ASSQ1

April 11, 2024

1 Question 1a

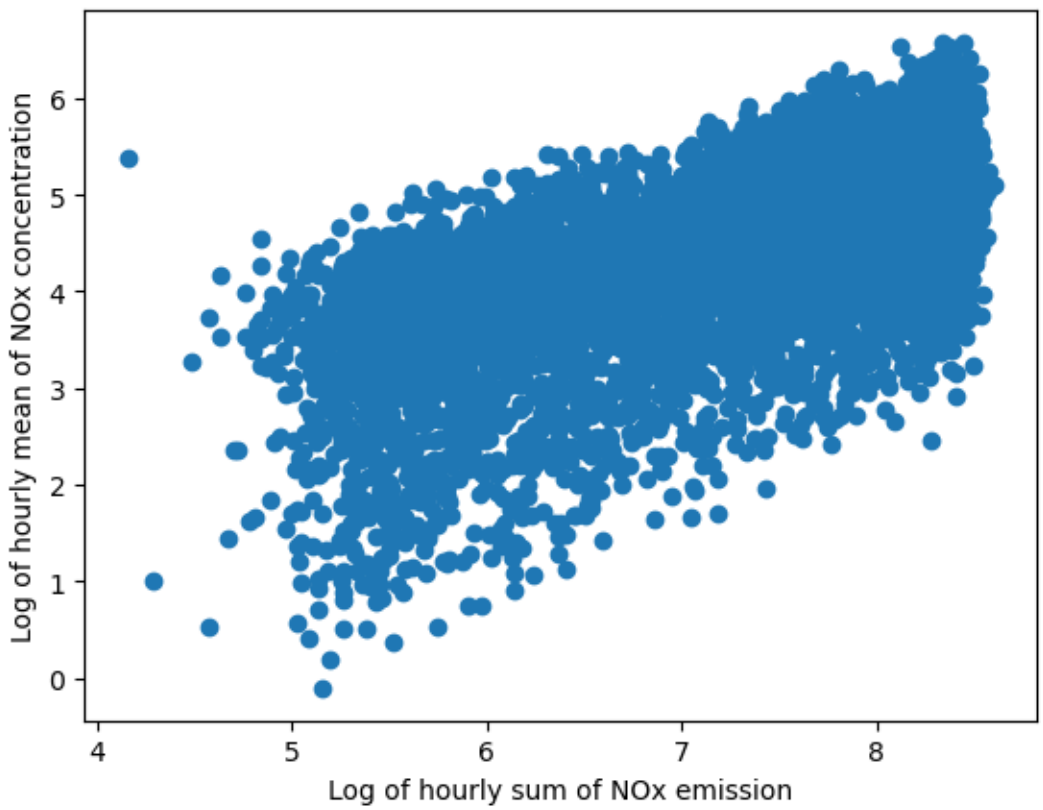
[6]:
```
#Q1 Reding the data into Python
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.stats.api as sms
import statsmodels.api as sm
from scipy import stats
df = pd.read_csv("NOxEmissions.csv");
print(df.head())
```

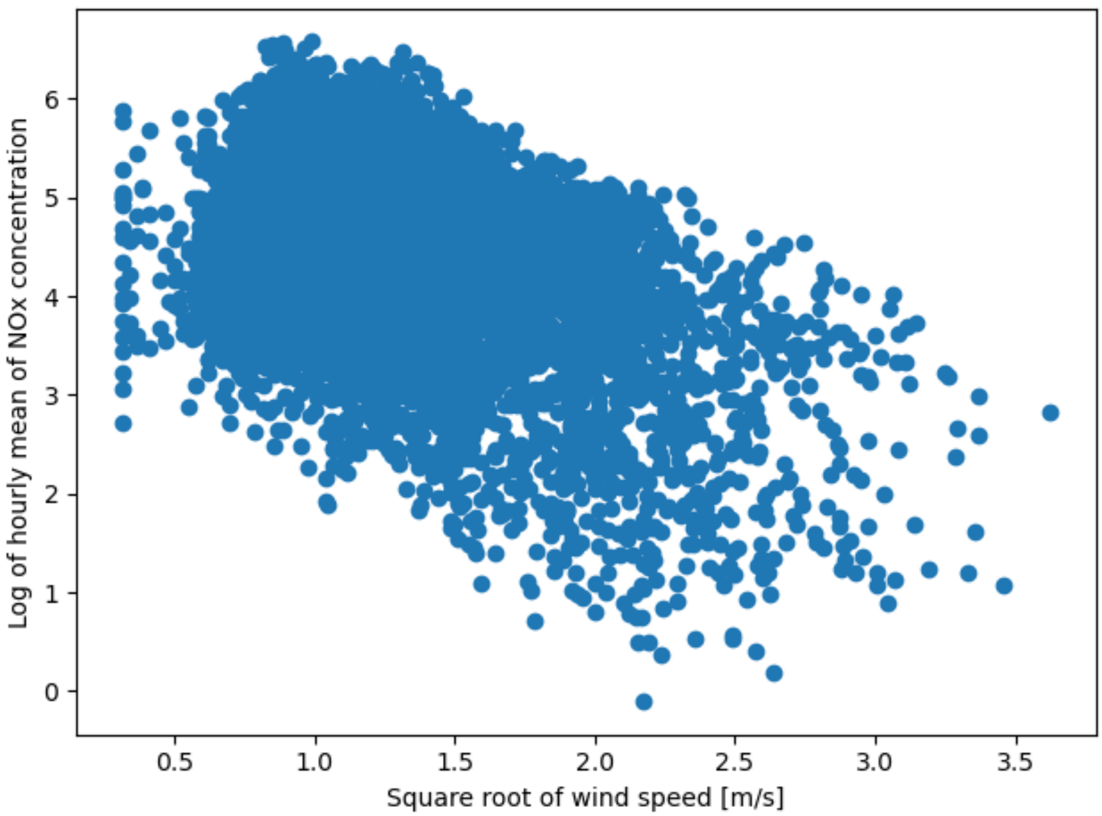
```
   rownames  julday      LNOx    LNOxEm    sqrtWS
0       193     373  4.457250  5.536489  0.856446
1       194     373  4.151827  5.513000  1.016612
2       195     373  3.834061  4.886994  1.095445
3       196     373  4.172848  5.138912  1.354068
4       197     373  4.322807  5.666518  1.204159
```

[7]:
```
# Q1a Data preprocessing.
print("Number of observation: ", df.shape[0])# check dimension
print("Any NA value:", df.isnull().values.any()); #check for missing values
print("Any row duplictaes:",df.duplicated().any());#check for dupllicates rows

#Check for data error(negative values)
print("Number of negative values in 'julday':", ((df['julday'] < 373) |␣
↪(df['julday'] > 730)).sum())
print("number of negative vaules in 'LNOx':",(df['LNOx']< 0).values.sum())
print("number of negative vaules in 'LNOxEm':",(df['LNOxEm']< 0).values.sum())
print("number of negative vaules in 'sqrtWS':",(df['sqrtWS']< 0).values.sum())

#Check if all the values are numeric
df.dtypes
```

```python
# Check for ouliers using scatter plot
plt.scatter(df["LNOxEm"],df["LNOx"])
plt.xlabel("Log of hourly sum of NOx emission")
plt.ylabel("Log of hourly mean of NOx concentration")
plt.show()
plt.scatter(df["sqrtWS"],df["LNOx"])
plt.xlabel("Square root of wind speed [m/s]")
plt.ylabel("Log of hourly mean of NOx concentration")
plt.tight_layout()
plt.show()
# check normalization
df[["LNOx","LNOxEm","sqrtWS"]].describe()
```

```
Number of observation:  8088
Any NA value: False
Any row duplictaes: False
Number of negative values in 'julday': 0
number of negative vaules in 'LNOx': 1
number of negative vaules in 'LNOxEm': 0
number of negative vaules in 'sqrtWS': 0
```





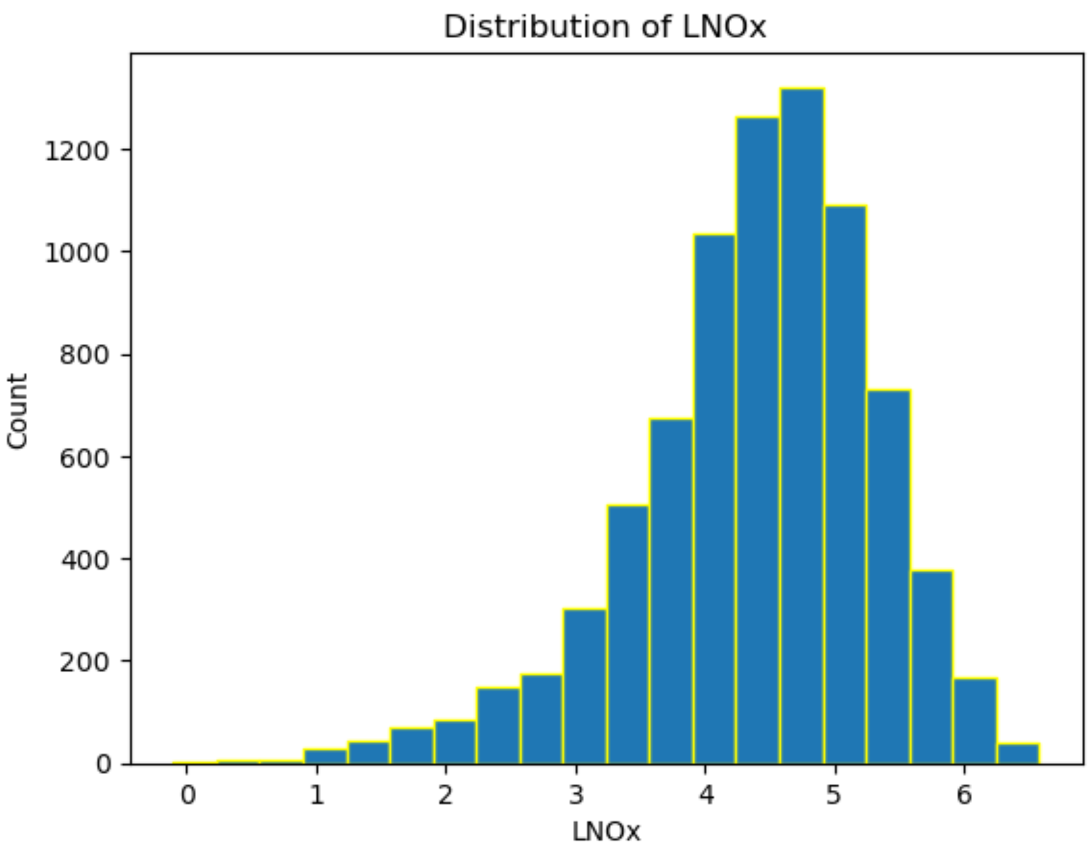
[7]:

	LNOx	LNOxEm	sqrtWS
count	8088.000000	8088.000000	8088.000000
mean	4.378691	7.338244	1.365253
std	0.937389	1.016658	0.466280
min	-0.105361	4.157866	0.316228
25%	3.891820	6.514982	1.016612
50%	4.497028	7.692495	1.284523
75%	5.012134	8.239159	1.648181
max	6.576121	8.600040	3.624017

In this dataset, we have 8088 observations with no missing values and no row duplicates.'julday' is considered as a discrete variable, whereas 'LLNOx', 'LNOxEm'and 'sqrtWS'are considered as continuous numeric variables. We observe one negative value in 'LN0x', which would not make sense in this study due to negative concentration. There are no obvious outliers present in 'LNOx' and 'sqrtWS' as indicated by scatter plots. Since we work with numerical variables, there is no need to worry about data balancing. Moreover, there is no need to standardize the data as they have similar scales, as indicated below. Finally, we could extract the features we want out of the data set to fit a linear model.

2 Question1 b

[8]:
```
plt.hist(df['LNOx'], bins = 20, edgecolor = "yellow")# plotting a histogram for␣
↪LNOx
plt.title('Distribution of LNOx')
plt.xlabel('LNOx')
plt.ylabel('Count')
plt.show()
# fluctuation range
print(df["LNOx"].min()) # Min value
print(df["LNOx"].max()) # Max value
print(df["LNOx"].mean())# Mean
print(df["LNOx"].median())#Median
```




```
-0.105360515657826
6.57612131900117
4.378690810185019
4.49702802736839
```

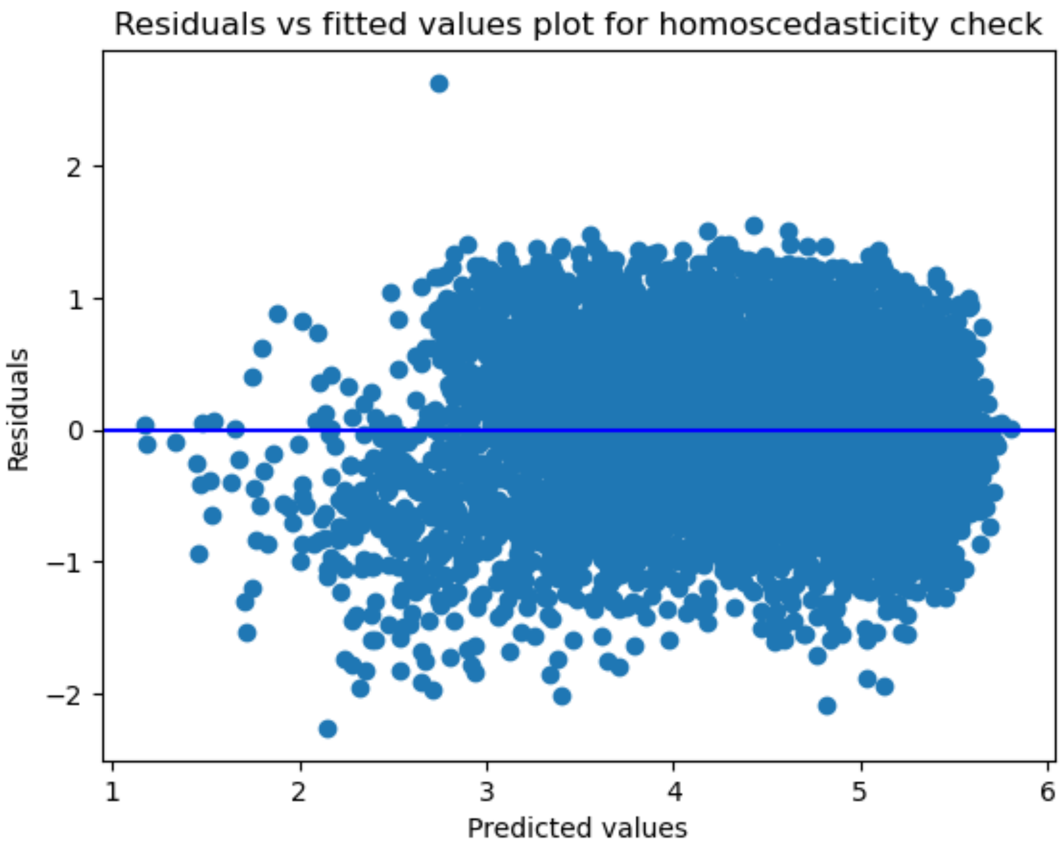
This distribution plot indicates that the data is centered around 4.5, fluctuating between -0.1 and 6.5. This distribution is left-skewed, meaning higher concentrations happen more often than lower *ones, as we can see in the long left tail.*

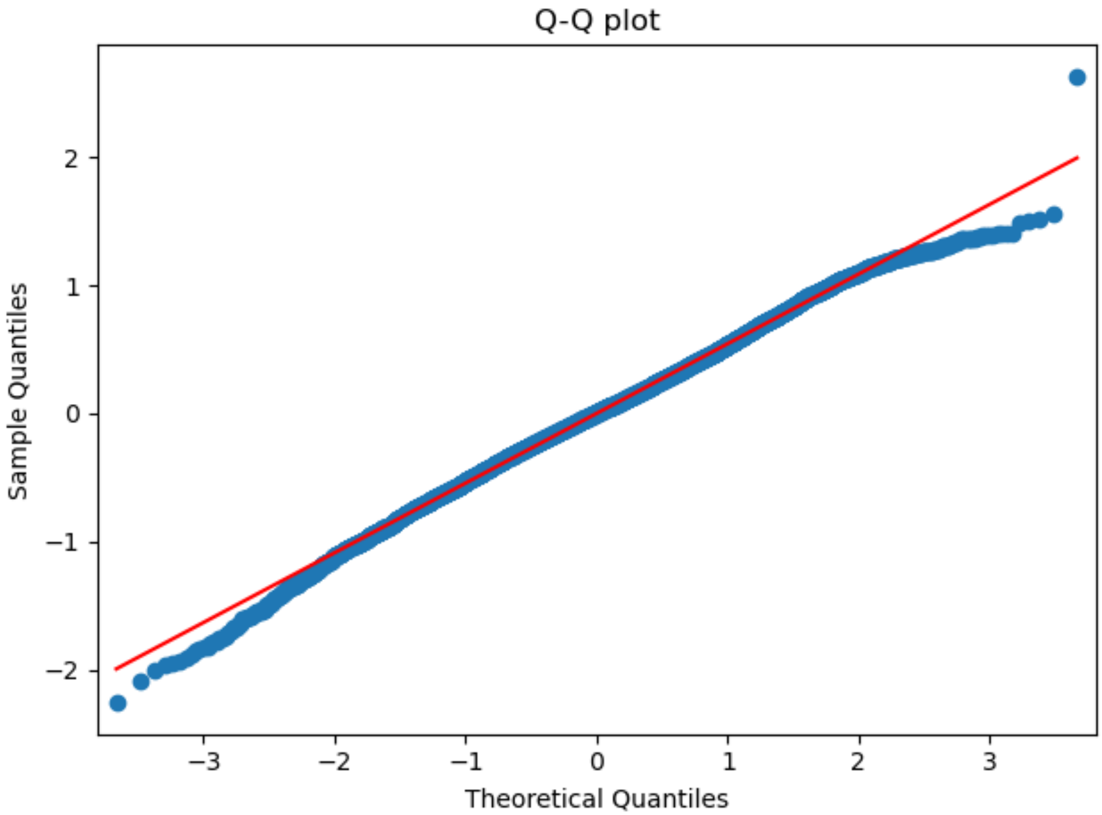
3 Question1 c

[9]:
```python
# Q1c
import statsmodels.formula.api as smf
# Fitting a linear model
mod= smf.ols("LNOx~ LNOxEm + sqrtWS" , data = df) # spefify the model
model = mod.fit() # Fit the data into the linear model
print(model.summary())
# check residual
ypred= model.predict(df)
residuals = np.array(df['LNOx']) - np.array(ypred) # from list to an array
p = plt.scatter(ypred, residuals)
plt.xlabel("Predicted values")
plt.ylabel("Residuals")
plt.axhline(y = 0.0, color = 'b', linestyle = '-')
p = plt.title("Residuals vs fitted values plot for homoscedasticity␣
↪check")
plt.show()
# QQ plot
sm.qqplot(residuals, line='s');
plt.title('Q-Q plot')
plt.tight_layout()
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   LNOx   R-squared:                       0.663
Model:                            OLS   Adj. R-squared:                  0.663
Method:                 Least Squares   F-statistic:                     7952.
Date:                Thu, 11 Apr 2024   Prob (F-statistic):               0.00
Time:                        21:51:02   Log-Likelihood:                -6554.7
No. Observations:                8088   AIC:                         1.312e+04
Df Residuals:                    8085   BIC:                         1.314e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      1.0619      0.046     23.097      0.000       0.972       1.152
LNOxEm         0.6414      0.006    107.092      0.000       0.630       0.653
sqrtWS        -1.0182      0.013    -77.969      0.000      -1.044      -0.993
==============================================================================
Omnibus:                       28.937   Durbin-Watson:                   0.497
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               30.943
Skew:                          -0.115   Prob(JB):                     1.91e-07
```

```
Kurtosis:                       3.198   Cond. No.                       58.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```





The model has an R-squared value of 0.663, indicating that 66.3% of the total variability in LNOx is explained by this model, suggesting the model fits the data okay. The result also indicates that those two predictors are statistically significant(P-values = 0), and they are useful for prediction. The condition number looks okay, so there is no serious multicollinearity issue. Based on those two plots, both constant variance and normality are approximately satisfied. Since we have a very large sample size, the normality assumption is not needed at all, as the Central Limit Theorem ensures that the distribution of residuals will approximate normality.

4 Question1 d

There is a positive linear relationship between LNOx and LNOxEm, meaning that as the NOx emission from the cars increases, the concentration increases as well. The relationship between LNOx and sqrtWS is negative, meaning that as the wind speed increases, the concentration tends to decrease. Interpretation of coefficients: For every unit increase in the hourly sum of NOx emission of cars(on the logarithm scale), the hourly mean of Nox concentration is expected, on average, to increase by 0.64 ppb (on the logarithmic scale), keeping other factors constant. For every unit increase in the square root of wind speed, we expect the log of the hourly mean of Nox concentration to decrease by 1.01 ppb(on a logarithmic scale), keeping other factors constant.

5 Question1 e

[10]:
```
#Q1e Predict the Nitrogen Oxides concentration for a LNOxEm = 7.5 and sqrtWS =␣
↪1.3.
pred_con = {"LNOxEm" :[7.5],"sqrtWS":[1.3]}
pred_con = pd.DataFrame(data = pred_con)# Creating a new dataframe for␣
↪prediction
prediction = model.predict(pred_con) # Use selected model to predict the␣
↪concentration.
print("The predicted log of hourly mean of NOx concentration is:"␣
↪,str(round(prediction[0],3))+ " ppb")
```

```
The predicted log of hourly mean of NOx concentration is: 4.549 ppb
```

Interpretation: If the log of hourly sum emission from cars on the motorway is 7.5 and the square root of wind speed is 1.3 m/s, the NOx concentration in the air near the motorway is predicted to be approximately 4.55 ppb.

With a more intuitive interpretation, if the hourly sum emission from cars on the motorway is e^7.5 = 1808.04 and the wind speed is 1.69 m/s, then the predicted hourly mean of NOx concentration is 94.54 ppb.

ASSQ2

April 11, 2024

1 Question2 a

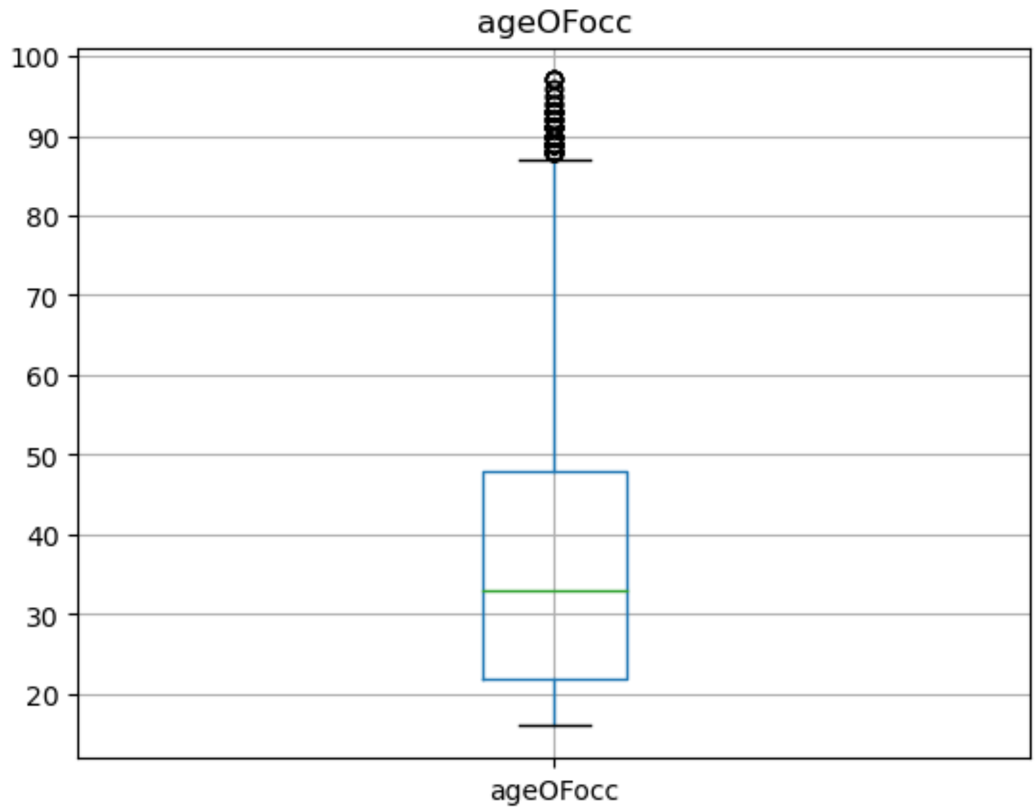
[1]:
```
#Q2 reading the data into Python
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.stats.api as sms
import statsmodels.api as sm
from statsmodels.stats.stattools import durbin_watson
from scipy import stats
from sklearn.utils import resample
import scipy;
df2 = pd.read_csv("nassCDS.csv");
print(df2.head())
```

```
   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
3         4  25-39  495.444  alive   airbag   belted        1   f        53
4         5  25-39   25.069  alive     none   belted        1   f        32

   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
3     1997   1995.0   deploy  driver       1          1.0   2:10:1
4     1997   1988.0  unavail  driver       0          3.0   2:11:1
```

[2]:
```
# Q1a Data preprocessing.
print("Number of observation: ", df2.shape[0])      # check dimension
print("Any NA value:", df2.isnull().values.any()); # Check for missing values
print("Any row duplictaes:",df2.duplicated().any());# check for dupllicates rows
```

```
df2 = df2.dropna() # drop all the NA values
#check for date error among all the variables of interestes.
print("Number of error values in 'dead':", ((df2['dead']!= "alive")&␣
 ↪(df2['dead'] != "dead")).sum())
print("Number of error values in 'seltbelt':", ((df2['seatbelt']!= "belted")&␣
 ↪(df2['seatbelt'] != "none")).sum())
print("Number of error values in 'frontal':", ((df2['frontal']!= 0)&␣
 ↪(df2['frontal'] != 1)).sum())
print("Number of error values in 'airbag':", ((df2['airbag']!= "none")&␣
 ↪(df2['airbag'] != "airbag")).sum())
print("Number of error values in 'sex':", ((df2['sex']!= "m")& (df2['sex'] !=␣
 ↪"f")).sum())
print("Number of error values in 'sex':", ((df2['sex']!= "m")& (df2['sex'] !=␣
 ↪"f")).sum())
print("Number of error values in 'ageOFocc':", ((df2['ageOFocc']<0) |␣
 ↪(df2['ageOFocc']>100)).sum())
print("Number of error values in 'deploy':", ((df2['deploy']!= 1)&␣
 ↪(df2['deploy'] != 0)).sum())
# Check outlier for numeric variable 'ageOFocc'
df2.boxplot("ageOFocc")
plt.title('ageOFocc')
plt.tight_layout
plt.show()
# Check data types
print(df2.dtypes)
# Check for data balancing
response_count = df2.groupby("dead")["dead"].count();
print(response_count);
print("Percentage of alive:", 100*response_count[0]/np.sum(response_count));
print("Percentage of dead:", 100*response_count[1]/np.sum(response_count));
print(df2.shape)
df2.reset_index(drop=True, inplace=True)
```

```
Number of observation:  26217
Any NA value: True
Any row duplictaes: False
Number of error values in 'dead': 0
Number of error values in 'seltbelt': 0
Number of error values in 'frontal': 0
Number of error values in 'airbag': 0
Number of error values in 'sex': 0
Number of error values in 'sex': 0
Number of error values in 'ageOFocc': 0
Number of error values in 'deploy': 0
```




```
rownames         int64
dvcat           object
weight         float64
dead            object
airbag          object
seatbelt        object
frontal          int64
sex             object
ageOFocc         int64
yearacc          int64
yearVeh        float64
abcat           object
occRole         object
deploy           int64
injSeverity    float64
caseid          object
dtype: object
dead
alive    24883
dead      1180
Name: dead, dtype: int64
```

```
Percentage of alive: 95.47250892069216
Percentage of dead: 4.527491079307831
(26063, 16)
```

In this dataset, we have 26217 observations with missing values and no duplicate rows. There is no obvious data error in the dataset, as all the values are plausible. There are some outliers on the upper side in age, as indicated by the box plot. Since we work with categorical variables, there is no need to perform any standardization. However, feature selection plays a crucial role in the later part of this question, such as finding the relation of two categorical variables(Chi-square, ANOVA). More importantly, we have unbalanced data in this question, and we are going to use oversampling techniques to balance it(This is performed in later parts).Before the analysis, we drop all the NA values.

2 Question2 b

[3]:
```python
#chi-square is used to determine whether two categorical are independent or not
 ↪("seatbelt" and "dead")
from scipy.stats import chi2_contingency
# Converting the characters in data set into 0s and 1s for simplicity.
# Replace 'alive' with 1 and 'dead' with 0
df2['dead'].replace({'alive': 1, 'dead': 0}, inplace=True)
# Replace 'belted' with 1 and 'none' with 0
df2['seatbelt'].replace({'belted': 1, 'none': 0},inplace = True)
# Replace 'airbag' with 1 and 'none' with 0
df2['airbag'].replace({'airbag': 1, 'none': 0},inplace = True)
# Replace 'm' with 1 and 'f' with 0
df2['sex'].replace({'m': 1, 'f': 0},inplace = True)

# Now we convert 'seatbelt' and 'dead' to category type for Chi-square analysis
df_chi = df2[["seatbelt","dead"]].astype("category")
# Hypothesis:
#H0: the features are independent
#H1: the features are not independent
contingency_table = pd.crosstab(df_chi['seatbelt'], df_chi['dead'])# Generate
 ↪contigency table

# Perform the Chi-square test
chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)
print("Statistics:",chi2_stat)
print("p-value:", round(p_value,2))
print("Degrees of freedom:", dof)
```

```
Statistics: 483.7579238069683
p-value: 0.0
Degrees of freedom: 1
```

Since the P-value is approximately zero, we have very strong evidence against the null hypothesis. We have strong evidence that 'seatbelt' and 'dead' are not independent, which is what we expect in *real life. In conclusion, we have enough evidence to keep the variable 'seatbelt' in the analysis that aims to explain the variable 'dead'.*

3 Question2 c

[4]:
```
# ANOVA is used to analyze the mean age difference between injury severity␣
↪groups.
from scipy.stats import ttest_ind
from scipy.stats import f_oneway

df_none = df2[df2["injSeverity"]== 0]; # dataset for none injury
df_possible = df2[df2["injSeverity"]== 1]; # dataset for possible injury
df_no = df2[df2["injSeverity"]== 2];# dataset for no incapacity injury
df_incapacity = df2[df2["injSeverity"]== 3];#dataset for incapacity injury
df_killed = df2[df2["injSeverity"]== 4];#dataset for killed injury
# Apply Oneway ANOVA
#hypothesis:
#H0:There is no age mean difference.
#H1: There is age mean differnce between injury severity groups.
print(f_oneway(df_none["ageOFocc"],␣
↪df_possible["ageOFocc"],df_no["ageOFocc"],df_incapacity["ageOFocc"],
               df_killed["ageOFocc"]));
```

```
F_onewayResult(statistic=78.26858783063506, pvalue=4.1325230342567886e-66)
```

The p-value is zero. Therefore, we have strong evidence against H0. There is sufficient statistical evidence to claim that the injury severity groups have different means. Therefore, it is not appropriate to exclude the variable experiment from the analysis.

4 Question2 d

[5]:
```
response_count = df2.groupby("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));
# We use overampling to balance our data.
df_minority = df2[(df2['dead']==0)];
df_majority = df2[(df2['dead']==1)];
df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples= response_count[1], # to match␣
↪majority class
                                 random_state=123);  # reproducible results
df_minority_upsampled.reset_index(drop=True, inplace=True); # reseting row␣
↪numbers
df_upsampled = pd.concat([df_minority_upsampled, df_majority]);
response_count = df_upsampled.groupby("dead")["dead"].count();
```

```
print(response_count); # Check for data balancing again and make sure they are␣
↪equal.


#train the model and fit
X =␣
↪df_upsampled[["airbag","seatbelt","frontal","sex","ageOFocc","yearVeh","deploy"]]#␣
↪explannatory variables
y = df_upsampled[['dead']];# response variable

# Here we define training and testing sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=0);

data_train = pd.concat([X_train, y_train], axis = 1)#trained dataset

#model= sm.GLM.from_formula("dead ~ C(airbag) + C(seatbelt) + C(frontal) +␣
↪C(sex) + ageOFocc + yearVeh + C(deploy) ", family = sm.families.Binomial(),
                             #data=data_train);
#result= model.fit();
#print(result.summary());

#Since the'yearVeh' is not significant(P-value greater the 0.05), we remove it␣
↪from the model.

model = sm.GLM.from_formula("dead ~ C(airbag) + C(seatbelt) + C(frontal) +␣
↪C(sex) + ageOFocc + C(deploy)",
                            family=sm.families.Binomial(),
                            data=data_train)
result = model.fit();
print(result.summary()); # Now all the variables are significant with p-values␣
↪less than 0.05.

#Check Over_dispersion
dev = result.deviance; # Residual Deviance
dof = result.df_resid; # Degree of freedoms of Residuals
pvalue = 1 - scipy.stats.chi2.cdf(dev, dof); # p-value
# H0: Logistic regression model provides an adequate fit for the data
# H1: Logistic regression model does not provide an adequate fit for the data
if pvalue < 0.05:
    print("Saturated model -- p-value: ", pvalue);
else :
    print("Logistic model is ok -- p-value=", pvalue);

# Calculation of Pearson chi2 / n - (p+1)
print("Pearson2 / Df",result.pearson_chi2 / result.df_resid);
```

```
# This value is close to 1
# We also fit a quasi-binomial model
result_quasi = model.fit(scale="X2");
print(result_quasi.summary());

# Predictions and model evaluation(Accuracy, sensetivity and specificity)
predictions = result.predict(X_test);
predictions_nominal = [ 0 if x < 0.5 else 1 for x in predictions];
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y_test, predictions_nominal)
print("Confusion matrix:", cm);
# The diagonal elements of the confusion matrix indicate correct predictions,
# while the off-diagonals represent incorrect predictions
print("Accuracy: ", round(np.sum(np.diagonal(cm))/np.sum(cm),3));
print("Sensitivity: ", round(cm[1,1]/np.sum(cm[1,:]),3));
print("Specificity: ", round(cm[0,0]/np.sum(cm[0,:]),3));
# We can also get those values as follows
print(classification_report(y_test, predictions_nominal,digits = 3))
```

```
dead
0     1180
1    24883
Name: dead, dtype: int64
Percentage of 0s: 4.527491079307831
Percentage of 1s: 95.47250892069216
dead
0    24883
1    24883
Name: dead, dtype: int64
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                   dead   No. Observations:                34836
Model:                            GLM   Df Residuals:                    34829
Model Family:                Binomial   Df Model:                            6
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -20487.
Date:                Thu, 11 Apr 2024   Deviance:                       40973.
Time:                        21:51:30   Pearson chi2:                 3.48e+04
No. Iterations:                     4   Pseudo R-squ. (CS):             0.1895
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept           -0.4583      0.039    -11.619      0.000      -0.536
-0.381
```

```
C(airbag)[T.1]       1.0322      0.036     28.521      0.000       0.961
1.103
C(seatbelt)[T.1]     1.4126      0.025     55.962      0.000       1.363
1.462
C(frontal)[T.1]      1.0829      0.026     41.036      0.000       1.031
1.135
C(sex)[T.1]         -0.2578      0.025    -10.479      0.000      -0.306
-0.210
C(deploy)[T.1]      -0.8494      0.039    -21.967      0.000      -0.925
-0.774
ageOFocc            -0.0261      0.001    -41.231      0.000      -0.027
-0.025
================================================================================
====
Saturated model -- p-value:  0.0
Pearson2 / Df 0.9990645482163427
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                   dead   No. Observations:                34836
Model:                            GLM   Df Residuals:                    34829
Model Family:                Binomial   Df Model:                            6
Link Function:                  Logit   Scale:                         0.99906
Method:                          IRLS   Log-Likelihood:                -20487.
Date:                Thu, 11 Apr 2024   Deviance:                       40973.
Time:                        21:51:30   Pearson chi2:                 3.48e+04
No. Iterations:                     6   Pseudo R-squ. (CS):             0.1895
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept           -0.4583      0.039    -11.624      0.000      -0.536
-0.381
C(airbag)[T.1]       1.0322      0.036     28.534      0.000       0.961
1.103
C(seatbelt)[T.1]     1.4126      0.025     55.988      0.000       1.363
1.462
C(frontal)[T.1]      1.0829      0.026     41.055      0.000       1.031
1.135
C(sex)[T.1]         -0.2578      0.025    -10.484      0.000      -0.306
-0.210
C(deploy)[T.1]      -0.8494      0.039    -21.977      0.000      -0.925
-0.774
ageOFocc            -0.0261      0.001    -41.250      0.000      -0.027
-0.025
================================================================================
```

```
====
Confusion matrix: [[5152 2357]
 [2430 4991]]
Accuracy:  0.679
Sensitivity:  0.673
Specificity:  0.686
              precision    recall  f1-score   support

           0      0.680     0.686     0.683      7509
           1      0.679     0.673     0.676      7421

    accuracy                          0.679     14930
   macro avg      0.679     0.679     0.679     14930
weighted avg      0.679     0.679     0.679     14930
```

The scale parameter is 0.999 from the quasi-binomial model, which is very close to 1. Hence, the logistic regression model provides an adequate fit for the data, even though this hypothesis was rejected according to the chi-square test above.

The logistic regression correctly predicted the survival statuses 67.9% of the time. The model correctly predicted 67.3% of the time those who survived car accidents. The model correctly predicted 68.6 % of the time those who died of car accidents.

5 Question2 e

ageOFocc : For every unit increase in age(one year), we expect that the odds of surviving decrease by a factor of (exp(-0.0261))= 0.974,keeping other factors constant, which means that as people get older, the odds of survival decreases.

Seatbelt: The expected odds of survival for those who have their seatbelt fastened over the odds of survival for those who do not increase by a factor of exp(1.41)=4.1, which means that people with seatbelt on would help save lives.

6 Question2 f

[6]:
```
## Q2f prediction
pred_1 = {"airbag":[0], "seatbelt":[0], "frontal":[1],"sex":[0],"deploy":
 ↪[0],"ageOFocc":[70]};
pred_1 = pd.DataFrame(data=pred_1);
pred_prob1 = result.predict(pred_1);# probability of survial for senario 1
prob_not1 = 1-pred_prob1[0] # probability of death for senario 1
odds_of_not1 = prob_not1/(1-prob_not1) # odds of not survial(death)is␣
 ↪calculated by p(not)/(1-p(not))



pred_2 = {"airbag":[1], "seatbelt":[1], "frontal":[1],"sex":[0],"deploy":
 ↪[1],"ageOFocc":[70]};
```

```python
pred_2 = pd.DataFrame(data=pred_2);
pred_prob2 = result.predict(pred_2);# probability of survial for senario 2
prob_not2 = 1-pred_prob2[0]# probability of death for senario 2
odds_of_not2 = prob_not2/(1-prob_not2)# odds of not survial(death)is calculated␣
 ↪by p(not)/(1-p(not))
print("The odds of not surving for scenario 1 is ", odds_of_not1)
print("The odds of not surving for scenario 2 is ",odds_of_not2 )
```

```
The odds of not surving for scenario 1 is  3.3208866098275056
The odds of not surving for scenario 2 is  0.6734880020488657
```

For the first scenario, where there is no airbag, the seatbelt is not fastened, the accident is frontal, and the person is 70 years old woman with the airbag not deployed, the odds of not surviving is 3.32, meaning that the person is 3.32 more likely to not survive with above conditions than to survive.

For the second scenario, where there is an airbag, the seatbelt is fastened, the accident is frontal, the person is 70 years old woman with the airbag being deployed, the odds of not surviving is 0.67, meaning that the person is 0.67 times more likely(less likely indeed) to not survive under those conditions than to survive.

Those predictions are indeed plausible as airbags and seatbelts play important roles in saving people's lives on the road in reality.

ASSQ3

April 11, 2024

1 Question3 a

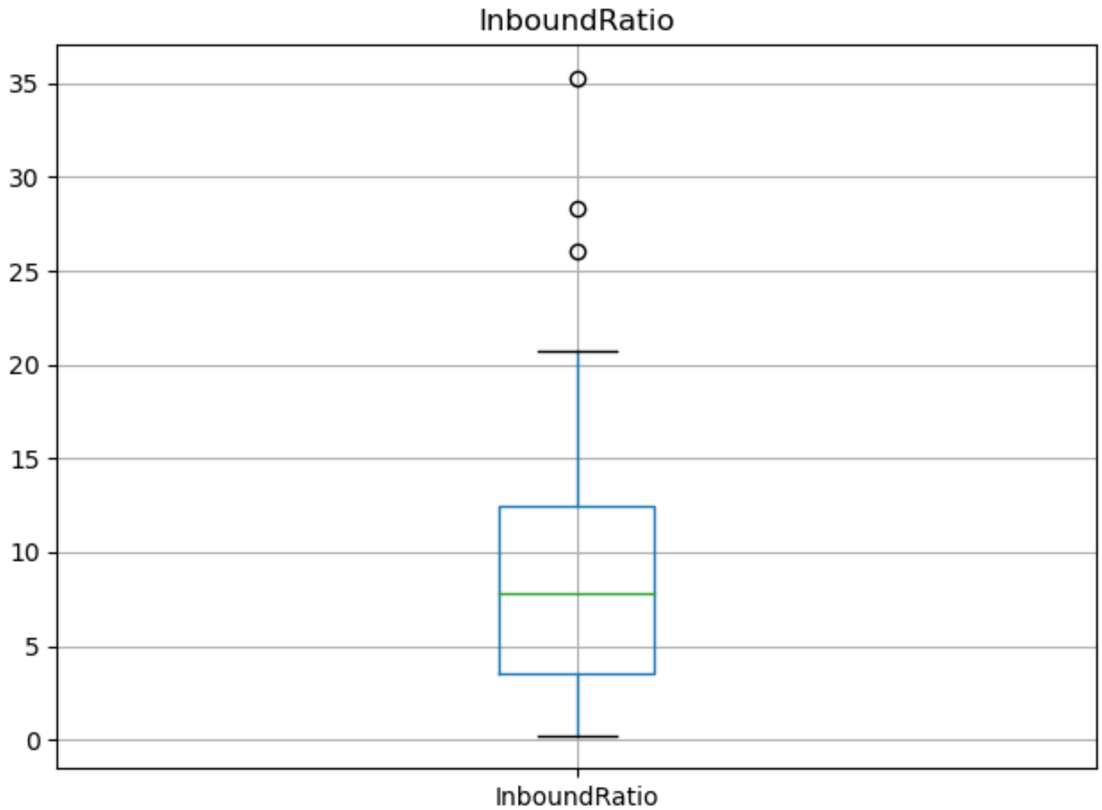
[10]:
```
# Reading the data into Python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from matplotlib.ticker import MaxNLocator
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.utils import resample
import warnings
df = pd.read_excel("data_q3.xlsx");
warnings.filterwarnings("ignore")
```

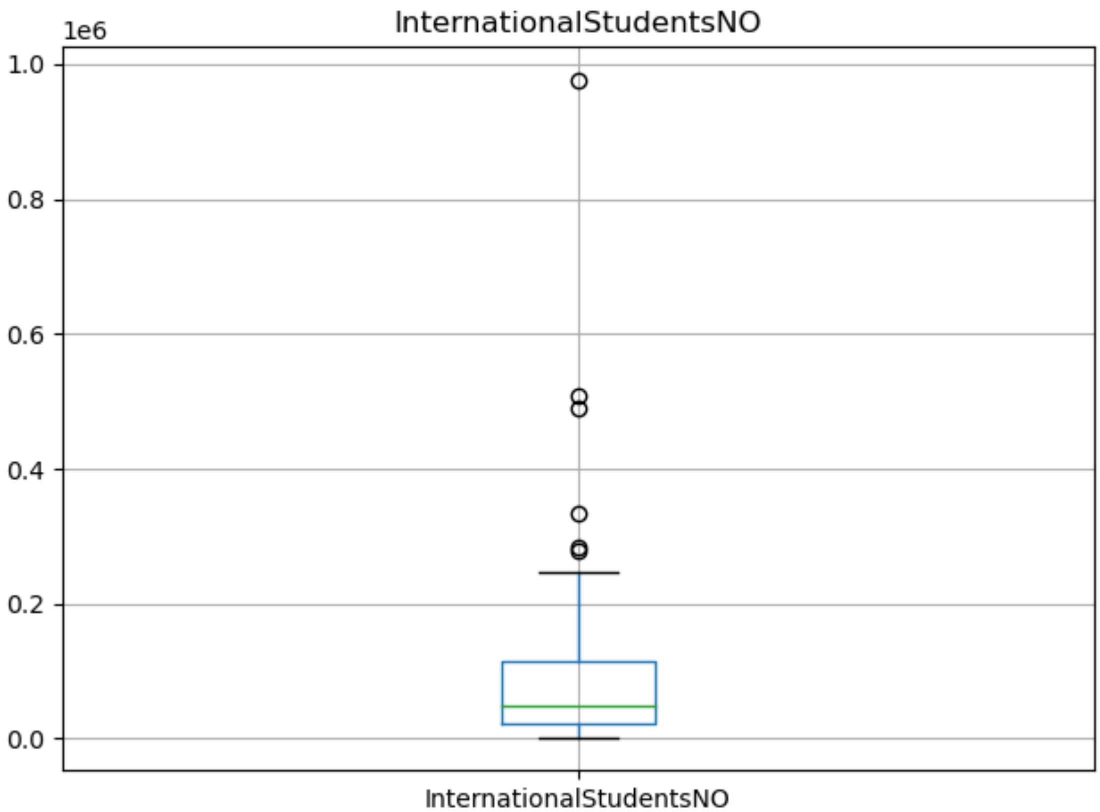
[11]:
```
# Q3a Data preprocessing.
print("Number of observation: ", df.shape[0])     # check dimension
print("Any NA value:", df.isnull().values.any()); # Check for missing values
print("Any row duplictaes:",df.duplicated().any());# check for dupllicates rows
df = df.dropna()
df.reset_index(drop=True, inplace=True)
#Check for data error(negative values)
num_error = (df.select_dtypes(include=['float64', 'int64']) < 0).sum()
print(num_error)
#Check datatype
print(df.dtypes)
#check outliers
interest = ["InboundRatio", "InternationalStudentsNO", "KOFPoGI", "KOFEcGI",␣
↪"KOFSoGI", "top_50_count",
                   "top_100_count", "top_500_count", "top_1000_count"]
for i in interest:
    df.boxplot(i)
    plt.title(i)
    plt.tight_layout()
    plt.show()
```

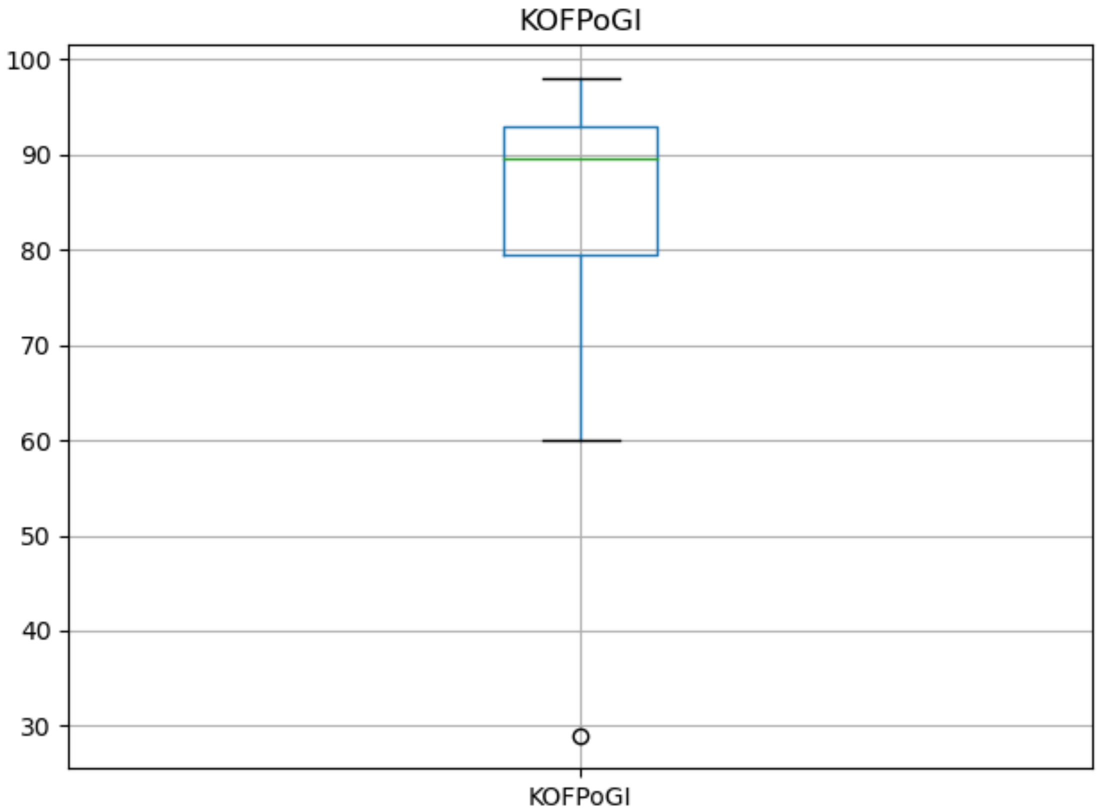
```
# check normalization
df[["InboundRatio","InternationalStudentsNO","KOFPoGI","KOFEcGI","KOFSoGI","ISCED5␣
↪Percentage","ISCED6 Percentage","ISCED7 Percentage","ISCED8 Percentage",
    "top_50_count","top_100_count","top_500_count","top_1000_count"]].describe()
```

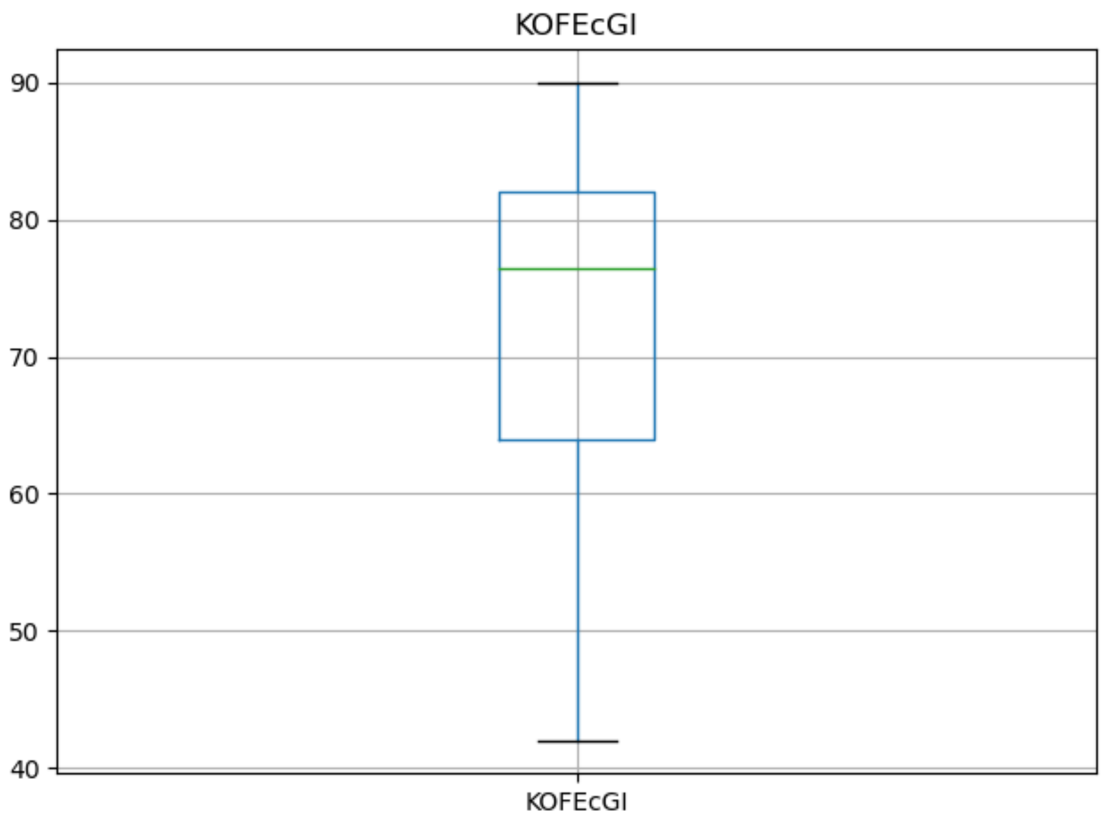
```
Number of observation:  49
Any NA value: True
Any row duplictaes: False
Tertiary Percentage              0
ISCED5 Percentage                0
ISCED6 Percentage                0
ISCED7 Percentage                0
ISCED8 Percentage                0
year                             0
InternationalStudentsNO          0
KOFGI                            0
KOFGIdf                          0
KOFGIdj                          0
KOFPoGI                          0
KOFPoGIdf                        0
KOFPoGIdj                        0
KOFSoGI                          0
KOFSoGIdf                        0
KOFSoGIdj                        0
KOFInGI                          0
KOFInGIdf                        0
KOFInGIdj                        0
KOFIpGI                          0
KOFIpGIdf                        0
KOFIpGIdj                        0
KOFCuGI                          0
KOFCuGIdf                        0
KOFCuGIdj                        0
KOFEcGI                          0
KOFEcGIdf                        0
KOFEcGIdj                        0
KOFTrGI                          0
KOFTrGIdf                        0
KOFTrGIdj                        0
KOFFiGI                          0
KOFFiGIdf                        0
KOFFiGIdj                        0
KOFSoGI_WithoutInterpersonal     0
InboundRatio                     0
top_50_count                     0
top_100_count                    0
top_500_count                    0
top_1000_count                   0
```

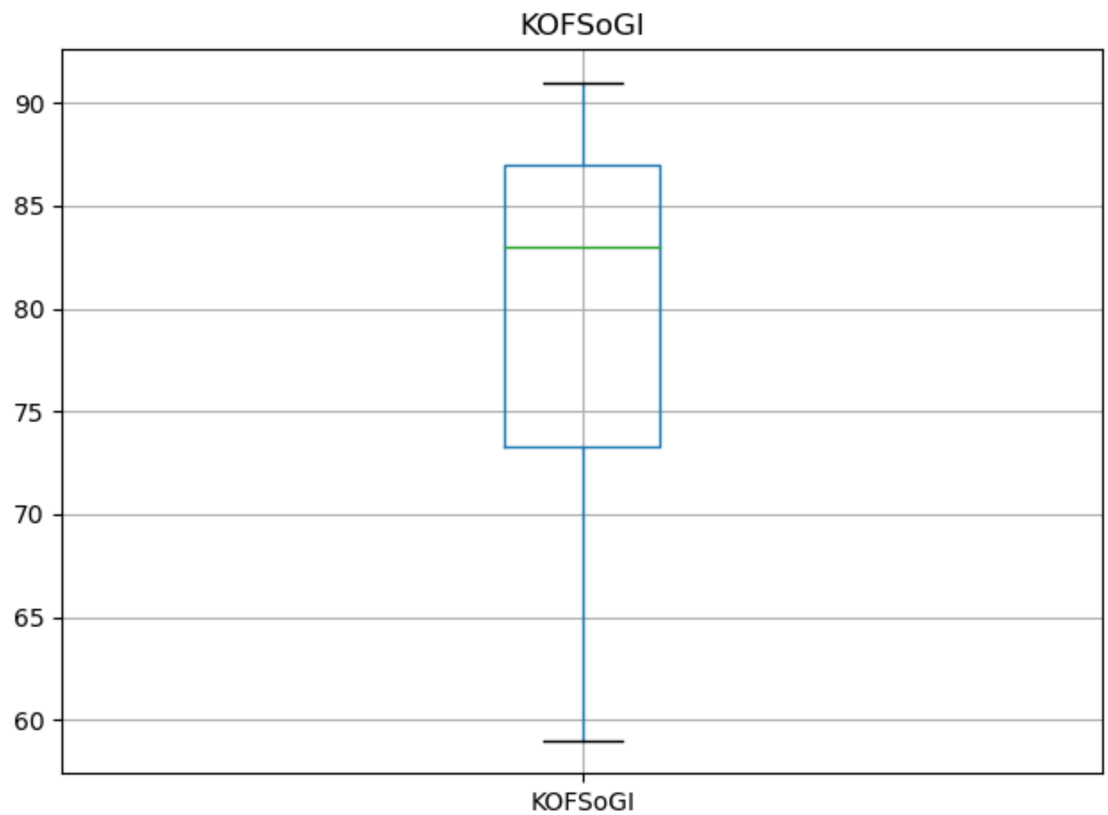
```
total_ranked_universities       0
dtype: int64
country_x                        object
code                             object
Tertiary Percentage             float64
ISCED5 Percentage               float64
ISCED6 Percentage               float64
ISCED7 Percentage               float64
ISCED8 Percentage               float64
country_y                        object
year                              int64
InternationalStudentsNO           int64
KOFGI                             int64
KOFGIdf                           int64
KOFGIdj                           int64
KOFPoGI                           int64
KOFPoGIdf                         int64
KOFPoGIdj                         int64
KOFSoGI                           int64
KOFSoGIdf                         int64
KOFSoGIdj                         int64
KOFInGI                           int64
KOFInGIdf                         int64
KOFInGIdj                         int64
KOFIpGI                           int64
KOFIpGIdf                         int64
KOFIpGIdj                         int64
KOFCuGI                           int64
KOFCuGIdf                         int64
KOFCuGIdj                         int64
KOFEcGI                           int64
KOFEcGIdf                         int64
KOFEcGIdj                         int64
KOFTrGI                           int64
KOFTrGIdf                         int64
KOFTrGIdj                         int64
KOFFiGI                           int64
KOFFiGIdf                         int64
KOFFiGIdj                         int64
KOFSoGI_WithoutInterpersonal    float64
InboundRatio                    float64
top_50_count                      int64
top_100_count                     int64
top_500_count                     int64
top_1000_count                    int64
total_ranked_universities         int64
WESP                             object
dtype: object
```

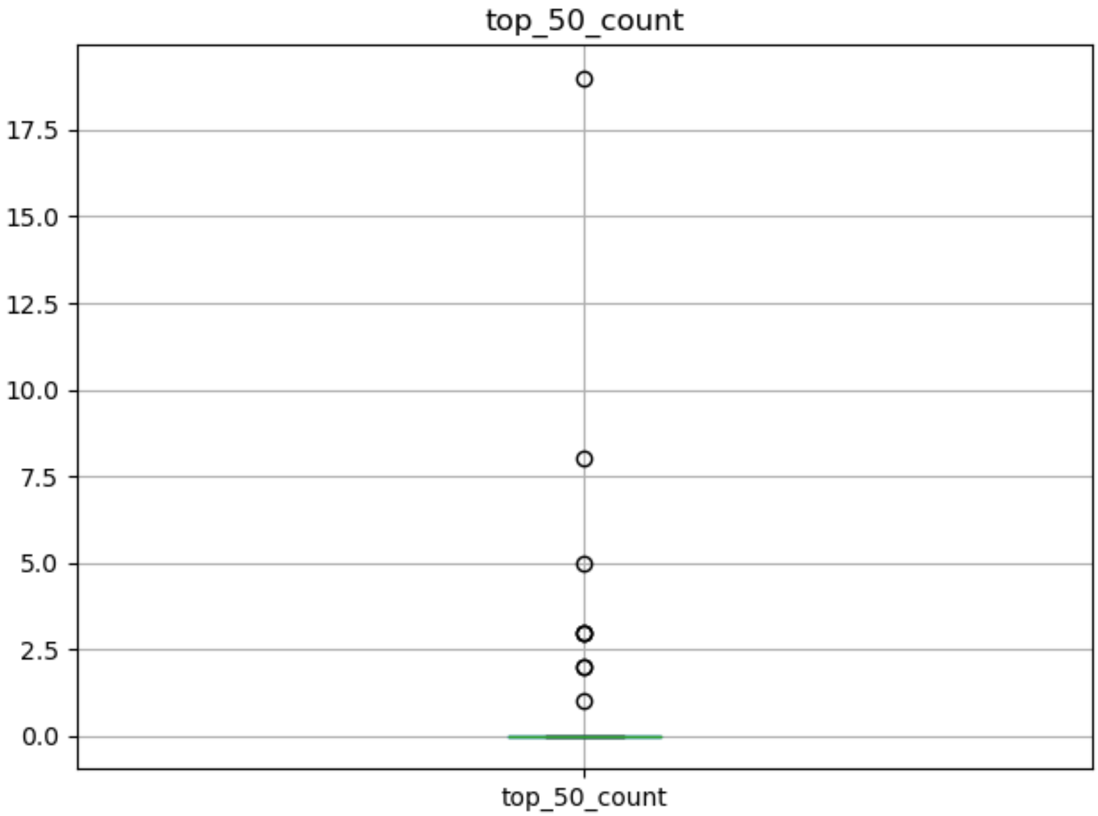

InboundRatio
35
30
25
20
15
10
5
0
InboundRatio


InternationalStudentsNO
1e6
1.0
0.8
0.6
0.4
0.2
0.0
InternationalStudentsNO

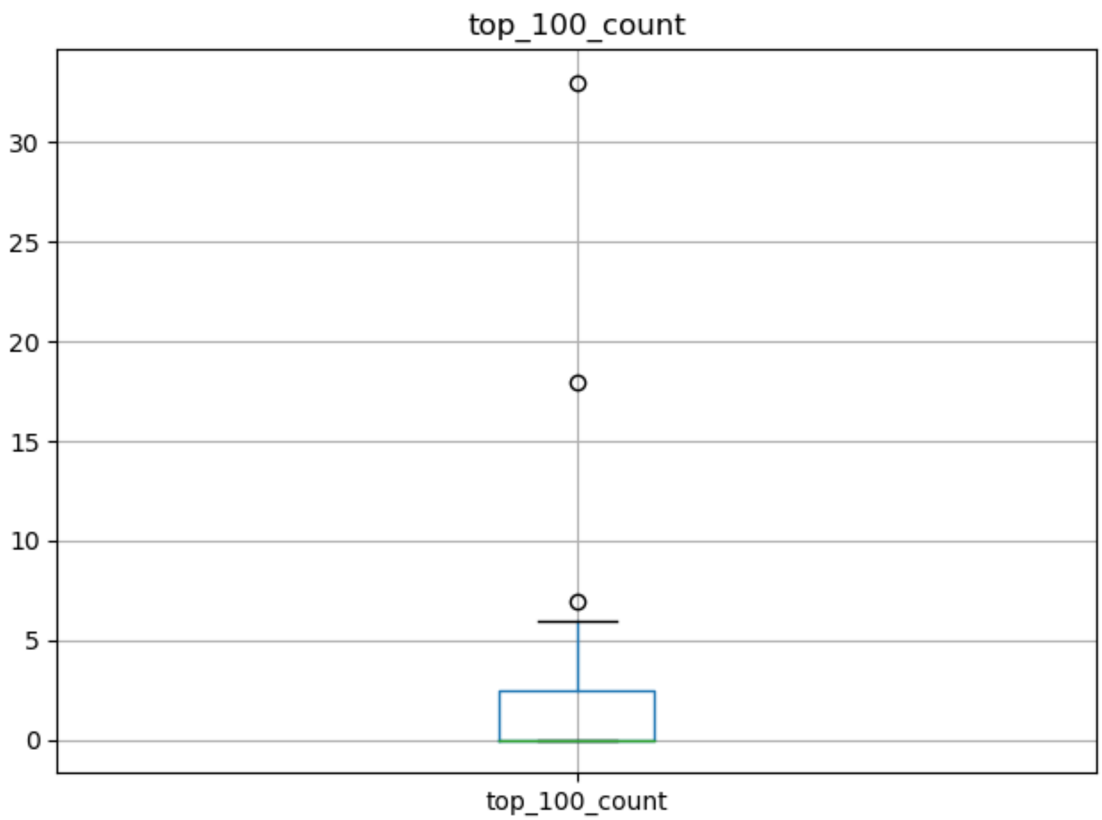

KOFPoGI
100
90
80
70
60
50
40
30
KOFPoGI

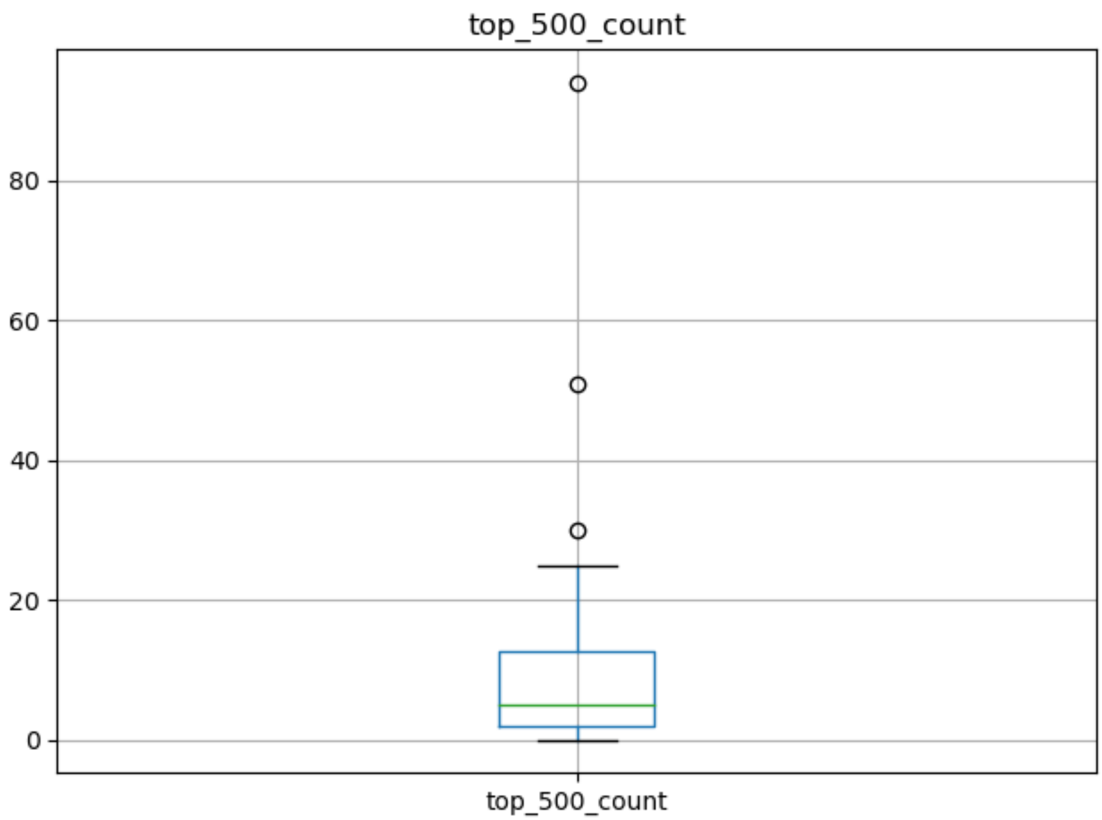

KOFEcGI
90
80
70
60
50
40
KOFEcGI

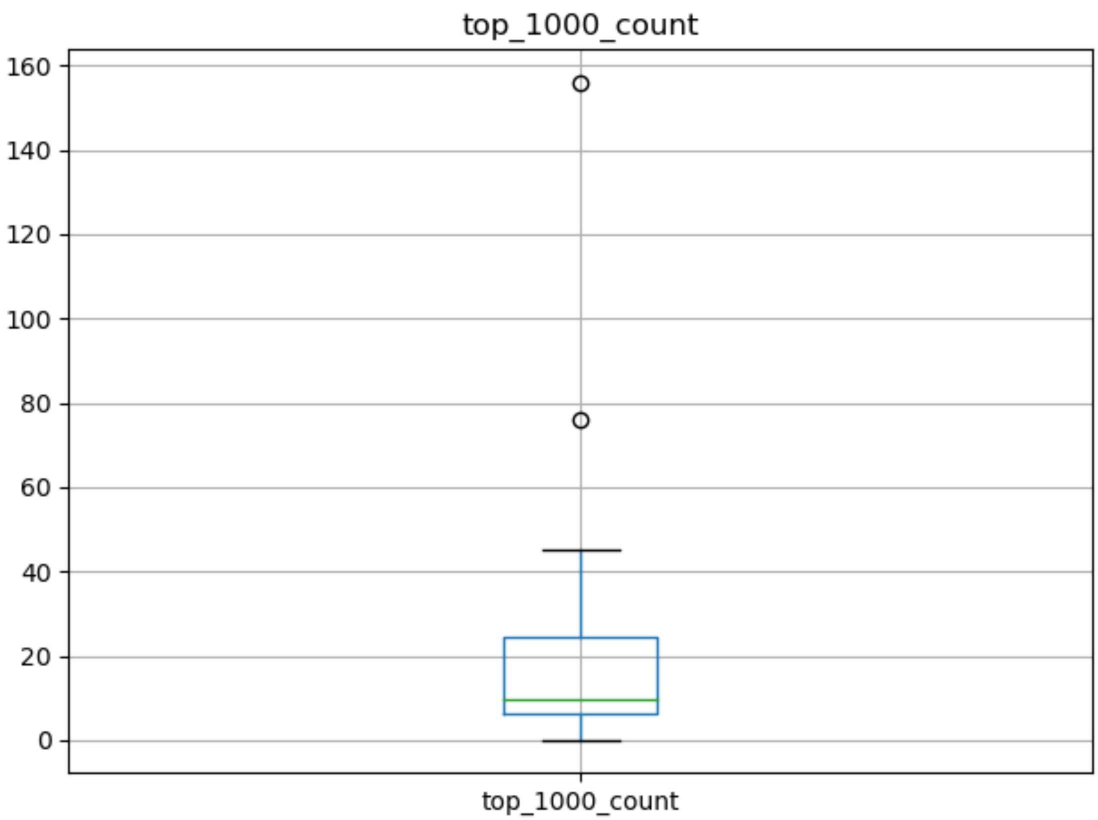

KOFSoGI
90
85
80
75
70
65
60
KOFSoGI



top_50_count
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
top_50_count


top_100_count
30
25
20
15
10
5
0
top_100_count


top_500_count
80
60
40
20
0
top_500_count



[11]:

	InboundRatio	InternationalStudentsNO	KOFPoGI	KOFEcGI	KOFSoGI	\
count	42.000000	42.000000	42.000000	42.000000	42.000000	
mean	9.368033	117317.380952	84.952381	71.976190	79.976190	
std	8.016693	183894.022375	13.510524	12.994348	9.358674	
min	0.219050	1546.000000	29.000000	42.000000	59.000000	
25%	3.549540	22034.250000	79.500000	64.000000	73.250000	
50%	7.800560	49007.000000	89.500000	76.500000	83.000000	
75%	12.455103	114335.750000	93.000000	82.000000	87.000000	
max	35.293780	976562.000000	98.000000	90.000000	91.000000	

	ISCED5 Percentage	ISCED6 Percentage	ISCED7 Percentage	\
count	42.000000	42.000000	42.000000	
mean	10.626414	45.236110	14.233167	
std	9.801015	13.083961	8.697049	
min	0.004350	12.319206	1.083925	
25%	2.523087	38.851575	6.738658	
50%	8.476903	44.474409	14.806317	
75%	16.899843	54.239022	21.464752	
max	41.863344	68.238077	35.507974	

	ISCED8 Percentage	top_50_count	top_100_count	top_500_count	\

```
count      42.000000   42.000000   42.000000   42.000000
mean        2.098529    1.095238    2.261905   10.214286
std         1.353961    3.259579    5.793434   16.543418
min         0.000000    0.000000    0.000000    0.000000
25%         0.804222    0.000000    0.000000    2.000000
50%         2.085667    0.000000    0.000000    5.000000
75%         2.887539    0.000000    2.500000   12.750000
max         5.152113   19.000000   33.000000   94.000000

       top_1000_count
count       42.000000
mean        18.642857
std         26.709660
min          0.000000
25%          6.250000
50%          9.500000
75%         24.250000
max        156.000000
```

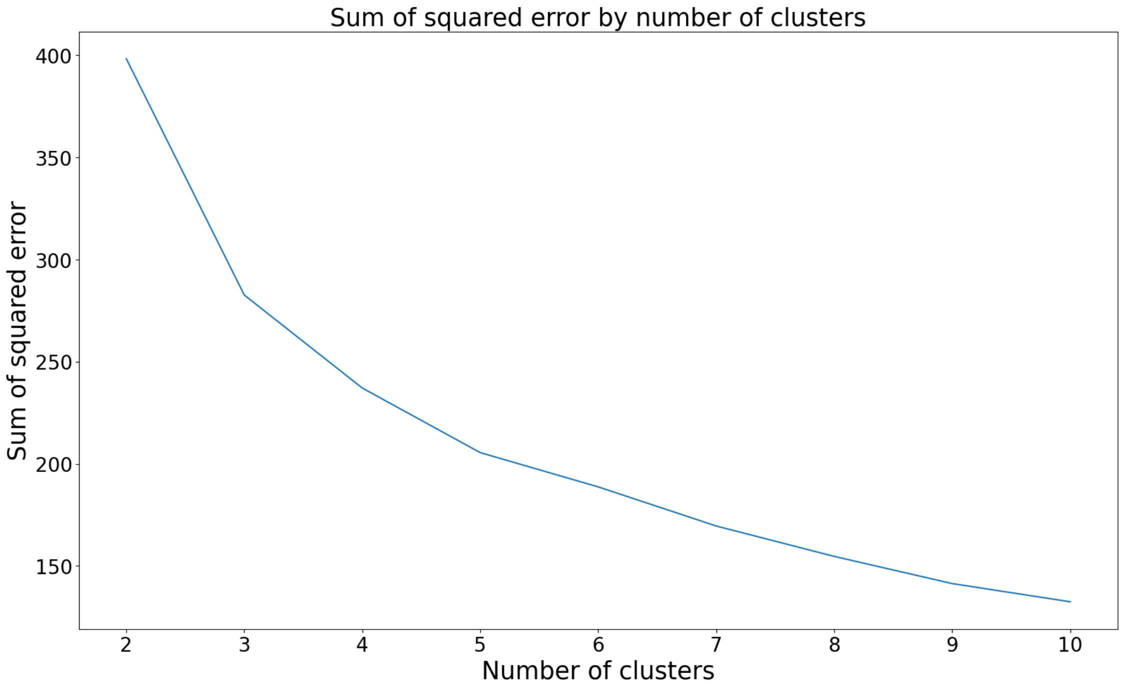
In this dataset, we have 49 observations with missing values and no row duplicates. There is no negative value in the numeric variables. We also observe a few outliers in those numeric variables, as indicated by box plots. Since the standard deviation of those variables is quite different, we have to standardize them(This is done in later parts), which is a crucial step for cluster analysis as the distance between data points is a major determinant. Variables on different scales will result in a bias for cluster analysis. Data balancing is not needed for this cluster analysis question. We begin our analysis by dropping all the NA values.

2 Question3 b

[12]:
```
# Standardising all continious variables
variables_ofstd =␣
 ↪["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"];
std= df[variables_ofstd];
scaler = StandardScaler(); # creating object
fitted = scaler.fit(std);
df_std  = pd.DataFrame(fitted.transform(std))
```

[13]:
```
# Elbow method
np.random.seed(1)
from sklearn.cluster import KMeans
def wcss(x, kmax):#wcss Function: The wcss function calculates the␣
 ↪within-cluster sum
#of squares (WCSS) for different numbers of clusters.
    wcss_s = [] #wcss_s: This list will store the WCSS values for different␣
 ↪numbers of clusters.
```

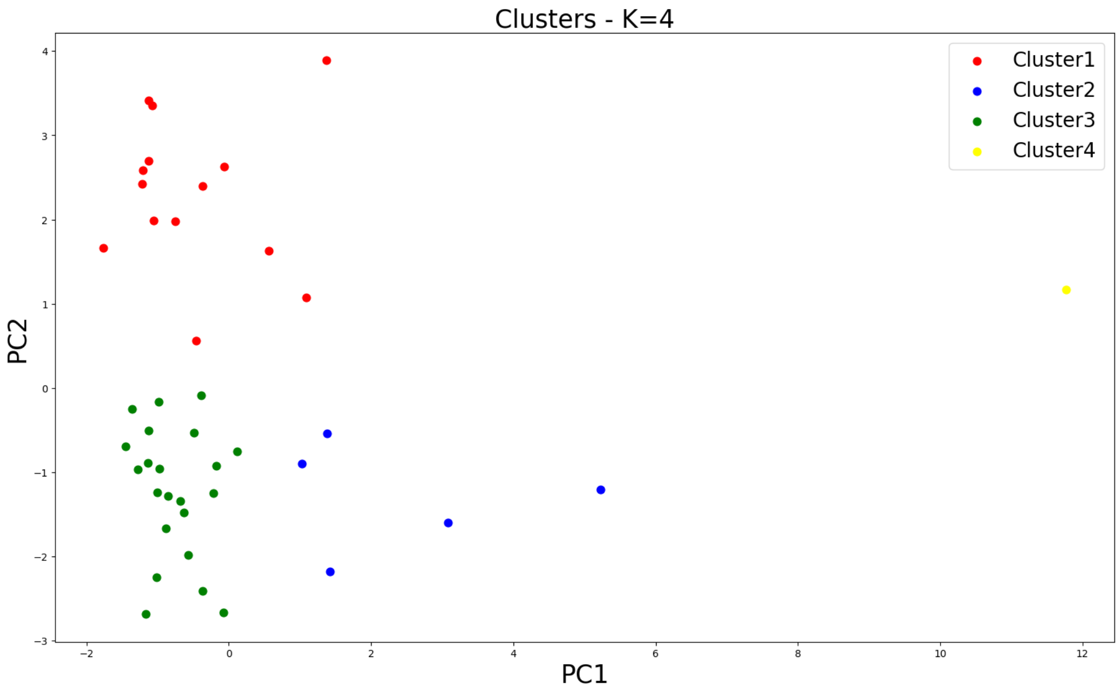
```
    for k in range(2, kmax + 1):
        kmeans = KMeans(n_clusters = k);
        kmeans.fit(x);
        wcss_s.append(kmeans.inertia_);# sample distances to closest cluster␣
 ↪center
    return wcss_s
# Plot
from matplotlib import pyplot as plt
fig = plt.figure(figsize = (19,11));
ax = fig.add_subplot(1,1,1);
kmax = 10; # maximum number of clusters
ax.plot(range(2, kmax + 1), wcss(df_std, kmax));
ax.tick_params(axis="both", which="major", labelsize=20);
ax.set_xlabel("Number of clusters", fontsize = 25);
ax.set_ylabel("Sum of squared error", fontsize = 25);
ax.set_title("Sum of squared error by number of clusters", fontsize = 25);
plt.show();
#Silihouse score
np.random.seed(100)
def Silhouette(x, kmax):
    sil = []
    for k in range(2, kmax+1):
        kmeans = KMeans(n_clusters = k).fit(x)
        sil.append(silhouette_score(x, kmeans.labels_, metric = "euclidean"))
    return sil
# Plot
fig = plt.figure(figsize = (19,11));
ax = fig.add_subplot(1,1,1);
ax.plot(range(2,kmax+1) , Silhouette(df_std,kmax));
ax.tick_params(axis="both", which="major", labelsize=20);
ax.set_xlabel("Number of clusters", fontsize = 25);
ax.set_ylabel("Silhouette score", fontsize = 25);
ax.set_title("Silhouette score by number of clusters", fontsize = 25);
ax.xaxis.set_major_locator(MaxNLocator(integer=True)) # to force intergers in␣
 ↪x-axis
plt.show();
```



It is not quite intuitive where the elbow effect happens, but it could be at K=3, 4, or 5 since the SSE decreased quite significantly with respect to lower K values. For K values greater than 5, the SSE decreases, but not as dramatically. The Silhouette scores plot favors K=2 and 4, but the SSEs for

K=2 are too high in the Elbow method. Therefore, we propose K=4(four clusters for the dataset). We can reduce the dimensionality of the data via PCA for visual inspection(See later parts).

[14]:
```
# We now perform visiual inspection via reducing dimesionality (PCA)
from sklearn.decomposition import PCA
pca = PCA(n_components=2);# First two components
principalComponents = pca.fit_transform(df_std);
PCs = pd.DataFrame(data = principalComponents, columns = ["PC1", "PC2"]);
kmeans = KMeans(n_clusters = 4, init = "k-means++", random_state = 42);
y_kmeans = kmeans.fit_predict(df_std); # predictions of clusters
# Plotting PCs
fig = plt.figure(figsize = (19,11));
ax = fig.add_subplot(1,1,1);
plt.scatter(PCs.iloc[y_kmeans == 0, 0], PCs.iloc[y_kmeans == 0, 1], s=60,
c="red", label = "Cluster1");
plt.scatter(PCs.iloc[y_kmeans == 1, 0], PCs.iloc[y_kmeans == 1, 1], s=60,
c="blue", label = "Cluster2");
plt.scatter(PCs.iloc[y_kmeans == 2, 0], PCs.iloc[y_kmeans == 2, 1], s=60,
c="green", label = "Cluster3");
plt.scatter(PCs.iloc[y_kmeans == 3, 0], PCs.iloc[y_kmeans == 3, 1], s=60,
c="yellow", label = "Cluster4");
plt.xlabel("PC1", fontsize = 25);
plt.ylabel("PC2", fontsize = 25);
ax.set_title("Clusters - K=4", fontsize = 25);
plt.legend(fontsize = 20);
plt.show();
# Total variability explained by first two
print(" The variability explained by first two principal components is " +␣
 ↪str(np.sum(pca.explained_variance_ratio_)*100) + "%")
```




```
The variability explained by first two principal components is
64.97013208955245%
```

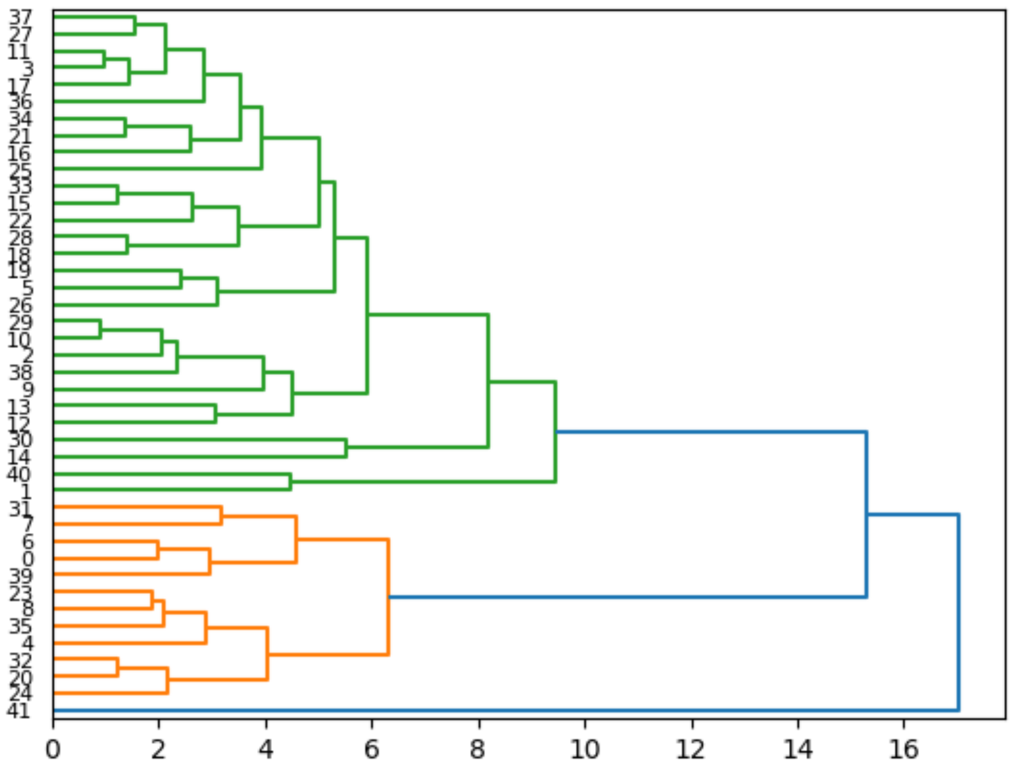
[15]:
```
# print out countries in differnt clusters.
df["Cluster_Kmean"] = pd.DataFrame(y_kmeans);
print("Cluster 1:\n", list(df["country_x"][(df["Cluster_Kmean"]==0)]));
print("Cluster 2:\n", list(df["country_x"][(df["Cluster_Kmean"]==1)]));
print("Cluster 3:\n", list(df["country_x"][(df["Cluster_Kmean"]==2)]));
print("Cluster 4:\n", list(df["country_x"][(df["Cluster_Kmean"]==3)]));
```

```
Cluster 1:
 ['Argentina', 'Brazil', 'Chile', 'China', 'Colombia', 'Japan', 'Kazakhstan',
'Malaysia', 'Mexico', 'Mongolia', 'Russia', 'Saudi Arabia', 'South Africa',
'Turkey']
Cluster 2:
 ['Australia', 'Canada', 'France', 'Germany', 'United Kingdom']
Cluster 3:
 ['Austria', 'Belgium', 'Cyprus', 'Czech Republic', 'Denmark', 'Hong Kong',
'Hungary', 'Iceland', 'Ireland', 'Italy', 'Latvia', 'Netherlands', 'New
Zealand', 'Norway', 'Poland', 'Portugal', 'Qatar', 'Slovak Republic',
'Slovenia', 'Spain', 'Sweden', 'Switzerland']
Cluster 4:
 ['USA']
```

We could observe a clear separation between the clusters, and there is no overlap in the figure. It seems that $K = 4$ also represents the definition of the clusters quite well.The first 2 principal *components explain approx 65% of the variability of the data. More importantly, the 4 clusters are well-defined in the PC1 and PC2 scatter plots.*

3 Question3 c

[16]:
```
#Q3C agglomerative cluster analysis
from scipy.cluster.hierarchy import dendrogram, linkage;
dendrogram(linkage(df_std, method="ward"), orientation = "right",# Generating dendrogram
labels = None);
plt.show()
```

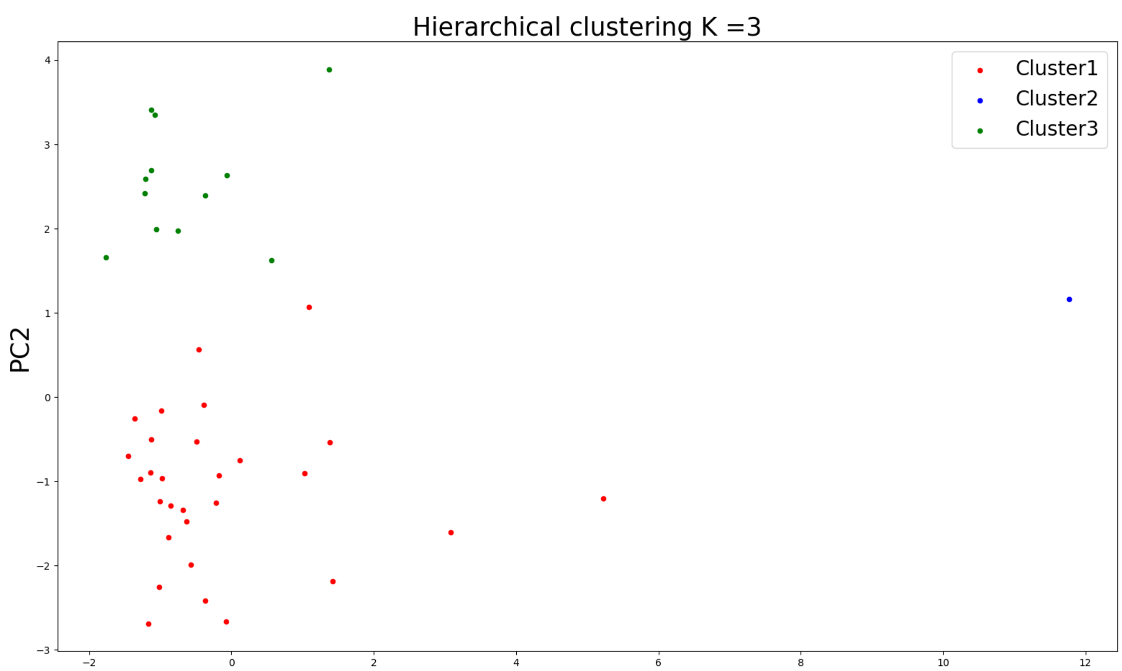


The largest distance can be found between approximately 9 and 15, generating 3 clusters(Vertical line at 9).Hence, we propose 3 clusters for this method.

[17]:
```
#We now perform visiual inspection via reducing dimesionality (PCA)
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters=3, linkage="ward",
compute_distances = True);
model.fit(df_std);
df["Cluster_Agg"] = pd.DataFrame(model.labels_);
clusters3 = model.labels_;
```

```
pca = PCA(n_components=2);
principalComponents = pca.fit_transform(df_std);
print("Variability explained by first 2 PCs: ", round(np.sum(pca.
    explained_variance_ratio_),2))
PCs = pd.DataFrame(data = principalComponents, columns = ["PC1", "PC2"]);

# Plotting PCs
fig = plt.figure(figsize = (19,11));
ax = fig.add_subplot(1,1,1);
plt.scatter(PCs.iloc[clusters3 == 0, 0], PCs.iloc[clusters3 == 0, 1], s=20,
    c="red", label = "Cluster1");
plt.scatter(PCs.iloc[clusters3 == 1, 0], PCs.iloc[clusters3 == 1, 1], s=20,
    c="blue", label = "Cluster2");
plt.scatter(PCs.iloc[clusters3 == 2, 0], PCs.iloc[clusters3 == 2, 1], s=20,
    c="green", label = "Cluster3");
plt.ylabel("PC2", fontsize = 25);
ax.set_title("Hierarchical clustering K =3", fontsize = 25);
plt.legend(fontsize = 20);
plt.show();
```

```
Variability explained by first 2 PCs:  0.65
```



Since we can not plot all the variables simultaneously, we would reduce the dimensionality of the dataset through PCA. The first 2 principal components explain 65% of the variability of the data. More importantly, the 3 clusters are well-defined in the PC1 and PC2 scatter plots.

4 Question3 d

[18]:
```
# print out countries in differnt clusters to describe.
print("Cluster 1:\n", list(df["country_x"][(df["Cluster_Agg"]==0)]));
print("Cluster 2:\n", list(df["country_x"][(df["Cluster_Agg"]==1)]));
print("Cluster 3:\n", list(df["country_x"][(df["Cluster_Agg"]==2)]));
```

```
Cluster 1:
 ['Australia', 'Austria', 'Belgium', 'Canada', 'Cyprus', 'Czech Republic',
'Denmark', 'France', 'Germany', 'Hong Kong', 'Hungary', 'Iceland', 'Ireland',
'Italy', 'Japan', 'Latvia', 'Malaysia', 'Netherlands', 'New Zealand', 'Norway',
'Poland', 'Portugal', 'Qatar', 'Slovak Republic', 'Slovenia', 'Spain', 'Sweden',
'Switzerland', 'United Kingdom']
Cluster 2:
 ['USA']
Cluster 3:
 ['Argentina', 'Brazil', 'Chile', 'China', 'Colombia', 'Kazakhstan', 'Mexico',
'Mongolia', 'Russia', 'Saudi Arabia', 'South Africa', 'Turkey']
```

The analysis of K-mean and agglomerative cluster analysis suggested a slightly different number of clusters (4 and 3), indicating that globalization of the country and education system are indeed quite complex as there is a lot factors influencing them. Interestingly, the USA itself was identified as one cluster in this case, meaning that the USA has unique characteristics not shared by other countries, as well as the dominance of the USA, such as a higher globalization index, international students' mobility, economic globalization index, etc. Moreover, Countries like Australia, Canada and France were in one cluster alone by the K-mean method; then they were put in the same cluster as countries like Spain and Japan. This potentially implies that those emerging countries are getting better and better at their education system and attracting more international students to come to their country.This agrees with what we have from the scatter plot, as we could observe a point on the far right end(USA).