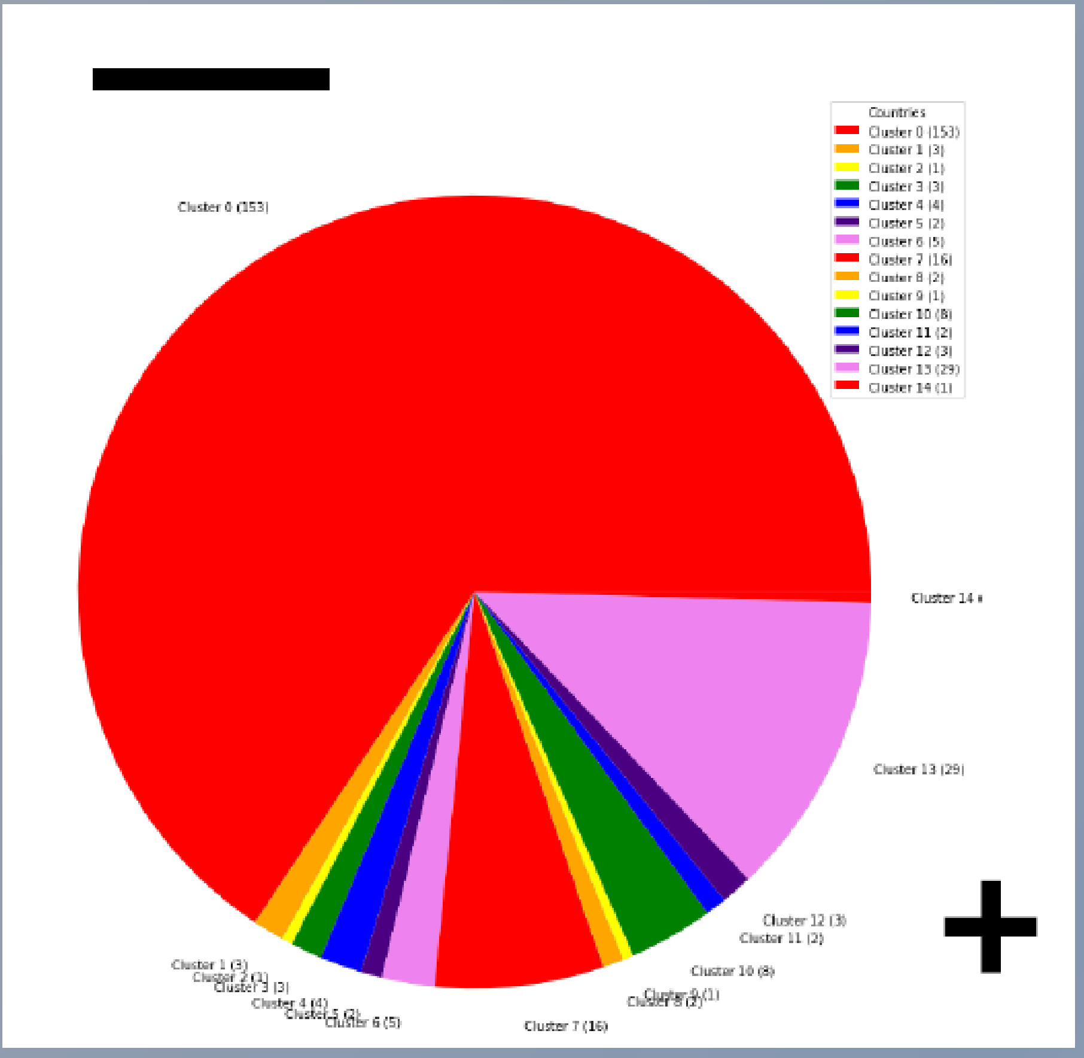
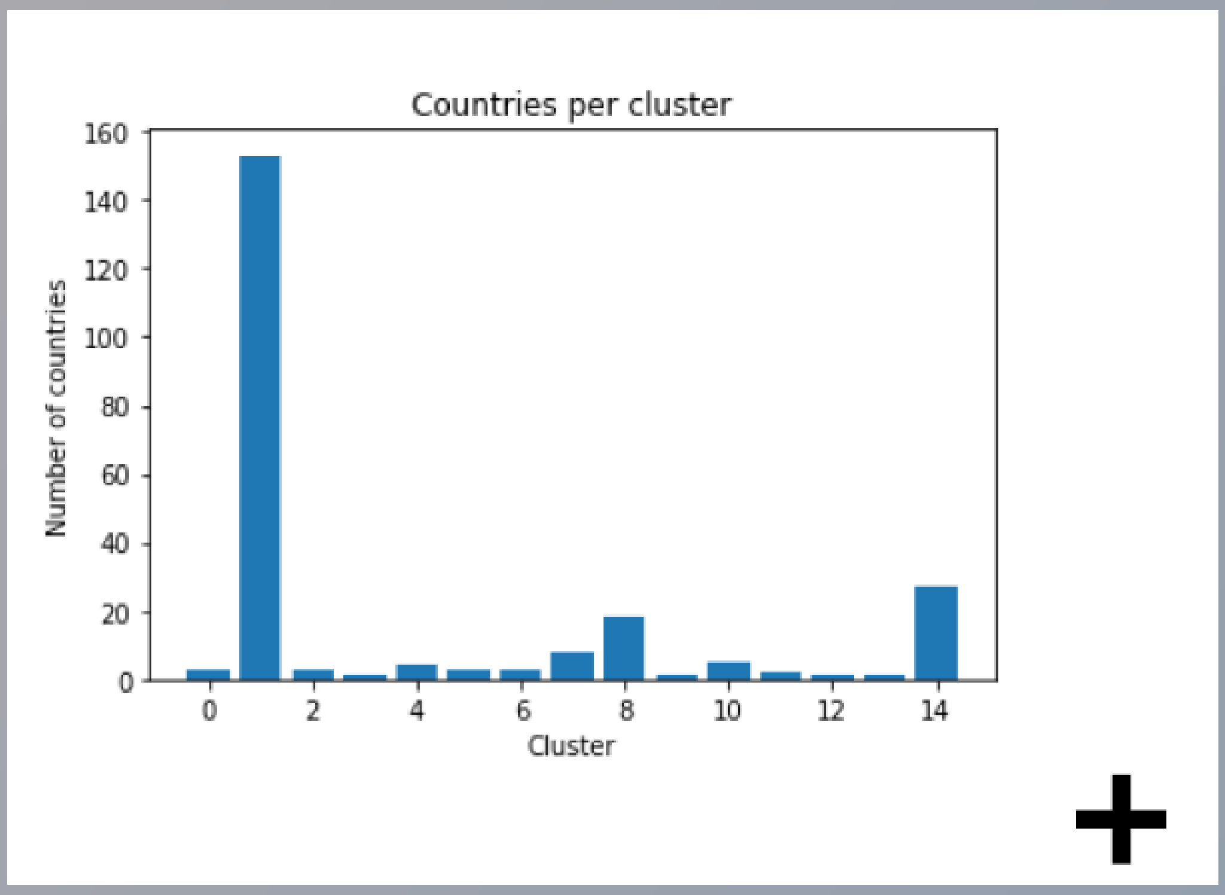
Project filtering and clusriting

The code above is performing the following steps: 1. Selecting the columns to cluster. The relevant columns are selected and stored in the X variable. These columns contain the CO2 emissions values for the years 2012 to 2019. 2. Scaling the data: The data is scaled using the StandardScaler method from scikit-learn. Scaling the data is important because the columns have different scales, which can affect the performance of the clustering algorithm. 3. Running the K-Means algorithm: The KMeans algorithm from scikit-learn is initialized with n_clusters-15 and then fit to the scaled data. 4. Getting the cluster labels: The cluster labels for each row in the data are obtained using the labels_attribute of the kmeans object. 5. Adding the cluster labels to the dataframe: The cluster labels are added as a new column to the dataframe. 6. Grouping the data by cluster and country: The data is grouped by both the cluster label and the country name. The resulting object is a Pandas DataFrameGroupBy object, which allows us to perform various operations on the grouped data.

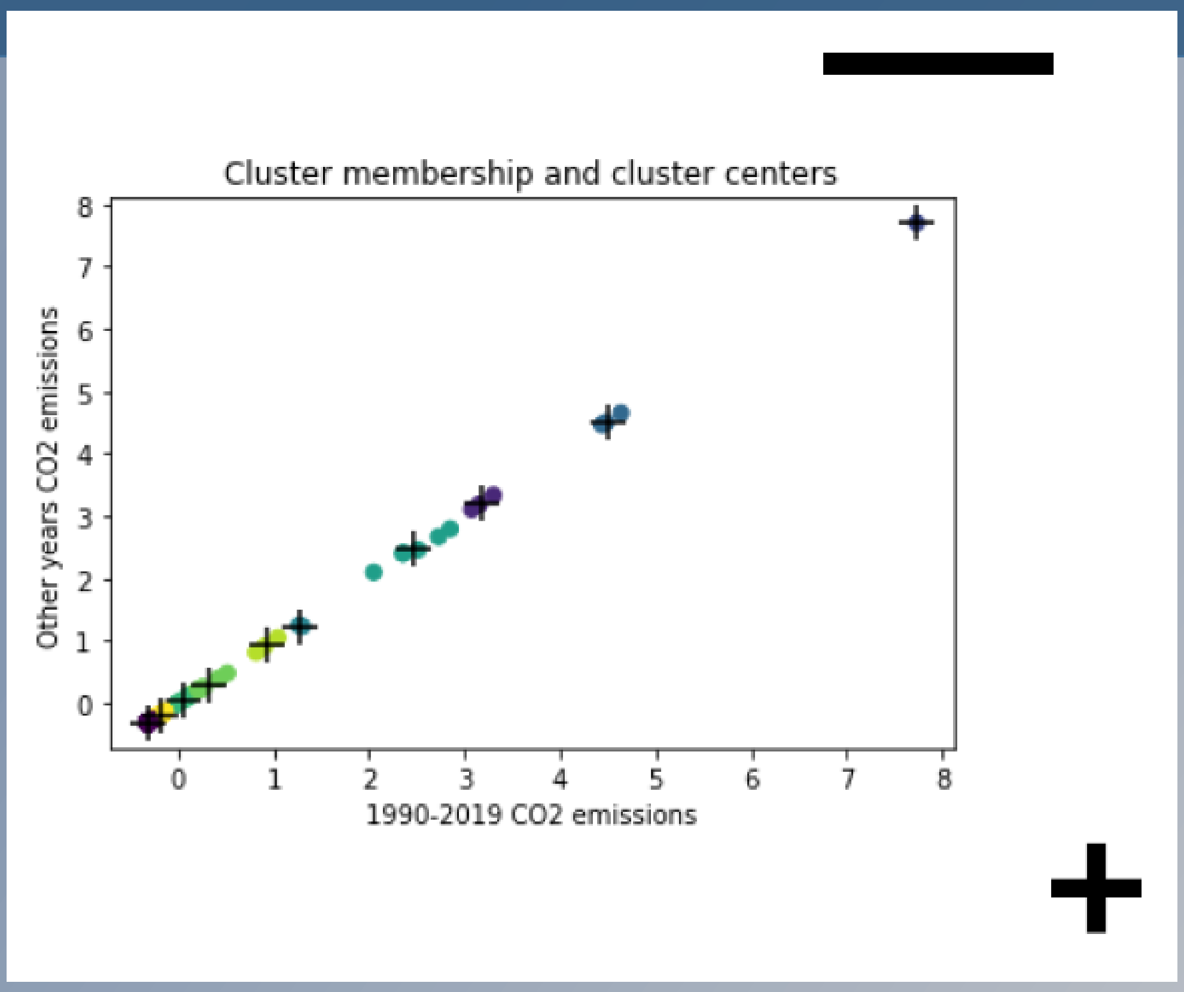


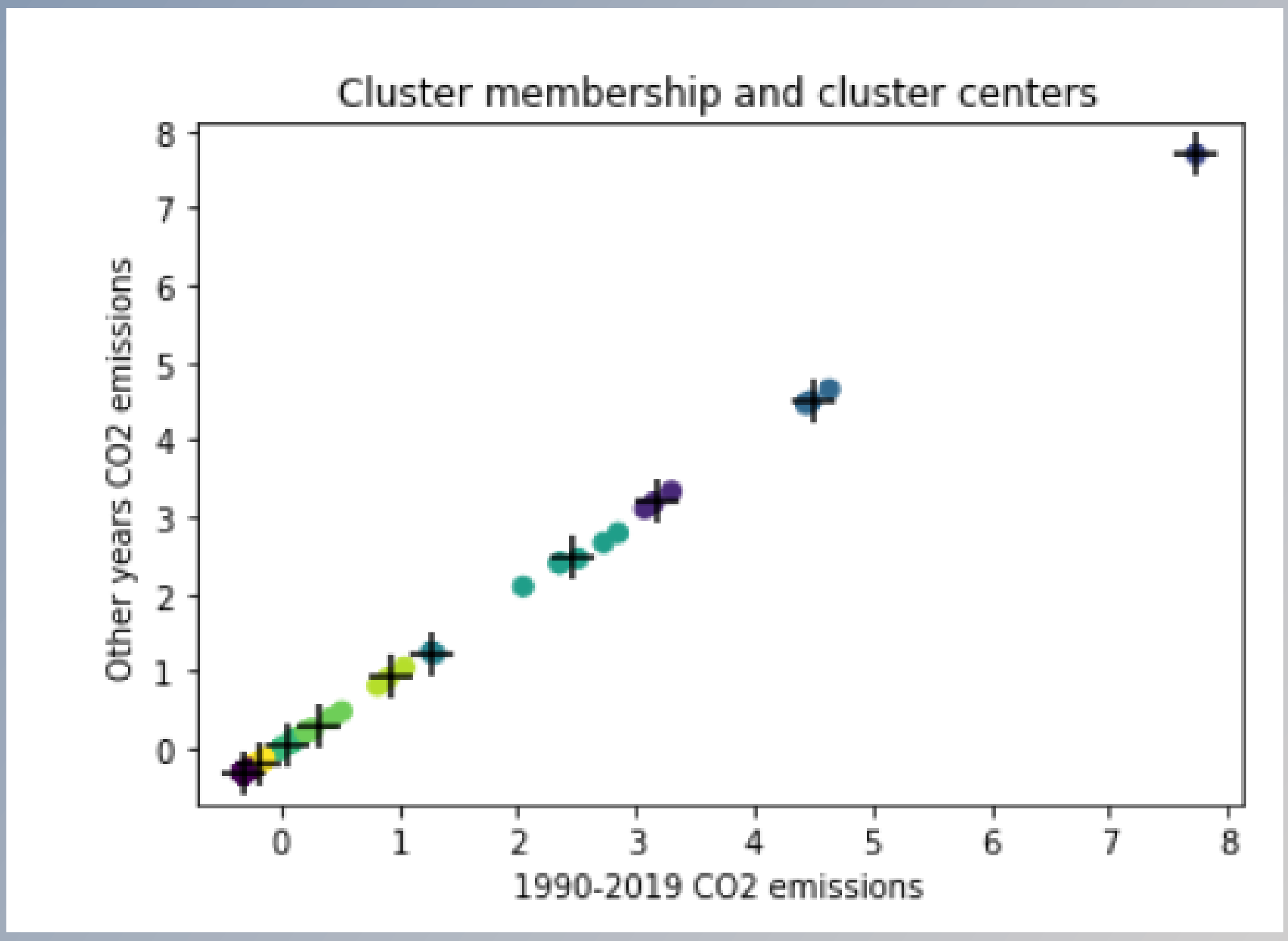
This code generates a pie chart that shows the distribution of countries among different clusters. The data is first grouped by cluster label using the groupbyfunction from the Pandas library. The resulting object is a dictionary-like object that maps cluster labels to groups of data. The code then iterates over this object and appends the size of each group (i.e., the number of countries in each cluster) to the values list. The labels for each cluster are also added to the labels list. Finally, a pie chart is created using the pie function from the Matplotlib library. The values and labels are passed as arguments to this function, and the resulting pie chart is displayed using the show function.

This code creates a bar chart showing the number of countries in each cluster. The grouped variable is a Pandas GroupByobject that has been grouped by the 'cluster' column of the data. The size() method is used to get the number of rows in each group. The cluster labels and country counts are extracted from the resulting Pandas Series object using the index.get_level_values() and values attributes, respectively. The bar chart is then created using the bar() function, with the cluster labels as the x-axis and the country counts as the y-axis. The x-axis label, y-axis label, and chart title are set using the xlabel(), ylabel(), and title() functions, respectively. Finally, the chart is displayed using the show() function.



In the previous code block, we are performing K-Means clustering on a dataset containing CO2 emission data from various countries. The data has been scaled using the StandardScalerto ensure that all features are on the same scale. The KMeansalgorithm is then applied to the scaled data, with the number of clusters specified as 10.



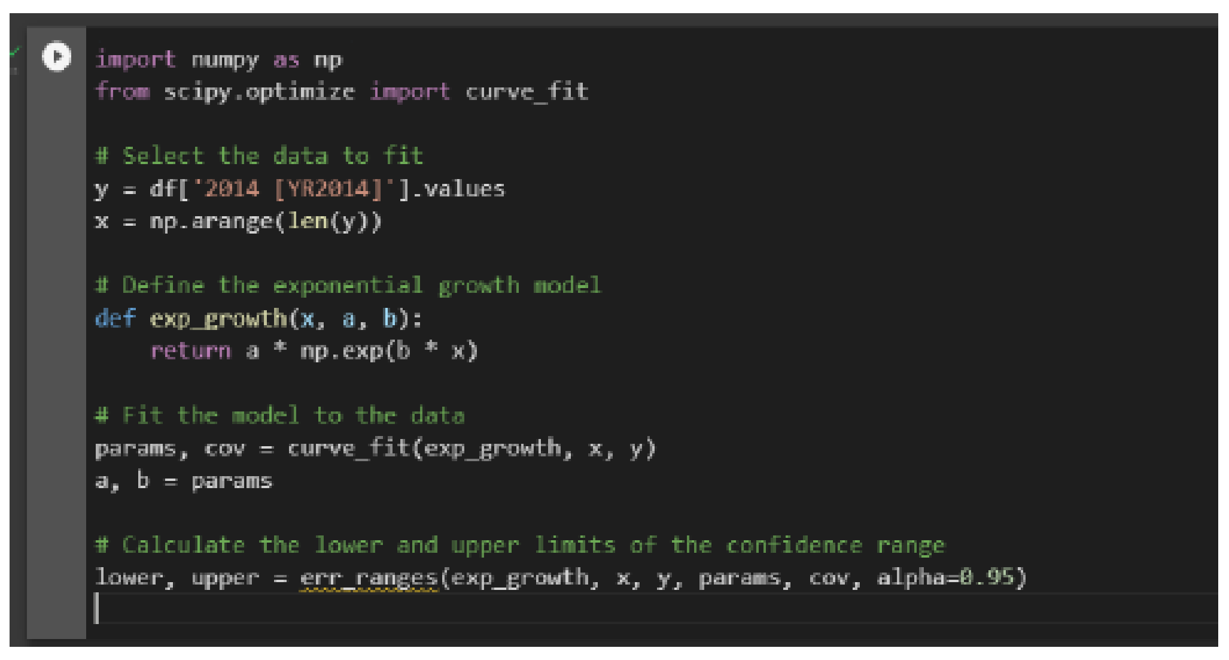


The plan:

We also plot the cluster centersas '+' signs on the scatter plot. The cluster centersare the mean of all the data points in the cluster, and they represent the center of each cluster.
–Finally, we add labels to the x-axis, y-axis, and the title of the plot. The resulting plot shows the cluster membership of each data point, as well as the location of the cluster centers.

The code imports the curve_fitfunction from the scipylibrary and the numpy library. It then selects the data to fit by assigning the values in the '2014 [YR2014]' column of the dfdataframe to the y variable and creating an array of integers from 0 to the length of y and assigning it to the x variable.
Next, the code defines the exponential growth model as a function called exp_growth. This function takes in an array x and two parameters a and b and returns the product of a and e raised to the power of b times x.

The code then fits the model to the data by calling the curve_fitfunction and passing in the exp_growthfunction and the x and y data as arguments. The function returns two variables, params and cov, which are assigned to the variables a, b, and covrespectively.
–Finally, the code calculates the lower and upper limits of the confidence range by calling the err_rangesfunction and passing in the exp_growthfunction, the x and y data, the params and covvariables, and a value of 0.95 for the alpha parameter. The function returns two variables, lower and upper, which are assigned to the variables of the same name.


```
import numpy as np
from scipy.optimize import curve_fit

# Select the data to fit
y = df['2014 [YR2014]'].values
x = np.arange(len(y))

# Define the exponential growth model
def exp_growth(x, a, b):
    return a * np.exp(b * x)

# Fit the model to the data
params, cov = curve_fit(exp_growth, x, y)
a, b = params

# Calculate the lower and upper limits of the confidence range
lower, upper = err_ranges(exp_growth, x, y, params, cov, alpha=0.95)
```

References:

[1] - Observadores De Supernovas, accessed: 24/11/2015. Fig.4 obtained by Joan Piggy.
<https://sites.google.com/site/obsdesn/home/sn2014j>
[2] - Supernova image: Bayfordbury archive – M82, B filter, 240s and 300s exposure times, Nov. 2013 and Feb 2014.
,