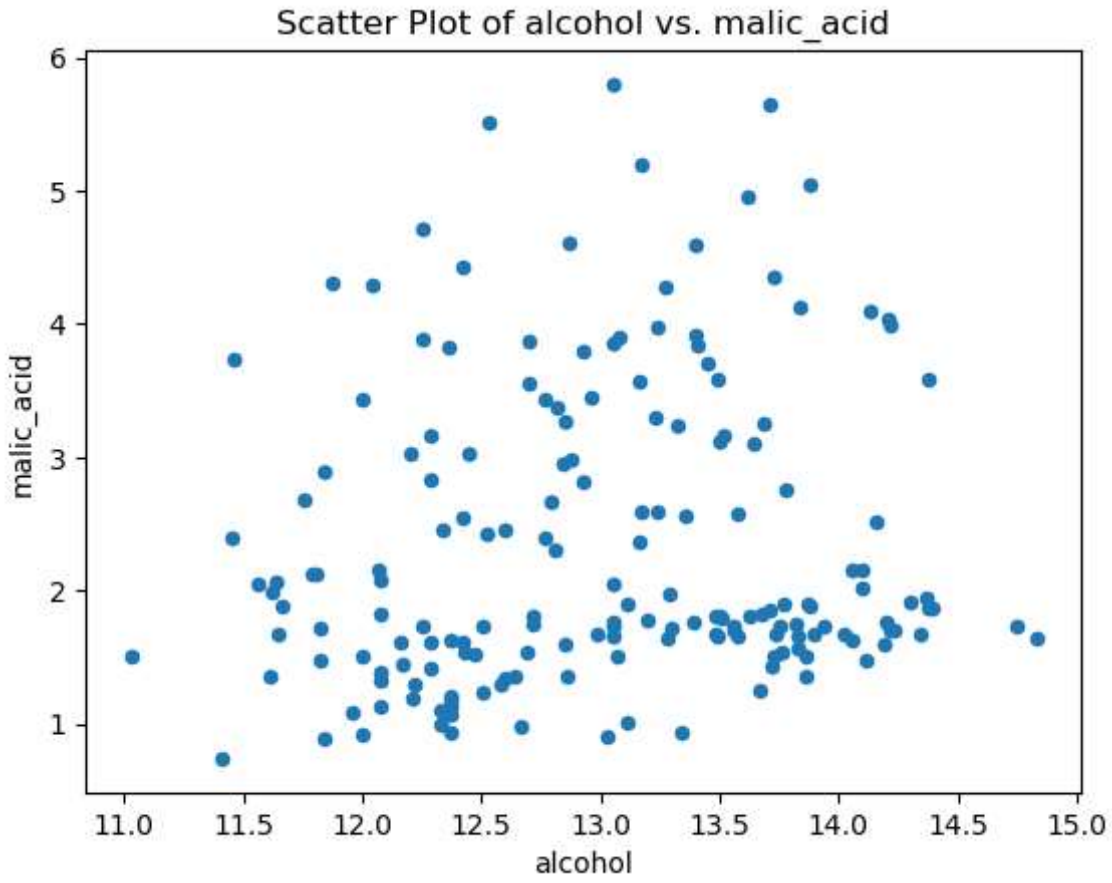
In [7]:
```
grouped = df.groupby('target')
#groupby median
medians = grouped.median()

print(medians)
```

```
        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.[4 marks]

In [8]:
```
correlation = df.corr()

#top two correlations
top = correlation.unstack().sort_values(ascending=False)
top2 = top.index[:2]
#scater plot for the top 2
df.plot(kind='scatter',  x='alcohol', y='malic_acid',title='Scatter Plot of alcohol
```

Out[8]:
```
<Axes: title={'center': 'Scatter Plot of alcohol vs. malic_acid'}, xlabel='alcoho
l', ylabel='malic_acid'>
```



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 [8 marks]

Bin 1: [1, 1, 1, 6] Bin 2: [7, 7, 7, 11] Bin 3: [12, 12, 12, 20]

using equal frequency 4 numbers per bin and using smoothing bin boundaries to order sales data to reduce the impact of outliers. Creating a gradual transitions between bins, this improves the stability of the data distribution, making it less sensitive to extreme values.

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. [8 marks]

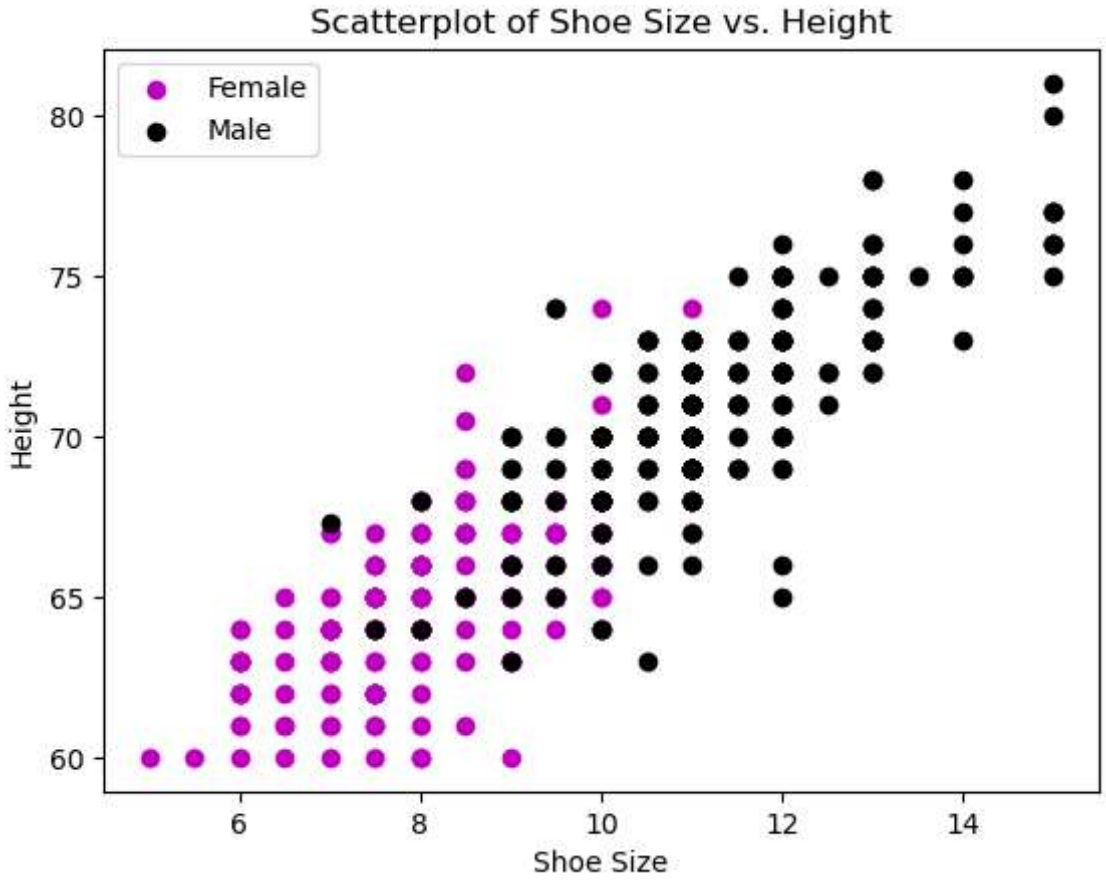
In [9]:
```
ci = pd.read_csv('country-income.csv')
```

```
#replace NaN with means
imputer = SimpleImputer(strategy='mean')
ci[ci.select_dtypes(exclude=['object']).columns] = imputer.fit_transform(ci.select_

#change categorical variables to numerical
label_encoder = LabelEncoder()

#change the categorical columns
categorical_columns = ci.select_dtypes(include=['object']).columns
for col in categorical_columns:
    ci[col] = label_encoder.fit_transform(ci[col])

print(ci)
```

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

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. [8 marks]

In [10]:
```
sz = pd.read_csv('shoesize.csv')

#filter data for female and male
fem= sz[sz['Gender'] == 'F']
mal = sz[sz['Gender'] == 'M']

#scatterplots for female and male
plt.scatter(fem['Size'], fem['Height'], label='Female', color='m')
plt.scatter(mal['Size'], mal['Height'], label='Male', color='k')

#Pearson's correlation coefficients
female, _ = pearsonr(fem['Size'], fem['Height'])
male, _ = pearsonr(mal['Size'], mal['Height'])

#plotlables
plt.xlabel('Shoe Size')
plt.ylabel('Height')
plt.title('Scatterplot of Shoe Size vs. Height')
plt.legend()

plt.show()

print('The Pearsons correlation coefficient of shoe size versus height for females
print('The Pearsons correlation coefficient of shoe size versus height for males is
```

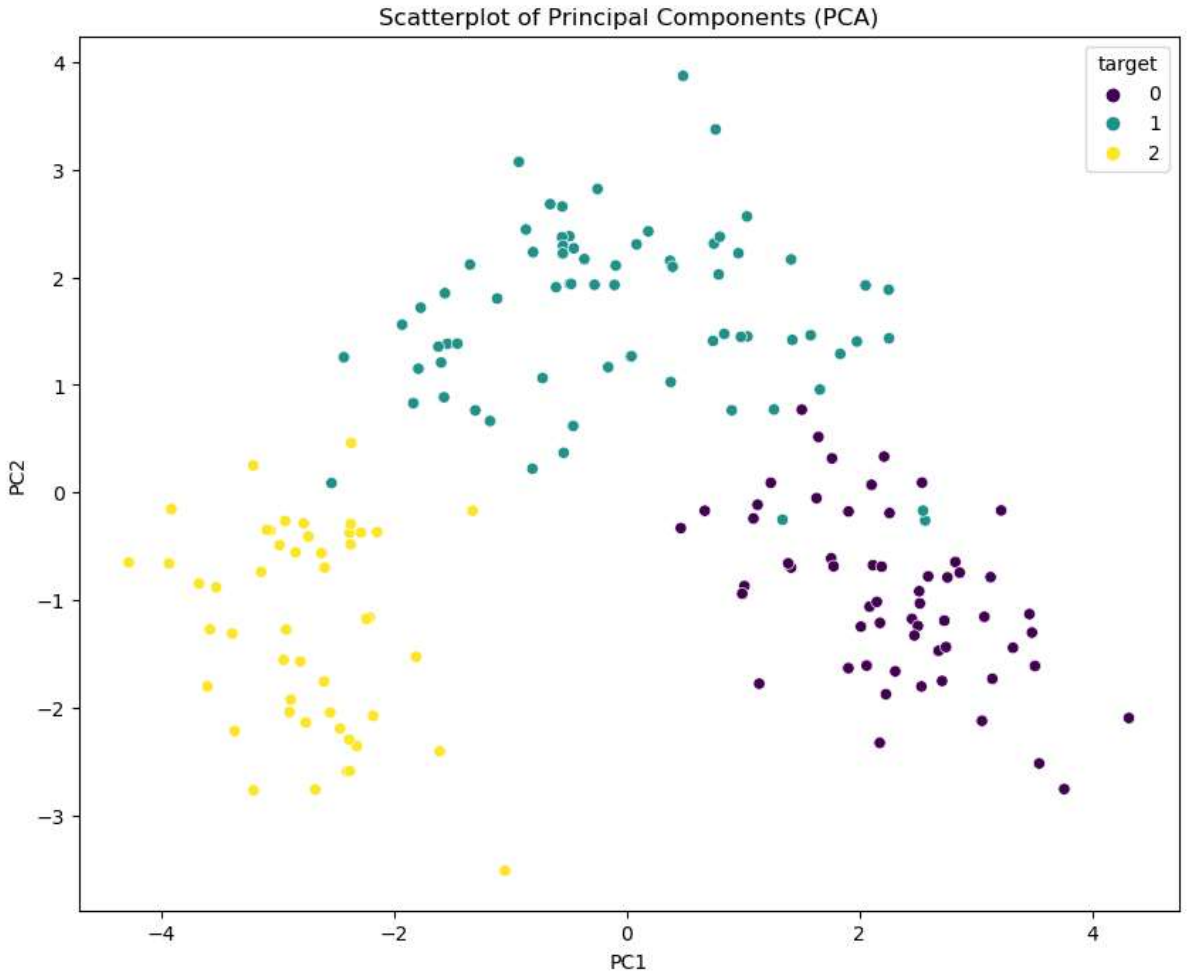



```
The Pearsons correlation coefficient of shoe size versus height for females is 0.7
07811941714397
The Pearsons correlation coefficient of shoe size versus height for males is 0.767
7093547300977
```

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. [20 marks]

In [11]:
```
#separate the target column for use
data= load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = pd.Series(data.target)
target = df['target']
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df.drop('target', axis=1))
#2 components pca
pca = PCA(n_components=2)
principal_components = pca.fit_transform(scaled_data)

pc_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])
```

```
pc_df['target'] = target

#scatterplot color-coded
plt.figure(figsize=(10, 8))
sns.scatterplot(x='PC1', y='PC2', hue='target', data=pc_df, palette='viridis', lege
plt.title('Scatterplot of Principal Components (PCA)')
plt.show()
```



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. [4 marks]

The heatmap visually represents the distance matrix, where lighter colours show more distances between rows. By sorting the dataset based on parental education, it shows that the average distance between students with some high school-educated parents and those with master's degree-educated parents is larger, as shown by the lighter colors on the heatmap. This suggests a greater difference in graduation rates between these two groups compared to students with similar parental education levels.

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]

In [12]:
```
import json
data = pd.read_csv('country-income.csv')

#Define dimensions/ measures
dimensions = ['Region', 'Age', 'Online Shopper']
measures = ['Income']

#pivot table to represent the data cube
cube_df = pd.pivot_table(data, values=measures, index=dimensions, aggfunc={
    'Income': ['sum', 'mean', 'min', 'max']
}).reset_index()

#dictionary OF cube model
cube_model = {
    "cubes": [{
        "name": "MyCube",
        "dimensions": dimensions,
        "measures": measures,
        "aggregates": {
            "sum_income": {"measure": "Income", "function": "sum"},
            "avg_income": {"measure": "Income", "function": "mean"},
            "min_income": {"measure": "Income", "function": "min"},
            "max_income": {"measure": "Income", "function": "max"}}}]}

#JSON file
with open('cube_model.json', 'w') as json_file:
    json_file.write(json.dumps(cube_model, indent=2))

print(cube_df)
```

	Region	Age	Online Shopper	Income			
				max	mean	min	sum
0	Brazil	32.0	Yes	57600.0	57600.0	57600.0	57600.0
1	Brazil	43.0	No	73200.0	73200.0	73200.0	73200.0
2	India	40.0	Yes	69600.0	69600.0	69600.0	69600.0
3	India	42.0	Yes	80400.0	80400.0	80400.0	80400.0
4	India	49.0	No	86400.0	86400.0	86400.0	86400.0
5	India	53.0	Yes	94800.0	94800.0	94800.0	94800.0
6	USA	35.0	No	64800.0	64800.0	64800.0	64800.0
7	USA	45.0	Yes	NaN	NaN	NaN	0.0
8	USA	55.0	No	99600.0	99600.0	99600.0	99600.0

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. [8 marks]

In [13]:
```
result_whole_cube = cube_df['Income'].agg(['max', 'mean', 'min', 'sum']).transpose(
print('Results for the whole data cube:')
print(result_whole_cube)

#resultsregion
result_per_region = cube_df.groupby('Region')['Income'].agg(['max', 'mean', 'min',
print('Results per region:')
print(result_per_region)

#results online shopping activity
result_per_online_shopper = cube_df.groupby('Online Shopper')['Income'].agg(['max',
print('Results per online shopping activity:')
```

```
print(result_per_online_shopper)

#results between 40 and 50
result_age_range = cube_df[cube_df['Age'].between(40, 50)]['Income'].agg(['max', 'm
print('Results for all people aged between 40 and 50:')
print(result_age_range)
```

```
Results for the whole data cube:
          max     mean      min       sum
max   99600.0  78300.0  57600.0  626400.0
mean  99600.0  78300.0  57600.0  626400.0
min   99600.0  78300.0  57600.0  626400.0
sum   99600.0  69600.0      0.0  626400.0
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3802, in Index.get_
loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File ~\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:138, in pandas._libs.ind
ex.IndexEngine.get_loc()

File ~\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:165, in pandas._libs.ind
ex.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

KeyError: 'Income'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[13], line 6
      3 print(result_whole_cube)
      5 #resultsregion
----> 6 result_per_region = cube_df.groupby('Region')['Income'].agg(['max', 'mea
n', 'min', 'sum'])
      7 print('Results per region:')
      8 print(result_per_region)

File ~\anaconda3\Lib\site-packages\pandas\core\groupby\generic.py:895, in DataFram
eGroupBy.aggregate(self, func, engine, engine_kwargs, *args, **kwargs)
    892 func = maybe_mangle_lambdas(func)
    894 op = GroupByApply(self, func, args, kwargs)
--> 895 result = op.agg()
    896 if not is_dict_like(func) and result is not None:
    897     return result

File ~\anaconda3\Lib\site-packages\pandas\core\apply.py:175, in Apply.agg(self)
    172     return self.agg_dict_like()
    173 elif is_list_like(arg):
    174     # we require a list, but not a 'str'
--> 175     return self.agg_list_like()
    177 if callable(arg):
    178     f = com.get_cython_func(arg)

File ~\anaconda3\Lib\site-packages\pandas\core\apply.py:357, in Apply.agg_list_lik
e(self)
    354 if not isinstance(obj, SelectionMixin):
    355     # i.e. obj is Series or DataFrame
    356     selected_obj = obj
--> 357 elif obj._selected_obj.ndim == 1:
    358     # For SeriesGroupBy this matches _obj_with_exclusions
    359     selected_obj = obj._selected_obj
    360 else:

File ~\anaconda3\Lib\site-packages\pandas\_libs\properties.pyx:36, in pandas._lib
s.properties.CachedProperty.__get__()

File ~\anaconda3\Lib\site-packages\pandas\core\groupby\groupby.py:755, in BaseGrou
pBy._selected_obj(self)
```

```
    753     return self.obj
    754 else:
--> 755     return self.obj[self._selection]

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:3807, in DataFrame.__getit
em__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3804, in Index.get_
loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: '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 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? [8 marks]

For x3 = (3,2), the 1-nearest neighbor classifier, which picks the closest neighbor, sees that x1 = (1,2) is the closest to x3. Since x1 belongs to Class 1 (y1 = 1), it says x3 is also in Class 1.

For x4 = (0,1), the classifier again looks for the nearest neighbor. Both x1 = (1,2) and x2 = (−1,0) are equally close to x4. Depending on their order in the training set, it might be Class 1. Based on distance, it could be seen as Class 1, following the class of x1 = (1,2).