INDIVIDUAL ASSESSMENT COVER SHEET

Faculty of Design and Creative Technologies

AUT

TE WĀNANGA ARONUI O TĀMAKI MAKAU RAU

First Name	**Vedant**	**Family Name**	**Marwadi**	**Student ID No**	**23208466**
Course Name	**Data Mining and Machine Learning**	**Course Code**	**COMP809**	**Assignment Due Date**	**12-04-2024**
Lecturer	**Dr Patricio Maturana Russel**	**Tutorial Day**	**-**	**Date Submitted**	**11-04-2024**
Tutor	**Dr Patricio Maturana Russel**	**Tutorial Time**	**-**	**No.Words/Pages**	**22**

In order to ensure fair and honest assessment results for all students, it is a requirement that the work that you hand in for assessment is your own work. If you are uncertain about any of these matters, then please discuss them with your lecturer.

Plagiarism and Dishonesty are methods of cheating for the purposes of General Academic Regulations (GAR) http://www.aut.ac.nz/calendar

Assignments will not be accepted if this section is not completed and signed.
Please read the following and **tick** ✓ to indicate your understanding:

1. I understand it is my responsibility to keep a copy of my assignment. ☒ **Yes** ☐ **No**
2. I have signed and read the **Student's Statement below**. ☒ **Yes** ☐ **No**
3. I understand that a software programme (Turnitin) that detects plagiarism and copying may be used on my assignment. ☒ **Yes** ☐ **No**

Student's Statement:

This assessment is entirely my own work and has not been submitted in any other course of study. I have submitted a copy of this assessment to Turnitin, if required. In this assessment I have acknowledged, to the best of my ability:

- The source of direct quotes from the work of others.
- The ideas of others (includes work from private or professional services, past assessments, other students, books, journals, cut/paste from internet sites and/or other materials).
- The source of diagrams or visual images.

Student's Signature: *Vedant* **Date:** **11-04-2024**

The information on this form is collected for the primary purpose of submitting your assignment for assessment. Other purposes of collection include receiving your acknowledgement of plagiarism polices and attending to administrative matters. If you choose not to complete all questions on this form, it may not be possible for the Faculty of Design and Creative Technologies to accept your assignment.

Assignment_1

April 11, 2024

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score, confusion_matrix,␣
↪accuracy_score
from sklearn.utils import resample
from scipy.stats import chi2_contingency
from scipy.stats import f_oneway
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from scipy.cluster import hierarchy
from sklearn.cluster import AgglomerativeClustering
```

1 Question 1 . A

[2]:
```
NOxEmissions = pd.read_csv('NOxEmissions.csv')
```

[3]:
```
NOxEmissions.head(3)
```

[3]:

	rownames	julday	LNOx	LNOxEm	sqrtWS
0	193	373	4.457250	5.536489	0.856446
1	194	373	4.151827	5.513000	1.016612
2	195	373	3.834061	4.886994	1.095445

[4]:
```
# it is mentioned in the requirement that the time dependency will be omitted,
# therefore droping the julday column.
# rownames column is unnecessary hence droping it as well.

NOxEmissions = NOxEmissions.drop(columns=['rownames','julday'])
```

[5]:
```python
# Checking if there are any null values
# to prevent potential errors.

null_values = NOxEmissions.isnull().sum()
print(null_values)
```

```
LNOx      0
LNOxEm    0
sqrtWS    0
dtype: int64
```

[6]:
```python
# checking if there are any duplicate values
# to ensure data integrity and accuracy in analysis.

duplicates = NOxEmissions[NOxEmissions.duplicated()]

if duplicates.empty:
    print("No duplicate values found.")
else:
    print("Duplicate values found:")
    print(duplicates)
```

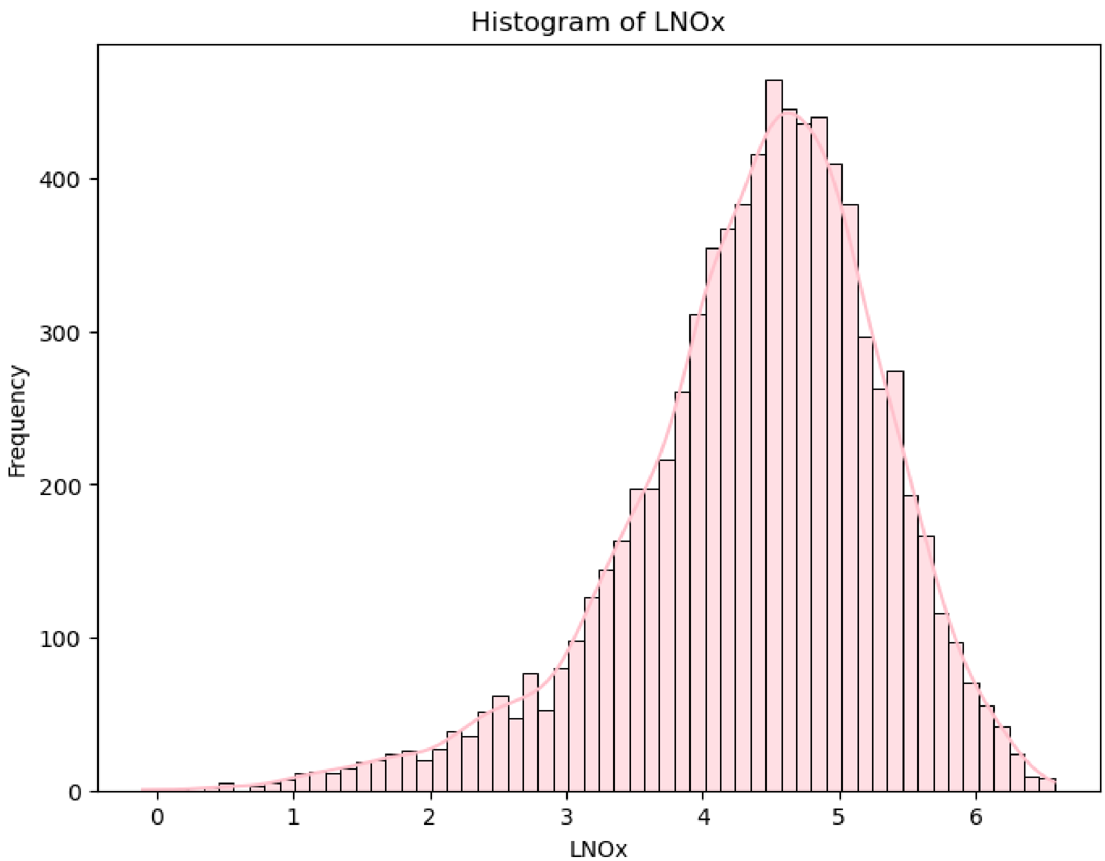
```
No duplicate values found.
```

2 Question 1 . B

[7]:
```python
summary_stats = NOxEmissions['LNOx'].describe()
print(summary_stats)
```

```
count    8088.000000
mean        4.378691
std         0.937389
min        -0.105361
25%         3.891820
50%         4.497028
75%         5.012134
max         6.576121
Name: LNOx, dtype: float64
```

[8]:
```python
plt.figure(figsize=(8, 6))
sns.histplot(NOxEmissions['LNOx'], kde=True, color='pink')
plt.title('Histogram of LNOx')
plt.xlabel('LNOx')
plt.ylabel('Frequency')
plt.show()
```



The highest frequency of LNOx values falls within the bin between 4 and 5. This means there are more data points with LNOx values between 4 and 5 than any other range. Moreover, the overall shape of the distribution appears symmetrical, which suggests the data might be normally distributed.

3 Question 1 . C

[9]:
```
X = NOxEmissions[['LNOxEm','sqrtWS']]
y = NOxEmissions['LNOx']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state = 0)

linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)

y_pred = linear_regressor.predict(X_test)
```

[10]:
```
comparison_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
comparison_df['Difference'] = comparison_df['Actual'] -␣
 ↪comparison_df['Predicted']
comparison_df.head(3)
```

[10]:

	Actual	Predicted	Difference
4947	4.945919	4.818206	0.127713
7550	4.951946	5.537413	-0.585466
4034	4.820282	4.974271	-0.153989

[11]:
```
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Absolute Error:", mae)
print("R-squared:", r2)
```

```
Mean Absolute Error: 0.42214185121597675
R-squared: 0.6864947626244496
```

It can be said that the model performs reasonably well in predicting the dependent variable because on average, the predicted values differ from the actual values by approximately 0.4221. Furethermore, around 68.65% of the variance in the target variable (LNOx) is explained by the features (LNOxEm, sqrtWS) included in the model.

4 Question 1 . D

[12]:
```
print("Coefficients:")
print("Intercept:", linear_regressor.intercept_)
print("Coefficient for LNOxEm:", linear_regressor.coef_[0])
print("Coefficient for sqrtWS:", linear_regressor.coef_[1])
```

```
Coefficients:
Intercept: 1.066912658396554
Coefficient for LNOxEm: 0.6393924443589645
Coefficient for sqrtWS: -1.0127358111457843
```

Intercept - The intercept is approximately 1.067. This implies that when the emissions of both LNOxEm and sqrtWS are zero, the predicted level of nitrogen oxides concentration (LNOx) in the air is around 1.067 units.

Coefficient for LNOxEm: - The coefficient for LNOxEm is around 0.639. This suggests that for every one-unit increase in the emissions of nitrogen oxides (LNOxEm), the predicted level of nitrogen oxides concentration (LNOx) increases by approximately 0.639 units. This increase occurs while keeping wind speed constant.

Coefficient for sqrtWS: - The coefficient for sqrtWS is approximately -1.013. This implies that for every one-unit increase in the square root of wind speed (sqrtWS), the predicted level of nitrogen oxides concentration (LNOx) decreases by approximately 1.013 units. This decrease happens while keeping nitrogen oxides emissions constant.

5 Question 1 . E

[13]:
```
new_data = pd.DataFrame({'LNOxEm': [7.5], 'sqrtWS': [1.3]})
predicted_nox_concentration = linear_regressor.predict(new_data)
print("Predicted Nitrogebn Oxides concentration:",
      predicted_nox_concentration[0])
```

```
Predicted Nitrogebn Oxides concentration: 4.5457994365992676
```

When the LNOxEm feature is 7.5 (representing the log of the hourly sum of NOx emissions of cars on the motorway) and the sqrtWS feature is 1.3 (representing the square root of wind speed), the model predicts that the Nitrogen Oxides concentration would be around 4.55 units.

6 Question 2 . A

[14]:
```
nassCDS = pd.read_csv("nassCDS.csv")
```

[15]:
```
nassCDS.head(3)
```

[15]:
```
   rownames  dvcat  weight   dead  airbag seatbelt  frontal sex  ageOFocc  \
0         1  25-39  25.069  alive    none   belted        1   f        26
1         2  10-24  25.069  alive  airbag   belted        1   f        72
2         3  10-24  32.379  alive    none     none        1   f        69

   yearacc  yearVeh    abcat occRole  deploy  injSeverity caseid
0     1997   1990.0  unavail  driver       0          3.0  2:3:1
1     1997   1995.0   deploy  driver       1          1.0  2:3:2
2     1997   1988.0  unavail  driver       0          4.0  2:5:1
```

[16]:
```
# Checking if there are any null values
# to prevent potential errors.

any_null = nassCDS.isnull().any().any()
print(any_null)
```

```
True
```

[17]:
```
# Dropping all the null values to ensure
# that the dataset is complete and reliable.

nassCDS.dropna(inplace=True)
```

[18]:
```
# checking if there are any duplicate values
# to ensure data integrity and accuracy in analysis.


duplicates = nassCDS[nassCDS.duplicated()]
```

```
if duplicates.empty:
    print("No duplicate values found.")
else:
    print("Duplicate values found:")
    print(duplicates)
```

```
No duplicate values found.
```

[19]:

```
# Selecting specific features mentioned in the requirements

nassCDS_with_selected_features = nassCDS[['dead','airbag','seatbelt','frontal',
                                          'sex','ageOFocc','yearVeh','deploy']]
```

[20]:

```
nassCDS_with_selected_features.head(3)
```

[20]:

```
    dead   airbag seatbelt  frontal sex  ageOFocc  yearVeh  deploy
0  alive     none   belted        1   f        26   1990.0       0
1  alive   airbag   belted        1   f        72   1995.0       1
2  alive     none     none        1   f        69   1988.0       0
```

[21]:

```
# Converting categorical variables into appropriate dummy variables
# to represent them numerically
# where,
# dead_dead = 0 if alive; 1 if dead
# sex_m = 0 if female; 1 if male
# seatbelt_none = 0 if belted; 1 if none
# airbag_none = 0 if airbag ; 1 if none

data = pd.get_dummies(nassCDS_with_selected_features,
                      columns=["dead", "sex", "seatbelt", "airbag"],
                      drop_first=True, dtype=int)
```

[22]:

```
data.head(3)
```

[22]:

```
   frontal  ageOFocc  yearVeh  deploy  dead_dead  sex_m  seatbelt_none  \
0        1        26   1990.0       0          0      0              0
1        1        72   1995.0       1          0      0              0
2        1        69   1988.0       0          0      0              1

   airbag_none
0            1
1            0
2            1
```

7 Question 2 . B

[23]:
```
# H0: The use of seat belts is independent of whether
# the passenger survives or not.
# H1: The use of seat belts is not independent of whether
# the passenger survives or not.

contingency_table = pd.crosstab(nassCDS_with_selected_features['seatbelt'],
                                nassCDS_with_selected_features['dead'])
print(contingency_table)
```

```
dead      alive  dead
seatbelt
belted    17965   500
none       6918   680
```

[24]:
```
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("p-value:", p)

if p < 0.05:
    print("There is evidence to reject the null hypothesis.")
else:
    print("There is no evidence to reject the null hypothesis.")
```

```
p-value: 3.2511305843401275e-107
There is evidence to reject the null hypothesis.
```

Given such a small p-value, we would reject the null hypothesis H0 in favor of the alternative hypothesis H1. Therefore, we have strong evidence to conclude that the use of seat belts is not independent of whether the passenger survives or not. In other words, there is a statistically significant relationship between the use of seat belts and survival.

8 Question 2 . C

[25]:
```
# H0: There is no difference in mean age across
# the different injury severity groups.
# H1: There is a difference in mean age across
# the different injury severity groups.

f_statistic, p_value = f_oneway(nassCDS[nassCDS['injSeverity'] == 0]
                                ['ageOFocc'],
                                 nassCDS[nassCDS['injSeverity'] == 1]
                                ['ageOFocc'],
                                 nassCDS[nassCDS['injSeverity'] == 2]
                                ['ageOFocc'],
                                 nassCDS[nassCDS['injSeverity'] == 3]
                                ['ageOFocc'],
                                 nassCDS[nassCDS['injSeverity'] == 4]
```

```
                                    ['ageOFocc'])

print("F-statistic:", f_statistic)
print("p-value:", p_value)

if p_value < 0.05:
    print("There is a statistically significant difference in mean age across␣
 ↪injury severity groups.")
else:
    print("There is no statistically significant difference in mean age across␣
 ↪injury severity groups.")
```

```
F-statistic: 78.26858783063506
p-value: 4.1325230342567886e-66
There is a statistically significant difference in mean age across injury
severity groups.
```

Given such a small p-value, we would reject the null hypothesis H0 in favor of the alternative hypothesis H1. Therefore, we have strong evidence to conclude that there is a statistically significant difference in mean age across the different injury severity groups.

9 Question 2 . D

[26]:
```
#Before fitting the data into the model, checking if the data is unbalanced

response_count = data.groupby("dead_dead")["dead_dead"].count()
print(response_count);
print("Percentage of 0s:", 100*response_count[0]/np.sum(response_count))
print("Percentage of 1s:", 100*response_count[1]/np.sum(response_count))
```

```
dead_dead
0    24883
1     1180
Name: dead_dead, dtype: int64
Percentage of 0s: 95.47250892069216
Percentage of 1s: 4.527491079307831
```

It is found that the data is unbalanced.

[27]:
```
# Note that if we train the model with this data set,
# the model could just predict any response
# Therefore, using oversampling to balance the data

data_minority = data[(data['dead_dead']==1)]
data_majority = data[(data['dead_dead']==0)]
data_minority_upsampled = resample(data_minority,
                                    replace=True,
                                    n_samples= response_count[0],
```

```
                                    random_state=123);
data_minority_upsampled.reset_index(drop=True, inplace=True)

data_upsampled = pd.concat([data_minority_upsampled, data_majority])
```

[28]:

```
response_count = data_upsampled.groupby("dead_dead")["dead_dead"].count();
print(response_count);
print("Percentage of 0s:", 100*response_count[0]/np.sum(response_count));
print("Percentage of 1s:", 100*response_count[1]/np.sum(response_count));
```

```
dead_dead
0    24883
1    24883
Name: dead_dead, dtype: int64
Percentage of 0s: 50.0
Percentage of 1s: 50.0
```

It could be seen that the data is now balanced.

[29]:

```
X = data_upsampled[['airbag_none', 'seatbelt_none', 'frontal',
                    'sex_m', 'ageOFocc', 'yearVeh', 'deploy']]
y = data_upsampled['dead_dead']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=42)

train_data = pd.concat([X_train, y_train], axis=1)


model = sm.GLM.from_formula("dead_dead ~ airbag_none + seatbelt_none + frontal␣
 ↪+ sex_m + ageOFocc + yearVeh + deploy",
                            family=sm.families.Binomial(), data=train_data)


result = model.fit()
y_pred_prob = result.predict(X_test)
y_pred_binary = (y_pred_prob > 0.5).astype(int)
```

[30]:

```
# Calculating confusion matrix

cm = confusion_matrix(y_test, y_pred_binary)

# Calculating sensitivity (also known as recall or true positive rate)

sensitivity = cm[1, 1] / (cm[1, 1] + cm[1, 0])

# Calculating accuracy
```

```
accuracy = accuracy_score(y_test, y_pred_binary)

print("Confusion Matrix:")
print(cm)
print("\nSensitivity (Recall):", sensitivity)
print("Accuracy:", accuracy)
```

```
Confusion Matrix:
[[5057 2406]
 [2404 5063]]

Sensitivity (Recall): 0.6780500870496853
Accuracy: 0.6778298727394507
```

The model seems to be performing well based on the high values on the diagonal (5057 and 5063). These numbers represent a significant number of correctly classified instances for both classes. Also, looking at the values, it appears the class distribution might be balanced. The counts for True Positives (5057) and True Negatives (5063) are quite similar. This was expected since we used over sampling to balance the dataset. Furthermore, the model seems to have fairly balanced performance, with sensitivity and accuracy both around 67.8%. This means it correctly predicted the outcome (whether a person survived or not) for about 67.8% of the cases in the test dataset and correctly identified about 67.8% of the deceased individuals.. While the overall accuracy seems good, there are still errors (2404 and 2406).

10 Queston 2 . E

[31]:
```
print(result.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              dead_dead   No. Observations:                34836
Model:                            GLM   Df Residuals:                    34828
Model Family:                Binomial   Df Model:                            7
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -20472.
Date:                Thu, 11 Apr 2024   Deviance:                       40944.
Time:                        19:46:31   Pearson chi2:                 3.49e+04
No. Iterations:                     4   Pseudo R-squ. (CS):             0.1902
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-
Intercept        -1.6549      6.170     -0.268      0.789     -13.747
10.438
airbag_none       1.0492      0.045     23.537      0.000       0.962
```

```
1.137
seatbelt_none      1.4144      0.025     55.775      0.000       1.365
1.464
frontal           -1.0911      0.026    -41.296      0.000      -1.143
-1.039
sex_m              0.2502      0.025     10.163      0.000       0.202
0.298
ageOFocc           0.0262      0.001     41.472      0.000       0.025
0.027
yearVeh           -0.0002      0.003     -0.056      0.956      -0.006
0.006
deploy             0.8635      0.039     22.387      0.000       0.788
0.939
================================================================================
=
```

1. **Seatbelt**:
    - The coefficient associated with `seatbelt_none` is 1.4144.
    - This positive parameter suggests that not wearing a seatbelt is associated with a higher likelihood of a person being dead in a car crash.
    - In simpler terms, individuals who do not wear seatbelts are more likely to experience fatal outcomes in car accidents compared to those who wear seatbelts.
    - The standard error of 0.025 suggests a high level of confidence in this estimate.
    - The coefficient has a p-value < 0.05, indicating statistical significance. This implies that the relationship between seatbelt usage and the outcome variable is likely not due to random chance.
2. **Age**:
    - The coefficient associated with `ageOFocc` is 0.0262.
    - This positive parameter suggests that an increase in age is associated with a higher likelihood of a person being dead in a car crash.
    - In other words, as individuals get older, their risk of experiencing fatal outcomes in car accidents tends to increase.
    - The standard error of 0.001 suggests a very high level of confidence in this estimate
    - The coefficient has a p-value < 0.05, indicating statistical significance. This suggests that the relationship between the age of the occupant and the outcome variable is likely not due to random chance.

11 Question 2 . F

11.1 1.

[32]:
```
new_data_1 = pd.DataFrame({'airbag_none': [1], 'seatbelt_none': [1],
                           'frontal': [1], 'sex_m': [0],
                           'ageOFocc': [70], 'yearVeh': [1993], #using median
                           'deploy': [0]})

predicted_prob_not_surviving = result.predict(new_data_1)
```

```python
# Converting probabilities to odds

odds_not_surviving = predicted_prob_not_surviving / (1 -␣
↪predicted_prob_not_surviving)

print("Odds of not surviving:", odds_not_surviving.values[0])

percentage_not_surviving = (odds_not_surviving / (odds_not_surviving + 1)) * 100
print("Percentage chance of not surviving:", percentage_not_surviving.values[0])
```

```
Odds of not surviving: 3.3545372607979407
Percentage chance of not surviving: 77.03544739408761
```

If we assume that the odds of surviving are 1, then the odds of not surviving would be approximately 3.35. This means that according to the logistic regression model and the described scenario:

- The likelihood of not surviving is approximately 3.35 times higher compared to the likelihood of surviving.
- Out of 100 similar scenarios, based on the model's predictions, we would expect approximately 77 scenarios where the individual does not survive and approximately 23 scenarios where the individual survives.

11.2 2

[33]:
```python
new_data_2 = pd.DataFrame({'airbag_none': [0], 'seatbelt_none': [0],
                           'frontal': [1], 'sex_m': [0],
                           'ageOFocc': [70], 'yearVeh': [1993], #using median
                           'deploy': [1]})

predicted_prob_not_surviving = result.predict(new_data_2)

# Converting probabilities to odds

odds_not_surviving = predicted_prob_not_surviving / (1 -␣
↪predicted_prob_not_surviving)

print("Odds of not surviving:", odds_not_surviving.values[0])

percentage_not_surviving = (odds_not_surviving / (odds_not_surviving + 1)) * 100
print("Percentage chance of not surviving:", percentage_not_surviving.values[0])
```

```
Odds of not surviving: 0.6771918120685391
Percentage chance of not surviving: 40.37652743089264
```

If we assume that the odds of surviving are 1, then the odds of not surviving would be approximately 0.68. This means that according to the logistic regression model and the described scenario:

- The likelihood of not surviving is approximately 0.68. That indicates that not surviving is less likely compared to surviving.

- Out of 100 similar scenarios, based on the model's predictions, we would expect approximately 40 scenarios where the individual does not survive and approximately 60 scenarios where the individual survives.

12 Question 3 . A

[34]:
```
data_q3 = pd.read_excel("data_q3.xlsx")
```

[35]:
```
data_q3.head(3)
```

[35]:
```
   country_x code  Tertiary Percentage  ISCED5 Percentage  ISCED6 Percentage  \
0  Argentina  ARG            95.447912          18.103877          68.238077
1  Australia  AUS           115.952037          25.407825          65.591820
2    Austria  AUT            86.475597          15.080255          40.310180

   ISCED7 Percentage  ISCED8 Percentage  country_y  year  \
0           8.368618           0.737339  Argentina  2019
1          21.327540           3.624852  Australia  2019
2          27.126033           3.959066    Austria  2019

   InternationalStudentsNO  …  KOFFiGIdf  KOFFiGIdj  \
0                   116330  …         65         55
1                   509160  …         81         75
2                    74631  …         89         80

   KOFSoGI_WithoutInterpersonal  InboundRatio  top_50_count  top_100_count  \
0                          78.0       3.50011             0              1
1                          94.5      28.37490             5              7
2                          90.5      17.64123             0              0

   top_500_count  top_1000_count  total_ranked_universities        WESP
0              5              15                         15  Developing
1             25              37                         37   Developed
2              5               8                          8   Developed

[3 rows x 45 columns]
```

[36]:
```
#selecting specific features mentioned in the requirements

variables = ['InboundRatio', 'InternationalStudentsNO', 'KOFPoGI', 'KOFEcGI',
             'KOFSoGI','ISCED5 Percentage', 'ISCED6 Percentage',
             'ISCED7 Percentage', 'ISCED8 Percentage',
             'top_50_count', 'top_100_count', 'top_500_count',
             'top_1000_count', 'WESP', 'country_x']

data_q3 = data_q3[variables].copy()
```

```
data_q3.head(3)
```

[36]:
```
   InboundRatio  InternationalStudentsNO  KOFPoGI  KOFEcGI  KOFSoGI  \
0       3.50011                   116330       91       48       72
1      28.37490                   509160       88       68       88
2      17.64123                    74631       95       83       88

   ISCED5 Percentage  ISCED6 Percentage  ISCED7 Percentage  ISCED8 Percentage  \
0          18.103877          68.238077           8.368618           0.737339
1          25.407825          65.591820          21.327540           3.624852
2          15.080255          40.310180          27.126033           3.959066

   top_50_count  top_100_count  top_500_count  top_1000_count        WESP  \
0             0              1              5              15  Developing
1             5              7             25              37   Developed
2             0              0              5               8   Developed

   country_x
0  Argentina
1  Australia
2    Austria
```

[37]:
```
# Checking if there are any null values
# to prevent potential errors.

data_q3.isnull().any().any()
```

[37]: True

[38]:
```
# Dropping all the null values to ensure
# that the dataset is complete and reliable.

data_q3.dropna(inplace=True)
```

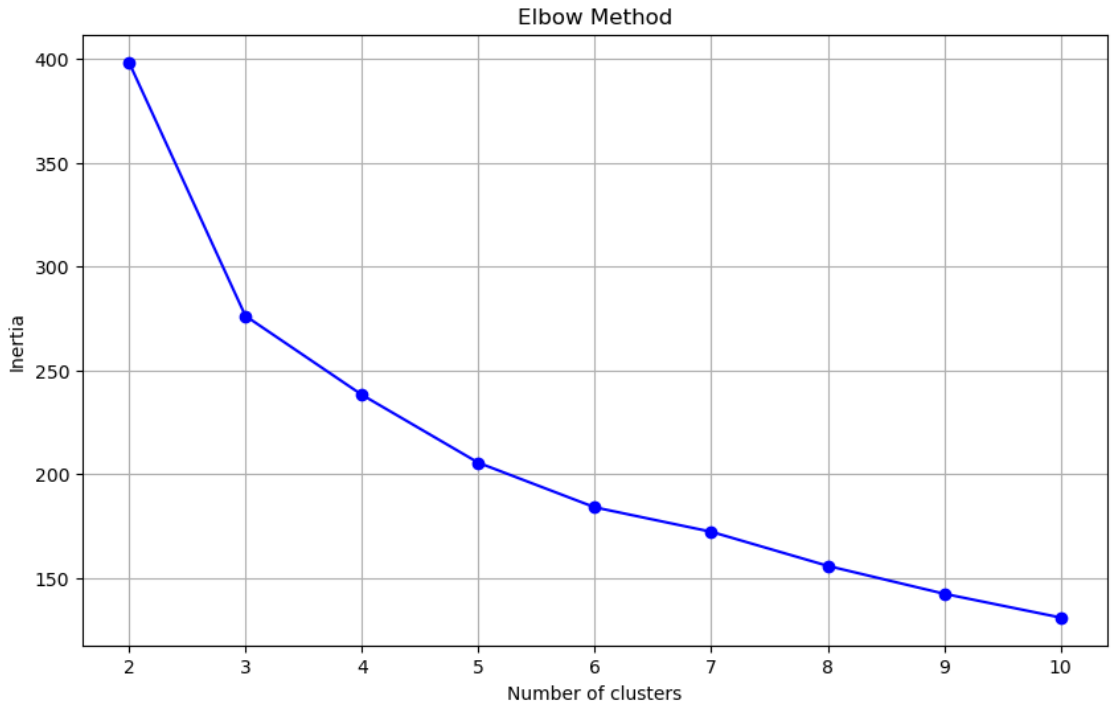
[39]:
```
# Selecting only the numerical columns for clustering
# to avoid potential issues with categorical variables.

numerical_cols = ['InboundRatio', 'InternationalStudentsNO', 'KOFPoGI',
                  'KOFEcGI', 'KOFSoGI', 'ISCED5 Percentage',
                  'ISCED6 Percentage', 'ISCED7 Percentage',
                  'ISCED8 Percentage', 'top_50_count',
                  'top_100_count', 'top_500_count', 'top_1000_count']

dataframe = data_q3[numerical_cols]
```

[40]:
```
# Scaling the data to ensure uniformity in feature contributions
# by bringing them to a comparable scale.
```

```
scaler = StandardScaler()
fitted = scaler.fit(dataframe)
dataframe_scaled = pd.DataFrame(fitted.transform(dataframe))
```

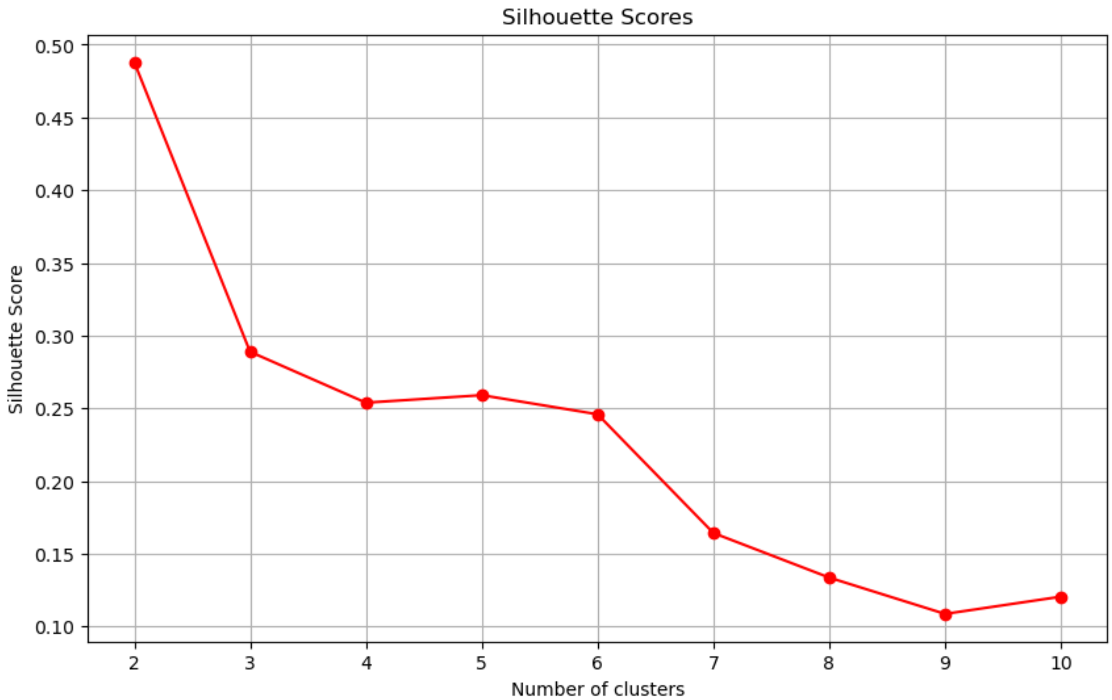
13 Question 3 . B

[41]:
```
# Suppressing the warning about memory leak on Windows with MKL
warnings.filterwarnings("ignore", message="KMeans is known to have a memory␣
 ↪leak on Windows with MKL")

inertia = []
silhouette_scores = []
range_values = range(2, 11)

for i in range_values:
    kmeans = KMeans(n_clusters=i, random_state=42,n_init=10)
    kmeans.fit(dataframe_scaled)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(dataframe_scaled, kmeans.labels_))

# Plotting the Elbow Method graph

plt.figure(figsize=(10, 6))
plt.plot(range_values, inertia, '-o', color='blue')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.xticks(range_values)
plt.grid(True)
plt.show()
```



The inertia graph shows a noticeable bend at 3 clusters, indicating that beyond this point, the decrease in inertia (sum of squared distances to the nearest cluster center) becomes less significant. This suggests that 3 clusters might be a good choice.

[42]:
```
# Plotting the Silhouette Scores

plt.figure(figsize=(10, 6))
plt.plot(range_values, silhouette_scores, '-o', color='red')
plt.title('Silhouette Scores')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.xticks(range_values)
plt.grid(True)
plt.show()
```



The silhouette scores, which measure how similar an object is to its own cluster compared to other clusters, peak at 3 clusters as well. This indicates that the clustering solution with 3 clusters has, on average, better-defined clusters.

Both methods suggest that 3 clusters would be the optimal number for this dataset, providing a balance between cluster cohesion and separation.

[50]:
```
# Running K-means clustering with 3 clusters

kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
kmeans.fit(dataframe_scaled)

# Adding the cluster labels to the original dataframe

dataframe.loc[:, 'Cluster'] = kmeans.labels_

dataframe.head()
```

[50]:

	InboundRatio	InternationalStudentsNO	KOFPoGI	KOFEcGI	KOFSoGI	\
0	3.50011	116330	91	48	72	
1	28.37490	509160	88	68	88	
2	17.64123	74631	95	83	88	
3	10.04272	52143	96	89	86	
4	0.24504	21803	90	42	62	

```
   ISCED5 Percentage  ISCED6 Percentage  ISCED7 Percentage  ISCED8 Percentage  \
0          18.103877          68.238077           8.368618           0.737339
1          25.407825          65.591820          21.327540           3.624852
2          15.080255          40.310180          27.126033           3.959066
3           3.399620          58.107011          15.999636           2.631904
4           0.004350          53.314007           1.083925           0.734018

   top_50_count  top_100_count  top_500_count  top_1000_count  Cluster
0             0              1              5              15        2
1             5              7             25              37        0
2             0              0              5               8        0
3             0              1              7               8        0
4             0              0              5              22        2
```

[44]:
```
# Using PCA to reduce dimensionality and plot the clusters in two dimensions.

# Separating clusters

cluster_0 = dataframe[dataframe['Cluster'] == 0].drop('Cluster', axis=1)
cluster_1 = dataframe[dataframe['Cluster'] == 1].drop('Cluster', axis=1)
cluster_2 = dataframe[dataframe['Cluster'] == 2].drop('Cluster', axis=1)

# Applying PCA on all data

pca = PCA(n_components=2)
X_pca = pca.fit_transform(dataframe.drop('Cluster', axis=1))

print("Explained variance ratio:", pca.explained_variance_ratio_)
```

```
Explained variance ratio: [9.99999972e-01 9.30702125e-09]
```

- The first principal component explains approximately 99.9999972% of the variance in the original data.
- The second principal component explains approximately 9.30702125e-09% (or essentially 0%) of the variance in the original data.

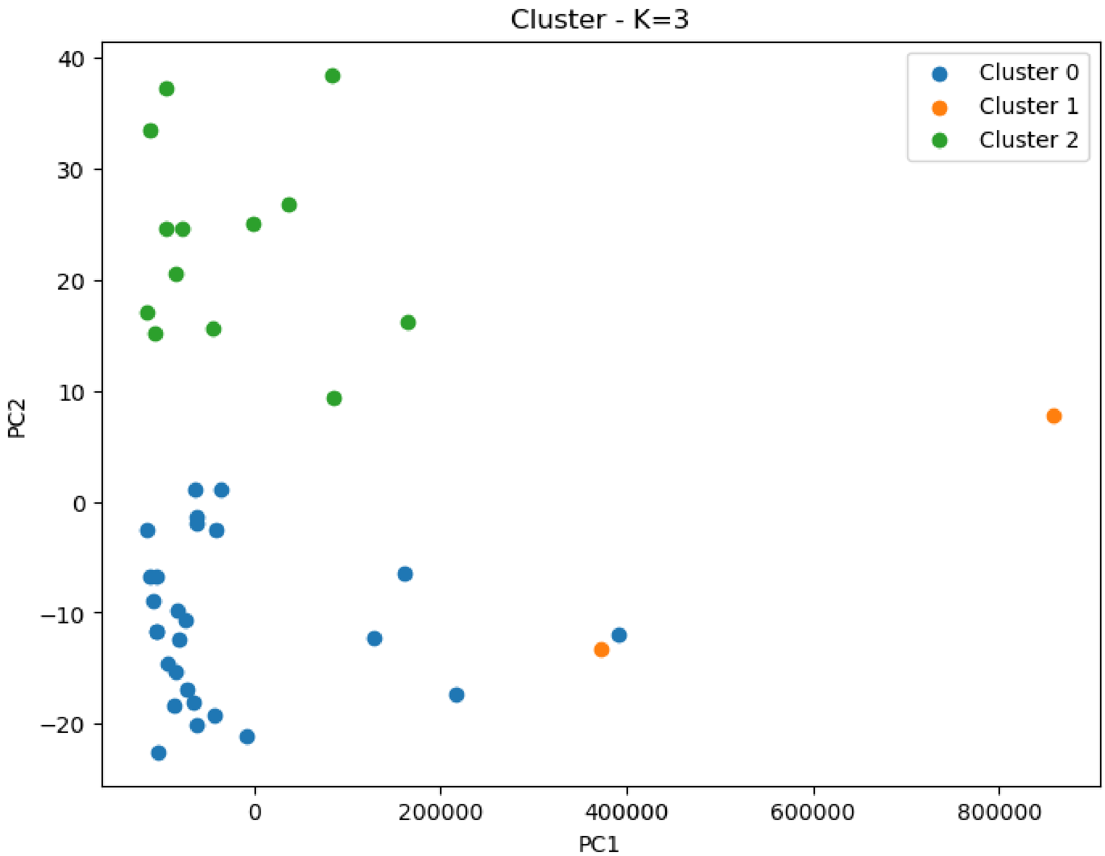
Given that the first principal component captures almost all of the variance in the data, reducing the dimensionality to just one dimension might be sufficient for this dataset.

However, we're still reducing the dimensionality to two components. While the second component doesn't contribute much to explaining the variance, it still helps visualize the data in two dimensions.

Therefore, the choice of using PCA with two components can be justified

[45]:
```
# Applying PCA transformation to each cluster

cluster_0_pca = pca.transform(cluster_0)
cluster_1_pca = pca.transform(cluster_1)
cluster_2_pca = pca.transform(cluster_2)
```

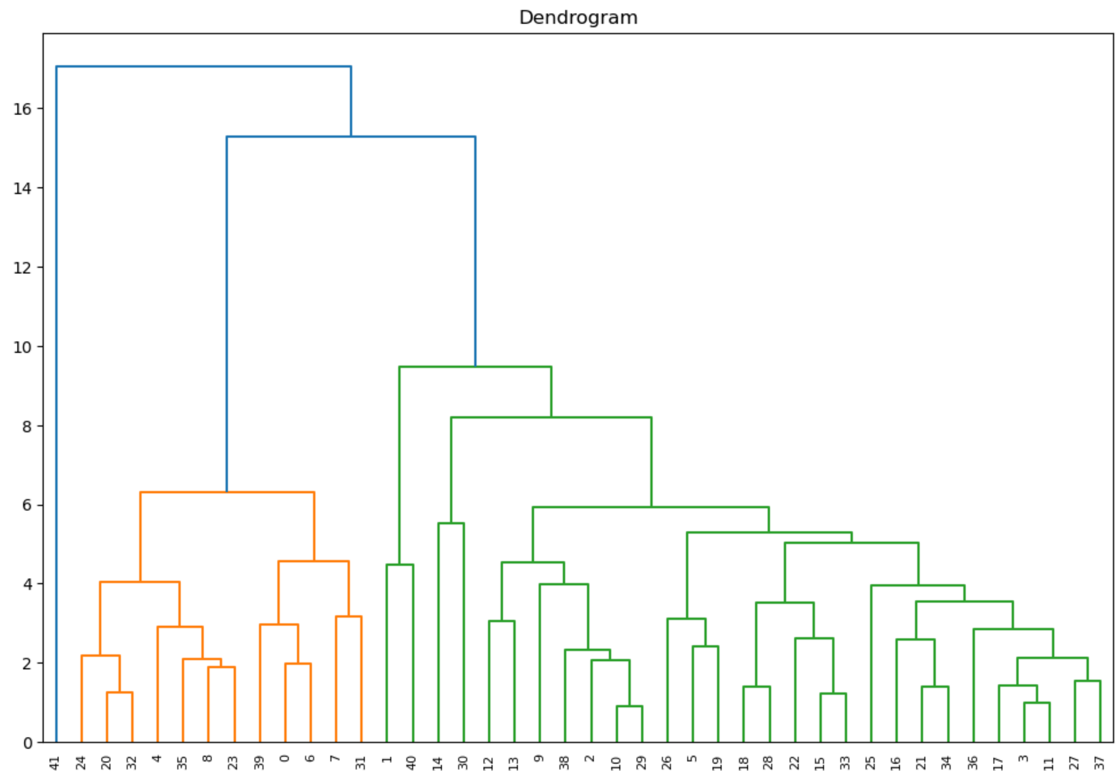
```python
# Plotting

plt.figure(figsize=(8, 6))
plt.scatter(cluster_0_pca[:, 0], cluster_0_pca[:, 1], label='Cluster 0')
plt.scatter(cluster_1_pca[:, 0], cluster_1_pca[:, 1], label='Cluster 1')
plt.scatter(cluster_2_pca[:, 0], cluster_2_pca[:, 1], label='Cluster 2')
plt.legend()
plt.title('Cluster - K=3')
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()
```



The scatter plot with the first 2 PCs shows a clear difference between the clusters. However, minor overlapping could be seen bewteen cluster 0 and 1.

14 Question 3 . C

[46]:
```
plt.figure(figsize=(12, 8))
dendrogram = hierarchy.dendrogram(hierarchy.linkage(dataframe_scaled,
                                                     method='ward'),
                                  labels=dataframe_scaled.index)
plt.title('Dendrogram')
plt.show()
```



[47]:
```
model = AgglomerativeClustering(n_clusters=3, linkage="ward",
                                compute_distances = True);
model.fit(dataframe_scaled);

dataframe.loc[:, "Cluster"] = model.labels_
```

[48]:
```
# Using PCA to reduce dimensionality and plot the clusters in two dimensions.

# Separating clusters

cluster_0 = dataframe[dataframe['Cluster'] == 0].drop('Cluster', axis=1)
cluster_1 = dataframe[dataframe['Cluster'] == 1].drop('Cluster', axis=1)
cluster_2 = dataframe[dataframe['Cluster'] == 2].drop('Cluster', axis=1)
```

```
# Applying PCA on all data

pca = PCA(n_components=2)
X_pca = pca.fit_transform(dataframe.drop('Cluster', axis=1))

print("Explained variance ratio:", pca.explained_variance_ratio_)
```

```
Explained variance ratio: [9.99999972e-01 9.30702125e-09]
```

- The first principal component explains approximately 99.9999972% of the variance in the original data.
- The second principal component explains approximately 9.30702125e-09% (or essentially 0%) of the variance in the original data.

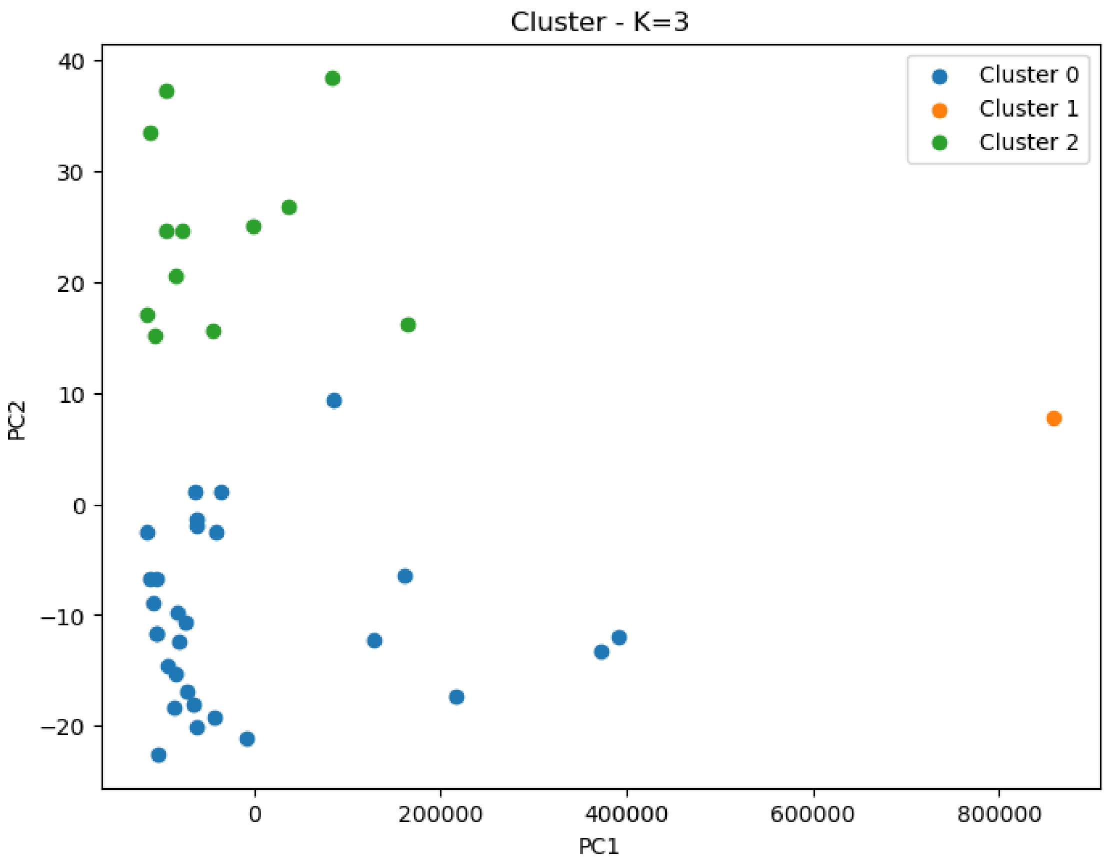
Given that the first principal component captures almost all of the variance in the data, reducing the dimensionality to just one dimension might be sufficient for this dataset.

However, we're still reducing the dimensionality to two components. While the second component doesn't contribute much to explaining the variance, it still helps visualize the data in two dimensions.

Therefore, the choice of using PCA with two components can be justified

[49]:

```
# Applying PCA transformation to each cluster

cluster_0_pca = pca.transform(cluster_0)
cluster_1_pca = pca.transform(cluster_1)
cluster_2_pca = pca.transform(cluster_2)

# Plotting

plt.figure(figsize=(8, 6))
plt.scatter(cluster_0_pca[:, 0], cluster_0_pca[:, 1], label='Cluster 0')
plt.scatter(cluster_1_pca[:, 0], cluster_1_pca[:, 1], label='Cluster 1')
plt.scatter(cluster_2_pca[:, 0], cluster_2_pca[:, 1], label='Cluster 2')
plt.legend()
plt.title('Cluster - K=3')
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()
```



The scatter plot with the first 2 PCs shows a clear difference between the clusters. There is no overlapping of clusters using this approach.

15 Question 3 . D

The findings from both K-Means and Agglomerative Clustering analyses indicate that the optimal number of clusters for the dataset is three. This consistency across different clustering methods strengthens the confidence in the proposed clustering structure. In the K-Means analysis, there is noticeable overlap or intermingling between Cluster 0 and Cluster 1. Such overlap implies that these clusters might share similar characteristics or features in the principal component space. In contrast, Agglomerative Clustering exhibited superior performance in segregating the clusters effectively, showcasing clearer boundaries between them.